БЪЛГАРСКА АКАДЕМИЯ НА НАУКИТЕ
ИНСТИТУТ ПО ИНФОРМАЦИОННИ И КОМУНИКАЦИОННИ ТЕХНОЛОГИИ

Петър Росенов Томов

ПРОГНОЗИРАНЕ НА ВРЕМЕВИ РЕДОВЕ С ИЗКУСТВЕНИ НЕВРОННИ МРЕЖИ

ДИСЕРТАЦИЯ

за придобиване на образователната и научна степен „доктор“

по докторска програма 02.07.20 „Комуникационни мрежи и системи“

професионално направление 5.3 „Комуникационна и компютърна техника“

Научен ръководител:
проф. д-р Владимир Василев Монов

София – 2022 г.

Съдържание

Списък на съкращенията

Чуждоезичен термин	Съкращение	Български термин	Съкращение
Artificial Neural Network	ANN	Изкуствена невронна мрежа	ИНМ
Back Propagation	BP	Обратно разрпостранение на грешката	ОРГ
Differential Evolution	DE	Еволюция на разликите	ЕР
Donated Distributed Computing	DDC	Дарени разпределени изчисления	ДРИ
Evolutionary Algorithms	EAs	Еволюционни алгоритми	ЕА
Evolution Strategy	ES	Еволюционна стратегия	ЕС
Genetic Algorithms	GAs	Генетични алгоритми	ГА
Learning Rate	LR	Норма за обучение	НО
Moving Averages	MA	Плъзгащи средни	ПС
Machine Learning	ML	Машинно самообучение	МС

Multi-Layer Perceptron	MLP	Многослоен перцептрон	МП
Multiobjective Evolutionary Algorithms	MOEA	Многокритериални еволюционни алгоритми	МЕА
Operating System	OS	Операционна система	ОС
Particle Swarm Optimization	PSO	Оптимизация с рояци от частици	ОРЧ
Patterns Recognition	PR	Разпознаване на образи	РО
Probabilistic Neural Networks	PNN	Вероятностни невронни мрежи	ВНМ
Support Vector Machines	SVM	Машини с поддържащи вектори	МПВ
Time Series	TS	Времеви редове	ВР
Volunteer Distributed Computing	VDC	Доброволчески разпределени изчисления	ДРИ

Списък на фигурите

Списък на таблиците

Списък на листингите

Увод

Ежедневно човек се сблъсква с много величини, които се променят постоянно. Често пъти те са случайни, като зависят от различни условия. Могат да имат циклично променящ се характер и да проявяват тенденция. Факторите, които влияят върху една величина са много и не могат да бъдат оценени. Затова те се разглеждат като статистически величини. Анализът на такива случайни процеси, с помощта на статистиката, обикновено се извършва, след като те се дискретизират и се представят като серия от данни. Въвежда се понятието период, който е постоянна величина и се нарича време на дискретизация. Дискретизираните случайни величини, които са подредени в хронологичен ред се наричат времеви редове. В ежедневието си човек се сблъсква с много времеви редове. Например, средните дневни температури, количествата продадени горива през годината, стойностите на валутите на финансовите пазари и други. В почти всички области на човешката дейност се наблюдават такива явления и затова те са предмет на изучаване и прогнозиране.

Изкуствените невронни мрежи постигат голяма популярност в последните пет десетилетия. Основното им предимство е възможността да възпроизвеждат нелинейни зависимости с помощта на примерни данни. Приложение намират като инструмент за класификация, разпознаване на образи и прогнозиране. При най-разпространения вариант, изкуствените невронни мрежи представляват насочени тегловни графи. Организацията е на слоеве, като информацията се предава от входния слой към изходния слой. Най-често възлите между отделните слоеве са пълно свързани, което означава, че всеки възел е свързан с всички други възли от съседния слой. Организацията на броя слоеве и колко възли да има във всеки слой е обект на емпи-

рично установяване и силно зависи от естеството на решаваната задача. Процесът на обучение най-често е с тренировъчни примери (обучение с учител) и целта е да се постигне такава оптимална стойност за теглата по ребрата на графа, така че изкуствената невронна мрежа максимално добре да извършва изчисленията, за които е предназначена. Веднъж обучени, изкуствените невронни мрежи са изключително бързо действащи. Тази тяхна характеристика ги прави особено желани в множество индустриални технически решения. Трудностите при употребата на изкуствени невронни мрежи са свързани с времето, необходимо за тяхното обучение. През десетилетията са разработени множество различни алгоритми за търсене на оптимални тегла в мрежата. Двете основни направления алгоритми са градиентни (точни числени алгоритми) и евристични (най-често стохастични с въведени емпирични правила). Ускоряването на процеса по обучение е основен проблем в практическата употреба на изкуствените невронни мрежи.

Цел на настоящия дисертационен труд е да се предложат хибридни алгоритми за ускоряване на обучението при изкуствени невронни мрежи от тип многослоен перцептрон за целите на прогнозирането на времеви редове. За реализиране на тази цел е необходимо да се изпълнят следните задачи:

* Да се направи обзорен анализ и класификация на алгоритмите за обучение на изкуствени невронни мрежи от тип многослоен перцептрон;

* Да се анализира възможността за комбиниране на различни алгоритми за реализиране на хибридно обучение на изкуствени невронни мрежи от тип многослоен перцептрон;

* Да се предложат алгоритми за обучение на изкуствени невронни мрежи от тип многослоен перцептрон в разпределена среда;

* Да се предложи подобрение с цел намаляване на времето за обучение на изкуствени невронни мрежи от тип многослоен перцептрон;

* Да се предложи софтуерна архитектура за реализиране на мобилни разпределени изчисления за прогнозиране;

* Да се направи програмна реализация на предложените хибридни алгоритми за обучение на изкуствени невронни мрежи от тип многослоен перцептрон с цел

доказване на тяхната работоспособност;

* Да се направи сравнителен анализ за ефективността на познатите алгоритми за обучение на изкуствени невронни мрежи от тип многослоен перцептрон.

Дисертационният труд е структуриран в увод, 4 глави, заключение, приноси, списък на публикациите, забелязани цитирания, декларация за оригиналност, библиография и 1 приложение.

В първа глава е направен обзорен анализ и класификация на широко използваните алгоритми за обучение на изкуствени невронни мрежи. Определени са предимствата и недостатъците на точните числени алгоритми и на евристичните алгоритми. Представени са възможностите за обучение на изкуствени невронни мрежи при последователни пресмятания, паралелни пресмятания и пресмятания в разпределена среда.

Във втора глава е изложена теорията за алгоритмите при обучението на изкуствени невронни мрежи от тип многослоен перцептрон. Предложени са модификации на някои от алгоритмите, които са приложими при прогнозирането на времеви редове. Приложените модификации се отнасят до: 1) определяне на теглата, чрез използване на генетичен алгоритъм, използващ операциите – селекция, кръстосване и мутация. Предложен и анализиран е нов оператор за селекция, основан на създаването на поколения при процедура за рекурсивно спускане. Изследвано е експериментално бързодействието на използваните евристични алгоритми; 2) апроксимиране на криви към множество точки – за инкрементална апроксимация на времеви редове се използват уравнение на права и ред от синус функции. Предложен е подход за пресмятане на коефициентите на синус функциите с оптимизатор, базиран на еволюция на разликите и рояк от частици; 3) обучението на изкуствена невронна мрежа – представен е разгърнат модел на обучението, което цели намиране на оптимални тегла за мрежа от тип трислоен перцептрон; 4) активационната функция в изкуствени невронни мрежи – предложена е алтернатива на производна за активационната функция в изкуствени невронни мрежи. Съществена особеност на предложената функция, е че показва обещаващи резултати по отношение на бързодействието и точността. За целите на числените тестване на предложените модификации е необходимо класифици-

ране според честотата на гласуване за всеки потребител и процент на успех на всеки подаден глас. Решаването на тази конкретна задача е реализирано чрез използване на самоорганизиращи се карти на Кохонен.

В трета глава е представена софтуерна архитектура, позволяваща реализация на избрани алгоритми и предложените модификации. За реализацията на софтуерната архитектура е предложен обектно-ориентиран модел, релационен модел, комуникационните протоколи и графичен потребителски интерфейс. Всички те основаващи се на подходящи структури от данни.

В четвърта глава е изложен сравнителен анализ на някои точни числени и евристични алгоритми. Описани са проведените експерименти и получените резултати. Анализирана е тяхната производителност и общо допусната грешка.

В заключението е направено обобщение на извършените изследвания. Посочени са също така и някои насоки за бъдещи изследвания, свързани с областта на приложението на изкуствените невронни мрежи.

Глава 1

Прогнозиране на времеви редове с помощта на машинно самообучение

1.1 Прогнозиране на времеви редове

Времевите редове са последователност от измервания, която следва строг ред във времето [80]. Измерванията най-често са на равни интервали, но не винаги това е възможно или рационално [94]. Изчислението на прогноза, базираща се на миналите стойности във времевия ред е своеобразна екстраполация [61]. При времеви редове, където нови стойности се отчитат твърде бързо във времето (например, едноминутен интервал на FOREX пазара) е нужно инструментът за прогнозиране да се обучава динамично, паралелно с режима за употреба [118]. В някои редки случаи, информацията за времевия ред трябва да бъде извлечена от съдържанието на неструктурирани документи [120]. В началото на 70-те години на XX век [42] се предлагат линейни модели за прогнозиране на времеви редове – авторегресионен (Autoregressive) и плъзгащи средни (Moving Averages) [102]. При авторегресионния модел се възприема, че прогнозната стойност е линейна комбинация от стойностите в миналото. При плъзгащите средни прогнозната стойност е функция на случайни смущения - смущения, които са повлияли на времевия ред. Между двата модела е възможна комбинация под формата на авторегресионна интегрирана плъзгаща

средна (Auto-Regressive Integrated Moving Average) [60]. При определянето на параметрите за авторегресията и интегрираната плъзгаща средна успешно могат да се приложат еволюционни алгоритми [22]. Линейните модели се оказват неприложими за множество времеви редове от реалната практика [10]. Изкуствените невронни мрежи се оказват един от подходите да се премине от линейност [129] към нелинейност в моделите. Трудностите при изкуствените невронни мрежи са свързани с по-големия брой параметри, които са трудни за определяне, като степен за обучение (learning rate) при обратно разпространение на грешката или размер на скрития слой [100]. В някои разработки се правят опити да се автоматизира определянето на параметрите [124]. Липсата на систематизиран подход за изграждане на изкуствени невронни мрежи [86] е пряко свързан с определянето на параметрите. Линейност и нелинейност могат да се съчетаят в хибридни реализации, базирани на плъзгащи средни и изкуствени невронни мрежи [130]. Недостатък при изкуствените невронни мрежи е нуждата от по-голям брой тренировъчни примери [70]. Освен броя на тренировъчните примери, изключително съществен параметър се явява размерът на входния слой, който определя прозореца от стойности в миналия периода от миналото [16]. С добавяне на обратни връзки в изкуствените невронни мрежи се постига ефект на кратковременна памет. Ефектът на кратковременната памет в изкуствените невронни мрежи често води до подобряване на получените прогнози [12]. Ускорение в процеса по обучението на изкуствени невронни мрежи може да се получи чрез използването на вероятностни невронни мрежи (Probabilistic Neural Networks) [62]. Комбинацията от линейни уравнения за постигането на разграничение при нелинейни класове се постига и чрез машина с поддържащи вектори (Support Vector Machines) [69]. Машините с поддържащи вектори се явяват подобрение на изкуствените невронни мрежи, обучавани по правилото за обратно разпространение на грешката. Времето за обучение се намалява, докато достоверността на прогнозите се увеличава [101]. Ефективността на машините с поддържащи вектори може да се подобри, когато се използват алгоритми за адаптиране на параметрите [11]. При времеви редове с ясно изразен тренд и сезонност [132], премахването им може да подобри възможностите за генериране на прогноза [131], включително и в комбинация с изкуствени невронни мрежи [49]. От

оригиналния времеви ред се изваждат стойностите на тренда и подбрани циклични функции [79] или уейвлети [50]. При слабо изразена сезонност, по-удачно може да се окаже сезонността да не се премахва [45]. Възможност за генериране на прогнози дават и алгоритмите за разпознаване на образи (Patterns Recognition). Времевият ред може да бъде изследван за наличието на конкретни шаблони/образи и прогнозата да се разпознава на появата на конкретен шаблон [96]. При много зашумени времеви редове е удачно да се приложи предварителна обработка на данните, така че шумът да се премахне или редуцира [73].

1.2 Обучение на изкуствени невронни мрежи

Най-разпространеният вид изкуствени невронни мрежи е многослойният перцептрон. Структурата им е под формата на насочен тегловен граф, който може да има различни конфигурации [25]. Организацията е на слоеве [48] (най-често три - входен, скрит и изходен [85]), като най-популярната практика е всички възли между два съседни слоя да са пълно свързани по между си. Пълната свързаност между слоевете понякога се нарушава с цел по-ефективно обучение [76]. Този тип мрежи се използват за решаването на задачи, в които са налични тренировъчни примери. Такъв тип обучение е известно като обучение с учител. Целта на изкуствената невронна мрежа е по входно-изходните двойки (тренировъчни примери) да формира функционална зависимост [55]. Този механизъм на работа прави многослойните перцептрони идеални за решаване на задачи за класификация [67] или прогнозиране [5]. Прогнозирането във финансовия свят е задача с голяма сложност при силна нелинейност, зависеща от много и различни фактори [9]. Процесът на самото обучение се състои в намирането на такива стойности за теглата в графа, че изкуствената невронна мрежа максимално добре да онагледява зависимостта между входните данни и изходните данни, без това да доведе до пренапасване (overfitting) [8]. Задачата е оптимизационна и е дефинирана в многомерни нелинейни пространства [63]. Най-често началните стойности на теглата се избират случайно в близка околност около нулата, но е възможно първоначалните стойности да се определят с някаква форма на евристика,

например генетични алгоритми [15]. Теглата на мрежата най-често участват в линейна комбинация с входните сигнали. Получената сума се подлага на нормиране, чрез прилагане на прагова функция (функция на активация) [38]. Предварителна обработка на данните, преди да бъдат подадени на входа на изкуствената невронна мрежа, често подобряват процеса на обучението [78]. Липсващи стойности в множеството от входни данни оказват влияние при процеса на обучението на изкуствената невронна мрежа [116]. Организация от три слоя е една от най-често използваните в съчетание с точен числен метод за обучение наречен обратно разпространение на грешката [84]. Алгоритъмът с обратно разпространение на грешката дава възможности да бъде подобряван по различни начини [65], което най-много се налага заради склонността да попада в локални минимуми [17]. Освен точни числени методи, при обучението на изкуствени невронни мрежи приложение намират и стохастичните еволюционни методи [97], които често са вдъхновени от природни феномени [24]. Точните числени методи превъзхождат евристичните по ефективност на обучението [82], но евристичните методи дават повече възможности за избягване на локални оптимуми. Също така, градиентните точни числени методи изискват диференцируеми функции [52], което спомага в търсенето на глобални оптимуми [89]. Популационните евристични методи имат много висока степен за паралелна обработка и са изключително подходящи при паралелни изчисления [31] или разпределени изчисления. Добавянето на случаен шум (например, нормално разпределен [134]) може значително да подобри процеса по обучението на многослойни изкуствени невронни мрежи [95]. Случайното изключване на различни възли от скрития слой също може да доведе до подобрено обучение [92]. Освен изключването на неврони, възможно е обучението да започва с по-голяма по размер мрежа, а с времето размера да се намалява (алгоритъм за подрязване) [54]. Противоположно на намаляването е подхода за избор на мрежова топология чрез нарастване [125]. В някои ситуации нарастването на размера на изкуствената невронна мрежа води до експоненциално нарастване на нужните за обучението изчислителни ресурси [123]. Ранно спиране на обучението също може да подобри обобщаващите свойства на изкуствената невронна мрежа [23].

1.3 Евристични оптимизационни алгоритми

Двете основни направления в глобалните оптимизационни алгоритми са точните числени методи и евристичните. При точните числени методи най-често се използва пресмятането на градиент и следването му. Те често са неприложими за нелинейни проблеми [66] или в зашумени входни данни [6]. При евристичните методи за глобална оптимизация (еднокритериална или многокритериална [18]) се залага на някаква евристика (интуитивно очакване) за намиране на решения по-близки до глобалните оптимуми (не се гарантира достигането на глобалния оптимум [20]), като приложението е в множество области [98], в които точните числени методи не биха дали разумен резултат в приемливо време. Възможностите в евристиките са толкова големи, че евристични алгоритми може да се ползват за проектиране на евристични алгоритми [32]. Целта при евристичните методи е да се постигне глобална сходимост [46] в търсене на оптимално решение [7], което понякога се постига и с комбинация от евристики [43], като дори в такава конфигурация евристиките не винаги са достатъчно ефективни [47]. Евристичните методи много често се реализират с помощта на генерирани случайни числа (така наречената генериране-тестване стратегия [126]), генератори на псевдо-случайни числа [57] (чувствителни по отношение на началната инициализация [35]) или хаотични последователности [13]. Докато за определени части от евристиката има теоретична обосновка, то в повечето случаи се използват и параметри, за които няма теоретична обосновка, а се избират стойности по подразбиране (например, вероятността за кръстосване и вероятността за мутация при генетичните алгоритми) [34]. Един от подходите за избор на стойности за параметрите е преди стартирането на изчисленията, а другия подход е адаптивна промяна на стойностите, по време на самото изпълнение. В някои разработки евристични оптимизационни методи се използват за настройка на параметрите при евристична глобална оптимизация. Чрез локално търсене може да се прави фина настройка на параметрите в самия процес по оптимизация [53], както и чрез статистически анализ [40]. Когато евристичните методи се базират на множество от кандидат решения (популация), то те спадат към групата на популационните евристични алгоритми [122].

Популацията може да е организирана в еволюционен процес (например, генетични алгоритми или еволюция на разликите) или в поведение на индивидите (например, колония на мравките и рояк от частици) [36]. Най-честият подход за попълване на популацията в началото е на случаен принцип, но са възможни и решения с предварително подбрани индивиди [19]. Всеки индивид в популацията представлява вектор от пространството на променливите (цели или дробни числа [51]). Този вектор може да бъде с фиксирана или с плаваща дължина [90]. Генетичното разнообразие в популацията може да бъде ключов фактор за ефективен оптимизационен процес [113]. При случаите, в които наличната популация не съдържа достатъчно разнообразни решения, така че някои решения в пространството на търсенето стават недостижими, може да се приложи схема за въвеждане на изцяло новосъздадена популация [119]. Наличието на множество кандидат решения и появата на нови кандидат решения в процеса на оптимизация, често води до забавяне в оценката на решенията, най-вече когато целевата функция се изчислява бавно. При бавни за пресмятане целеви функции [77] е разумно да се търсят някакви начини за ускорена оценка на кандидат решенията [91]. В процеса на евристична оптимизация може да се приложи изследване на пространството и апроксимация с експериментално определено вероятностно разпределение да ускори намирането на оптимално или субоптимално решение [59]. Едно от най-големите предимства на популационните алгоритми е възможността да се изпълняват паралелно или в разпределена среда [117]. Дори, когато изпълнението се извършва с програмни средства за паралелна обработка, но не еднопроцесорна машина, някои популационни алгоритми дават по-добри резултати, спрямо последователната версия на кода [2]. При оптимизационни проблеми с наложени ограничения (особено нелинейни ограничения), допълнителни мерки трябва да се вземат при генерирането на недопустими решения и разрешаването на такива решения да бъдат част от популацията [71].

1.4 Изчисления в разпределена среда

Една значителна част от съвременните алгоритми са изцяло линейни. Това означава, че всяка следваща инструкция зависи от резултатите пресметнати в предходната инструкция. Тази особеност не позволява линейните алгоритми да бъдат изпълнявани на повече от едно аритметично-логическо устройство. Единственият начин за ускоряване на пресмятанията при линейните алгоритми е чрез скъсяване на времето, за което се изпълняват различните инструкции. Друга, доста по-малка част от съвременните алгоритми, позволяват различни групи от инструкции да се пресмятат едновременно. Идеята е по-големите задачи да се раздробяват на по-малки [74], които да се решават едновременно. Един от най-популярните примери е бързото сортиране (QuickSort). Алгоритъмът при бързото сортиране е свързан с първоначално разделяне на масива на две части. След първия пас при възходящо сортиране, лявата част съдържа по-малките елементи, а дясната част по-големите елементи. На свой ред, двете части на масива се възприемат като два разбъркани независими масива. Всяка от двете половини може да се подаде на различно ядро или процесор за пресмятане. От двете половини се получават четири нови подмасива, след това осем подмасива и т.н. От една страна, бързото сортиране има рекурсивна природа, но от друга страна то е идеален кандидат за паралелни пресмятания. Раздробяването на половини може да продължи докато се натоварят всичките налични ядра/процесори. Те заедно работят над обща памет, тъй като няма припокриване между подмасивите. Групата на паралелните алгоритми са подходящи за изпълнение на многопроцесорни системи [112], които споделят обща памет. Стъпка напред в децентрализираното пресмятане, е когато различните процесори са в отделни изчислителни машини, които разполагат със своя собствена памет. Този вид пресмятания се организират в клъстъри [26]. Често клъстърите са съставени от еднотипни изчислителни машини [87], но това не е задължително условие [121]. При свързването на няколко клъстъра и/или супер компютри се формира изчислителен грид (Grid Computing) [103]. Когато изчислителните машини териториално се намират на различни места и контрола върху тях се осъществява от различни хората, изчисленията са в разпределена среда и в почти всички

случаи изчислителните машини са различни [56]. Някои нискобюджетни проекти не могат да си позволят наемането на голям брой изчислителна техника [75] и в такава ситуация се създават дарени изчисления в разпределена среда (Donate/Volunteer Distributed Computing) [114]. При дарените разпределени изчисления, хора ентусиасти (броят участници може да е много голям [14]) преотстъпват изчислителната си техника безвъзмездно [21], така с нея да се извършват изчисления в периодите на занижена употреба. Поради липсата на контрол върху изчислителните машини, надеждността на пресметнатите резултати е винаги под съмнение [27]. Много често такъв вид проекти се реализират под формата на скринсейвъри (Screensavers). Скринсейвърите се появяват заедно с катодно лъчевите тръби при мониторите и задачата им е била да съхраняват покритието на монитора в периоди, в които потребителят не използва компютърната си система. Активацията на скринсейвъра е ясен индикатор, че компютърната система преминава в състояние на намалена употреба. Наличието на скриптови езици [88] в уеб браузърите дава възможност за навлизането и на уеб разпределените изчисления [33]. С масовото навлизане на мобилните устройства през последните петнадесет години, се разкриха нови възможности за дарени изчисления в разпределена среда [41], които се извършват на умни телефони или таблети, докато те са в режим на занижена употреба. Основно предимство на мобилните устройства е, че те работят в непрекъснат режим от 24/7, но пък това е свързано с недостатъка, че изчислителната мощност е значително по-ниска в сравнение със съвременна настолна компютърна система.

Възможността изкуствените невронни мрежи да се обучават с еволюционни алгоритми [28] (обичайно много времеемък процес [44]), понякога комбинирани с обратно разпространение на грешката [133], позволява ефективната реализация на такова обучение в разпределена среда [81]. Най-често глобалната популация се разпределя на множество машини [83], като локални популации и решения мигрират между машините [99]. Другият подход е различни задачи да се възлагат на отделните изчислителни машини [3]. Стратегията за възлагане на отделни изчислителни задачи [1] в разпределената среда, от своя страна, също може да е осъществена с еволюционни евристични алгоритми [93]. Освен оптимизация на теглата в мрежата е възможна

и оптимизация на структурата ѝ [29]. Характерно за разпределените изчисления е високата консумация на електрическа енергия [39] и увеличаване на въглеродния отпечатък [68]. Обучението на изкуствени невронни мрежи с еволюционни алгоритми може да се реализира и в разпределена среда със специализиран хардуер [37]. При евристичните алгоритми (особено еволюционните и/или популационните) най-често се залага на многократно изчисляване на целевата функция. Общото правило е, че евристиките са ефективни за задачи, в които целевата функция се пресмята бързо. В реалната практика обаче, често целевата функция е най-бавната част от оптимизационния процес. Точно при такива задачи изчисленията в разпределена среда са най-силният инструмент [72]. Разпределянето на пресмятанията не води до 100% ускорение, тъй като има допълнителна работа при синхронизацията на резултатите и комуникацията в мрежата [41]. Повечето съвременни програмни езици са обектно-ориентирани. В това число влиза и JavaScript, макар и там да няма класове, но все пак се работи с обекти. При реализацията на разпределени изчисления една част от програмните езици разчитат на бинарна сериализация на обектите и разпращането им по мрежата (например, RMI, DCOM или CORBA) [30]. Бинарната сериализация има редица недостатъци, така че в уеб базираните приложения и в RESTful приложенията се залага на описателна серизализация (например, JSON или XML). При използването на JSON, релациите съхранени в релационната база данни се оформят като JSON комуникационен пакет и от страната на клиента, ако това е JavaScript, директно биват използвани във формата на JavaScript обекти. Когато клиентското приложение не е JavaScript, се преминава през Parser, трансформиращ структурираната текстова JSON информация в обект за съответния програмен език.

1.5 Дискусия и изводи

Изработването на прогнози е от голяма важност за съвременните общества, като започнем от прогноза за метеорологичната обстановка и стигнем до прогнози за промяната на цените за стоки, акции и валути. В случая с прогнозирането на цени, данните успешно се представят във формата на времеви ред. Съвсем логично, вся-

ка следваща стойност да има някаква зависимост от предходните стойности. През последните десетилетия са разработени много начини за прогнозиране на финансови времеви редове, но като един от най-обещаващите се открояват изкуствените невронни мрежи. Характерно за изкуствените невронни мрежи е, че те са много ефективен инструмент, след като веднъж са обучени. Процесът на обучение, от своя страна, често отнема твърде дълго време и се нуждае от голямо количество изчислителни ресурси. Един от най-използваните начини за обучение на изкуствени невронни мрежи е алгоритъмът с обратно разпространение на грешката. Този алгоритъм спада към групата на точните числени, градиентни алгоритми. Негови слабости са невъзможността да бъде ефективно реализиран в паралелни изчисления и склонността му да изпада в локални оптимуми, без да има ефективни способи за избягването им. Алгоритъмът за обратно разпространение на грешката много добре се допълва с евристичните, еволюционни алгоритми за глобална оптимизация. Характерното за този вид евристики е, че те се поддават на изключително висока степен за паралелна обработка. Някои от тези евристики са специално създадени за избягване на локалните оптимуми. Широките възможностите за паралелна обработка при еволюционните и популационните евристики позволяват реализацията им на хетерогенни системи за разпределени изчисления. С цел за по-висока финансова ефективност, този вид разпределени изчисления могат да се изпълняват на принципите за дарената изчислителна мощност. Наличието на значително повече мобилни устройства (умни телефони и таблети), спрямо настолните компютърни системи, води до мотивация пресмятанията да се реализират под формата на мобилни разпределени изчисления.

Всичко изброено до тук, дава основанието да се търси реализация на система за мобилни разпределени изчисления, която обучава изкуствени невронни мрежи, с хибриден алгоритъм (обратно разпространение на грешката и популационна глобална оптимизация), за прогнозиране на финансови времеви редове.

Обобщената информацията от направения обзор е представена в Таблица 1.1.

	Точно числени методи за обучение	Евристични методи за обучение
Вид	Обратно разпространение на грешката; Еластично обратно разпространение на грешката; Манхатън обучение; Метод на конюгиран градиент.	Еволюция на разликите; Генетичен алгоритъм; Еволюционна стратегия.
Приложение	Работят добре при решаване на задачи с по-ниска сложност.	Възможност за паралелна обработка или разпределени изчисления; Прилагат се там, където точните числени методи не дават резултата в приемливо време.
Особености	Изискват диференцируеми функции	Могат да се използват при прекъснати функции, стъпаловидни функции и функции без дефинирана производна в определени области.
Недостатъци	Склонност да попада в локален оптимум; Трудно се организират в паралелни или разпределени изчисления.	Обучението може да отнеме много време. Целевата функция се пресмята бавно, което е основна причина за забавянето.
Предимства	Значително по-бърза сходимост на процеса по обучение, стига да не се попадне в локален оптимум.	Спомагат за избягването на локални оптимуми.

Таблица 1.1: Обобщение на точните и евристичните методи за обучение

Глава 2

Алгоритми при прогнозиране и обучение на ИНМ

2.1 Ограничения в пространството на променливите

Теглата в класическия многослоен перцептрон са реални числа, като не се поставят ограничения за техните стойности. Теглата могат да варират от големи отрицателни числа до големи положителни числа. Стойностите на тези тегла най-често се представят в матрица, стълбовете и редовете, на която матрица показват кой неврон с кой неврон е свързан, чрез конкретно тегло в ред и стълб. При популационните оптимизационни алгоритми се работи с вектори (индивидите в популацията), така че матрицата бива представена като вектор. Задачата за търсене на оптимални стойности на теглата не поставя ограничения за самите стойности, но когато оптимизацията се извършва при поставени ограничения, и най-вече нелинейни ограничения, то популационните оптимизационни алгоритми лесно изпадат в локални оптимуми или в невъзможност да генерират допустими решения. За да се избегнат тези две трудности, може да се въведе подход за допълнителна популация (пул от кандидат решения), където да се съхраняват всички допустими решения в процеса на еволюиране. Класическият подход за отразяване на факта, че едно решение (индивид) в основната популация е недопустимо, става чрез въвеждане на санкция (penalty).

Санкцията е някаква форма на числено влошаване на жизнената стойност, присвоена на индивида. Наличието на санкции в популацията възпрепятства присъствието на недопустими решения и по този начин тласка алгоритъма към изпадане в локални оптимуми. Получава се така, че допустимите решения доминират недопустимите решения, а за ефективно изследване на пространството е от съществено значение алгоритъмът да може да преминава през недопустими решения, така че да достигне до следващи (неизвестни до сега) допустими решения. Този пул от допустими решения може да се използва на определени интервали от време за внасяне на генетично разнообразие в основната популация. За да се избегне израждането на основната популация, а и на пула с допустими решения, то на всеки индивид, освен жизнена стойност, може да се постави характеристика за стареене (aging). Коефициентът за стареене е своеобразна вероятност за изборът на индивида в процеса по селекция при създаването на следващи поколения. Колкото по-дълго е стоял индивидът в основната популация и в пула с допустими решения, толкова по-малка е вероятността този индивид да бъде избран за бъдещо репродуциране. Пулът с допустими решения има и вторична роля, която е свързана с предлагане на решение след приключване на оптимизационния процес. Имайки ролята на операция за елит, най-доброто решение в този пул се предлага като окончателен резултат от оптимизацията.

2.2 Оператор за селекция с локално търсене, пълно изчерпване и рекурсивно спускане

Генетичните алгоритми са глобална оптимизационна мета-евристика, вдъхновена от идеите в естествената еволюция. Процесът на оптимизация е организиран в три общи операции - селекция, кръстосване и мутация. Кръстосването и мутацията са отговорни за предлагането на нови решения в популацията, а селекцията е отговорна за по -добър избор на родители. През последните пет десетилетия в литературата са предложени много оператори за селекция - пропорционална селекция, турнирна слекция, ранг базиран селекция, болцманова селекция, слекция с нелинейно кла-

сиране, конкурентна селекция и други. Тук се предлага нов оператор за селекция, основан на създаването на рекурсивни поколения. На всяко ниво на рекурсия всички индивиди в популацията се чифтосват помежду си (груба сила). Алгоритъмът на грубата сила се комбинира с локално търсене, като груба сила се прилага, докато има излъчване на по-добър индивид в поколението

Листинг 2.1: Равномерно кръстосване

```
static double[] crossover(double first[], double second[]) {
        double[] child = new double[INPUT_SIZE];

        for (int i = 0; i < INPUT_SIZE; i++) {
                if (PRNG.nextBoolean()) {
                        child[i] = first[i];
                } else {
                        child[i] = second[i];
                }
        }

        return child;
}
```

Листинг 2.2: Мутация със случайно число

```
static void mutate(double child[]) {
        for (int i = 0; i < INPUT_SIZE; i++) {
                if (PRNG.nextDouble() >= MUTATION_RATE) {
                        continue;
                }

                child[i] += PRNG.nextDouble() - 0.5D;
        }
}
```

Популацията в генетичния алгоритъм е организирана като йерархична структура. На най-ниското ниво на структурата популацията се състои от произволно генерирани индивиди (Листинг 2.3). Във всяка популация всеки индивид се чифтосва с всички други (кръстосване - Листинг 2.1 и мутация - Листинг 2.2). Само най-жизненият индивид се изпраща към по-горното ниво на рекурсивната структура (Листинг 2.3). По този начин всяка по-горна популация се установява от най-добрите индивиди, изпратени от по-долните нива.

Листинг 2.3: Йерархична селекция

```
private static double[] solution(int depth, int size, Selection selection,
                Function function) {
        double[] result = null;

        if (depth > 0) {
                double population[][] = new double[size][];
                for (int j = 0; j < size; j++) {
                        population[j] = solution(depth - 1, size, selection, function);
                }

                result = selection.best(population, function);
        } else if (depth == 0) {
                /* First generation is selected randomly. */
                result = new double[INPUT_SIZE];
                for (int i = 0; i < INPUT_SIZE; i++) {
                        result[i] = function.minimum() + PRNG.nextDouble()
                                        * (function.maximum() - function.minimum());
                }
        }

        return result;
}
```

Като тестове се взети общо известни функции за проверка на евристични оптимизационни алгоритми (Michalewicz, Ackley, Schwefel Rastrigin, Griewank). Всички експерименти се извършват в 10K пространствено реално число [107]. Подобрението, постигнато с добавянето на локално търсене, спрямо варианта използващ единствено пълно изчерпване е илюстрирано в Таблица 2.1.

	Michalewicz	Ackley	Schwefel	Rastrigin	Griewank
Local Search	1062492546	531027435	533232986	507650704	592370933
Brute-Force	403141211	166733879	175069174	159862955	218428047

Таблица 2.1: Време за изчисление [ms] при ниво на рекурсия 7 и размер на популацията 11

Експериментите са проведени при ниво на рекурсия 7 и размер на популацията 11. В Таблица 2.2 са визуализирани броят изчисления на целевата функция, спрямо вида алгоритъм, за всяка от тестовите функции.

Броят изчисления на целевата функция не може да бъде предварително определен във варианта, когато се прилага локално търсене [109]. Достигнатите субоптимални стойности за всяка от функциите, при ниво на рекурсия 7 и размер на популацията 11, са представени в Таблица 2.3. Таблично са представени граничния случай с ниво

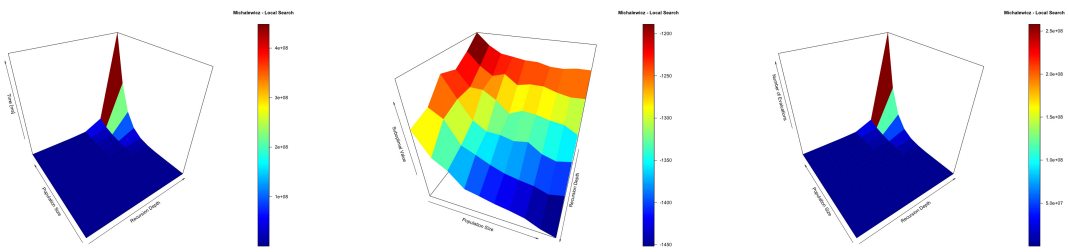
на рекурсия 7 и размер на популацията 11. Графично се представят резултатите от експериментите проведени за нива на рекурсия от 2 до 7 и размери на популацията от 2 до 11 (Фиг. 2.1а-2.9в).

	Michalewicz	Ackley	Schwefel	Rastrigin	Griewank
Local Search	641125639	641024362	640914978	641092606	640830762
Brute-Force	235794757	235794757	235794757	235794757	235794757

Таблица 2.2: Брой изчисления на целевата функция при ниво на рекурсия 7 и размер на популацията 11

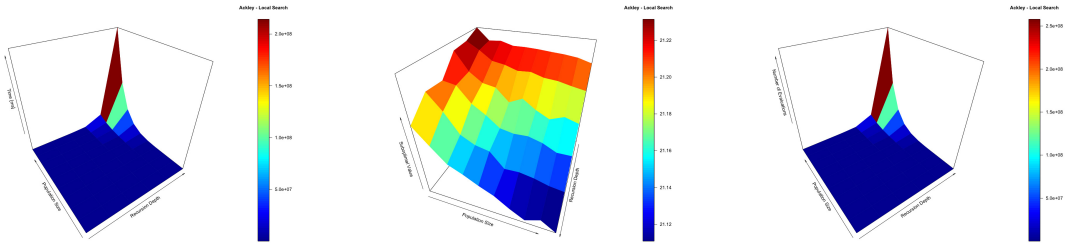
	Michalewicz	Ackley	Schwefel
Local Search	-1484.7137949531716	21.09334816052046	3877924.0971615044
Brute-Force	-1439.2296970724608	21.114702255301292	3919318.729777085
	Rastrigin	Griewank	
Local Search	170204.87849875208	259918.15469527297	
Brute-Force	171780.33307271387	262621.61053178157	

Таблица 2.3: Изчислени субоптимални стойности при ниво на рекурсия 7 и размер на популацията 11



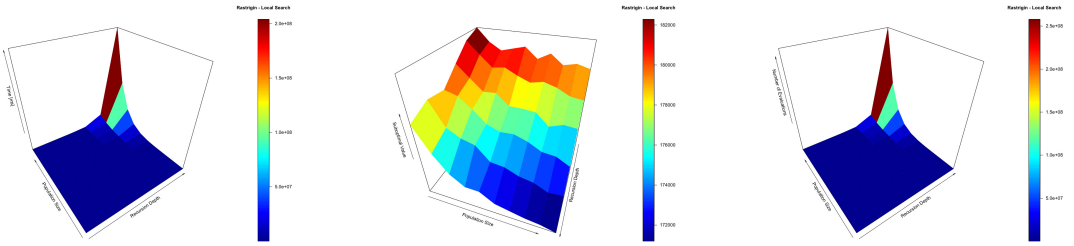
(а) Изчислително време [ms] (б) Субоптимални стойности (в) Брой смятания на функцията

Фигура 2.1: Функцията Michalewicz с локално търсене



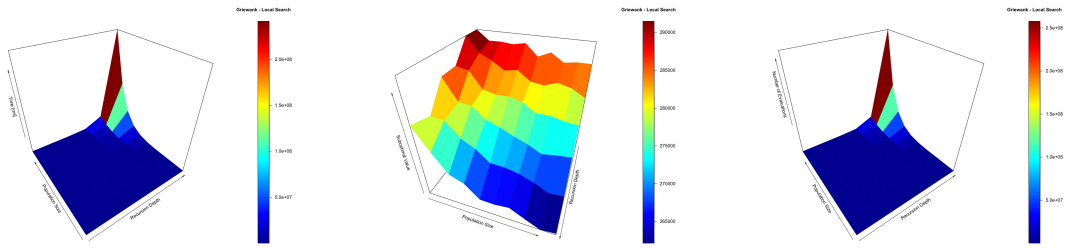
(а) Изчислително време [ms] (б) Субоптимални стойности (в) Брой смятания на функцията

Фигура 2.2: Функцията Ackley с локално търсене



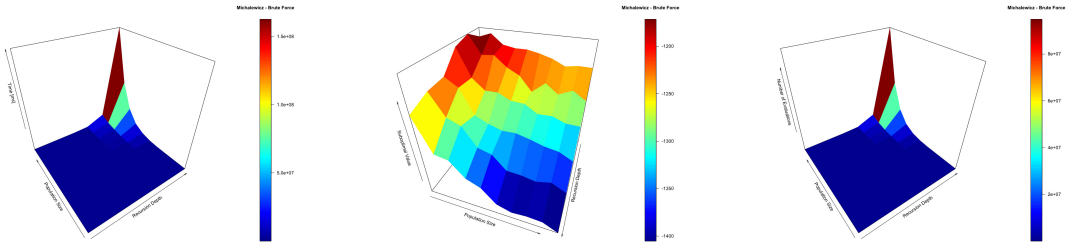
(а) Изчислително време [ms] (б) Субоптимални стойности (в) Брой смятания на функцията

Фигура 2.3: Функцията Rastrigin с локално търсене



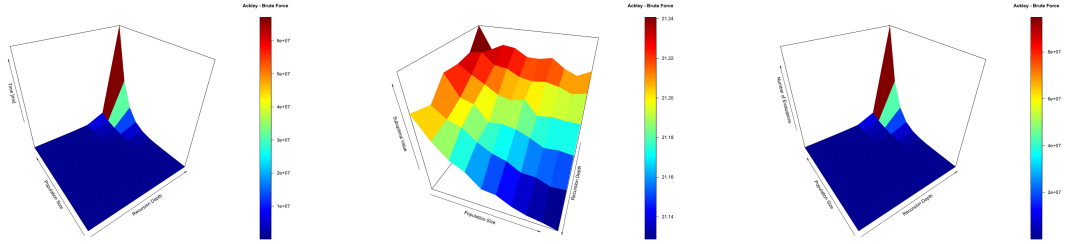
(а) Изчислително време [ms] (б) Субоптимални стойности (в) Брой смятания на функцията

Фигура 2.4: Функцията Griewank с локално търсене



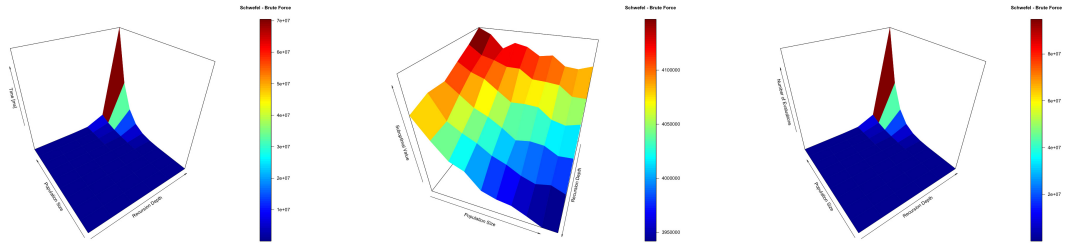
(а) Изчислително време [ms] (б) Субоптимални стойности (в) Брой смятания на функцията

Фигура 2.5: Функцията Michalewicz с пълно изчерпване



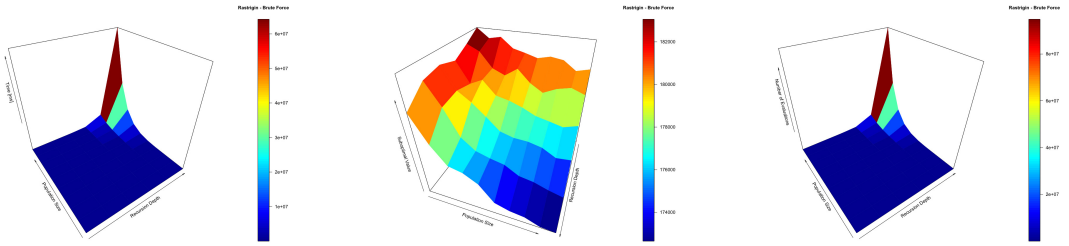
(а) Изчислително време [ms] (б) Субоптимални стойности (в) Брой смятания на функцията

Фигура 2.6: Функцията Ackley с пълно изчерпване



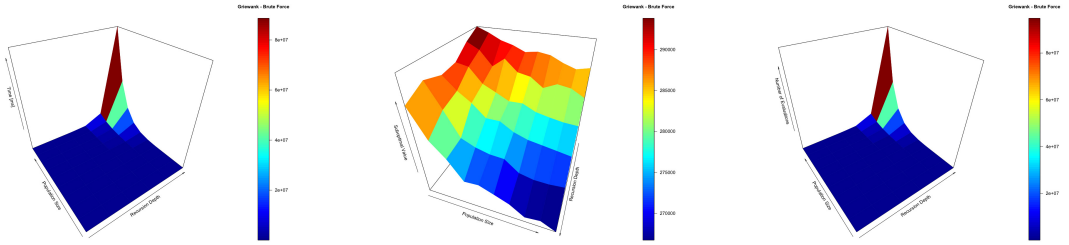
(а) Изчислително време [ms] (б) Субоптимални стойности (в) Брой смятания на функцията

Фигура 2.7: Функцията Schwefel с пълно изчерпване



(а) Изчислително време [ms] (б) Субоптимални стойности (в) Брой смятания на функцията

Фигура 2.8: Функцията Rastrigin с пълно изчерпване



(а) Изчислително време [ms] (б) Субоптимални стойности (в) Брой смятания на функцията

Фигура 2.9: Функцията Griewank с пълно изчерпване

От извършените експерименти, ясно се вижда, че добавката на локално търсене води до по-добри резултати, но това е за сметка на по-дългото време за пресмятане.

2.3 Инкрементална апроксимация със синусоиди и тренд

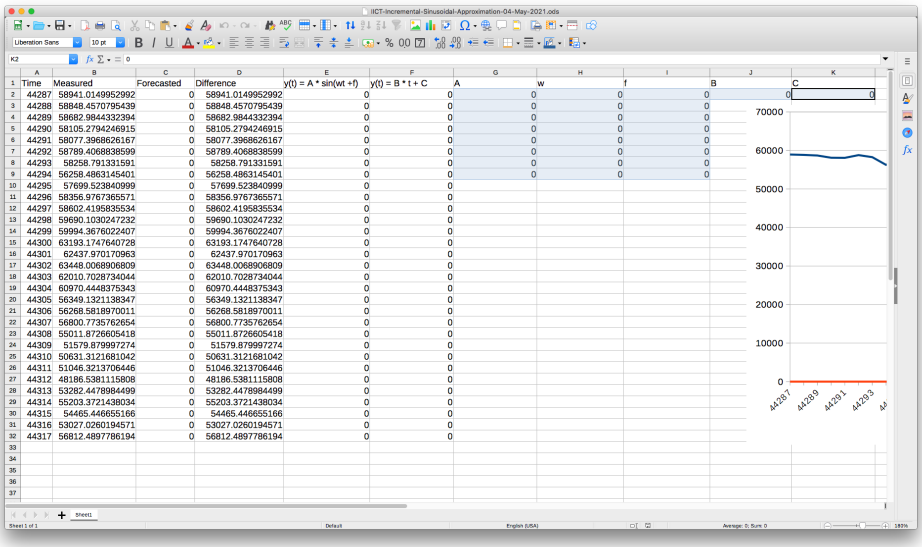
Тъй като времевите редове представляват измервания, извършени в строга последователност по отношение на оста на времето, получените точки на измерване могат да се подложат на анализ с инструментите за обработка на сигнали. През измерените точки или в близост до тях, е възможно да се построят криви (пример е полиномът на Лагранж), описани с математически формули. При достигане на достатъчно близост до точките, надеждата е построените криви да имат поне частични свойства за формиране на прогноза, извън интервала на известните измервания, засягащ бъде-

щите моменти във времето. В общия случай става въпрос за търсене на подходящи коефициенти в математически формули. Търсенето на подходящите коефициенти е оптимизационна задача със значителна сложност, поради най-често използваните нелинейни компоненти във формулите. Колкото повече коефициенти са обект на оптимизация, толкова по-сложна става оптимизационната задача. По аналогия с трансформацията на Фурие, възможно е през времевия ред да се построи крива, изградена от множество синус функции и уравнение на права (обозначава тренда). Теоретично е доказано, че през краен брой точки в двумерното пространство могат да се прекарат безкраен брой криви. Запазването на свойството за обобщение (прогноза) силно зависи от възможността броя синус функции да бъде по възможност по-малък.

При финансовите времеви редове е много характерно да има множество флуктуации (движение нагоре или надолу). Тази особеност прави финансовите времеви редове идеални кандидати за апроксимация, чрез ред от синус функции. Във финансовите времеви редове е изключителна рядкост да отсъства тренд. Най-често във финансовите времеви редове се откроява ясно различим тренд на нарастване или тренд на спадане. Трендът ефективно се описва с уравнение на права в двуизмерното пространство, като чрез методи като линейната регресия лесно може да се установят двата параметъра – наклон и срез.

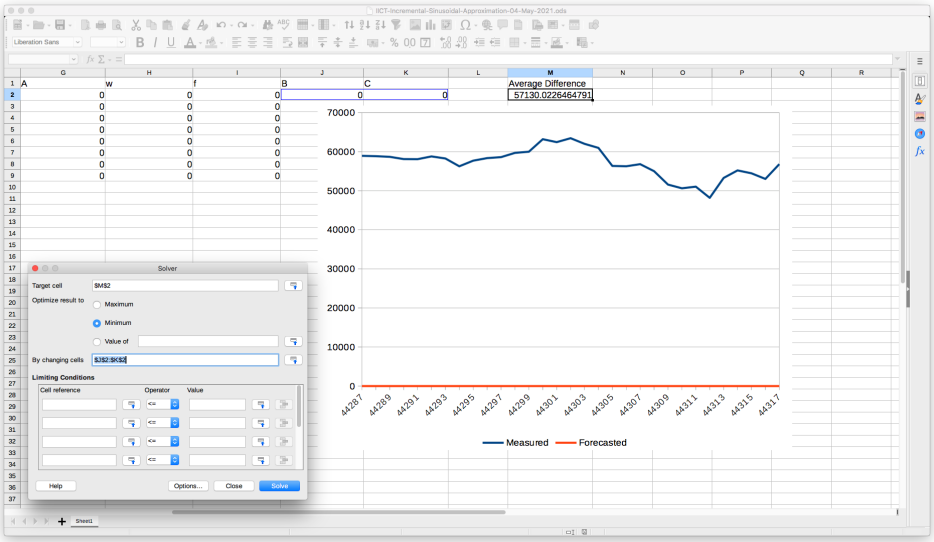
$$y(t) = B.t + C + A_1.sin(\omega_1.t + \varphi_1) + A_2.sin(\omega_2.t + \varphi_2) + ... \tag{2.1}$$

Уравнението за апроксимация чрез тренд и синус функции (2.1) съдържа множество коефициенти. Коефициентът B задава наклона на правата. Коефициентът C задава среза в уравнението на правата. Коефициентите A_n задават амплитудата на синус функциите. Коефициентът ω_n задава ъгловата скорост. Коефициентът φ_n задава фазовото отместване. Точният брой синус функции се определя от процедурата по инкрементално оптимизиране на коефициентите. Стремежът е този брой да бъде възможно най-малък, но и изчислените прогнозни стойности да бъдат възможно най-близки до реалните бъдещи стойности.



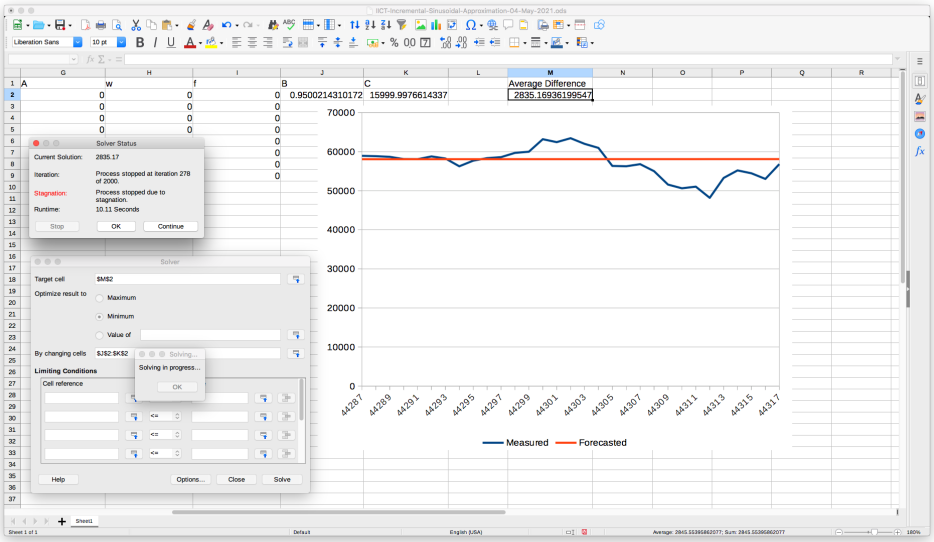
Фигура 2.10: Входни данни за цената на биткойн

Инкременталният процес по оптимизация започва с оптимизацията само на коефициентите в линейния компонент (B и C). Постигнатите оптимални или субоптимални стойности за тренда служат като основа за разширяване на пространството на променливите, като се добавят три коефициента за първата синус функция (A_1, ω_1 и φ_1). Тъй като оптимизаторът използван в LibreOffice Calc е стохастичен, базиран на еволюция на разликите и рояк от частици, процесът по оптимизация се прекратява, когато постигнатото най-приближено решение не се подобрява в предварително зададен интервал от време. След изчерпването на възможностите за подобряване на решенията с една синус функция се добавя втора синус функция, която разширява пространството на променливите с още три (A_2, ω_2 и φ_2). Добавянето на синус функции се съблюдава ръчно, тъй като всеки времеви ред се характеризира със своя собствена уникална форма. Важно е да не се добавят твърде много синус функции, защото това би довело до пренапасване на модела (overfitting) и до загуба на възможността за обобщаване (прогнозиране).



Фигура 2.11: Настройки за оптимизация

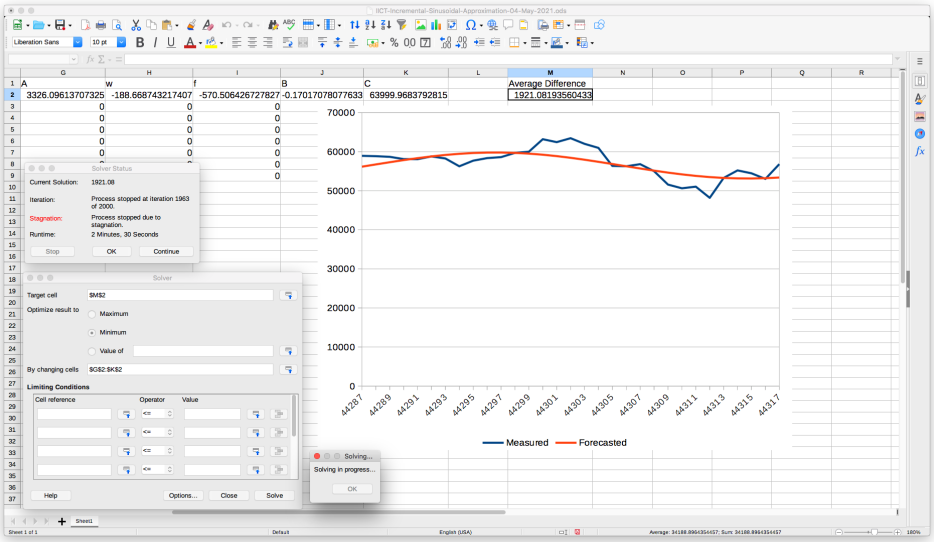
За демонстрация на описания начин за пресмятане се използват данни за стойността на дигиталната криптовалута биткойн в щатски долари, на дневна база, за период от един месец (Фиг. 2.10). Стохастичните оптимизационни алгоритми могат да започнат процеса за оптимизация от различни точки в пространството на променливите. Колкото по-благоприятни са началните точки, толкова по-големи са шансовете за достигане на глобално оптимално решение. В предложения пример оптимизацията започва от вектор, съдържащ нули във всичките си компоненти. В колона A е зададено времето, под формата на числена стойност, според интерпретацията на LibreOffice Calc. В колона B е стойността на отваряне (началото на интервала) за дигиталната валута в съответния ден. В колона C е поместена пресметнатата прогнозна стойност. В колона D е квадратния корен от втората степен на разликата между реалната стойност и прогнозираната стойност. Сумата от всички разлики е пресметната в клетка $M2$ (2.11).



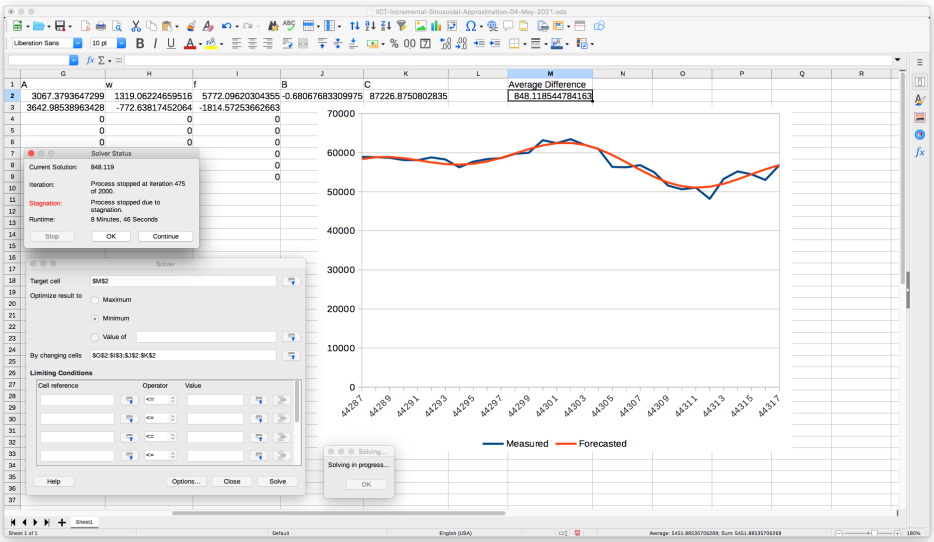
Фигура 2.12: Коефициенти за уравнение на права

След първия цикъл на инкрементална оптимизация, оптимизаторът на LibreOffice Calc определя коефициенти за уравнението на правата, която да преминава възможно най-близо до измерените точки (Фиг. 2.12). Поради стохастичната природа на двата оптимизатора (еволюция на разликите и рояк от частици), най-често решенията са субоптимални, като доближават глобалния оптимум, но не го достигат гарантирано. Също така, при различни стартирания на цикъла за оптимизация, се получават различни субоптимални решения.

След като коефициентите за определяне на тренда бъдат подбрани, следва включване на първа синус функция във втория цикъл от инкременталната оптимизация (Фиг. 2.13). Оптимизацията се извършва едновременно за вече определените коефициенти и за новодобавените три коефициента на първата синус функция. Този подход за надграждане дава възможност за допълнителна фина нагласа на коефициентите за тренда.



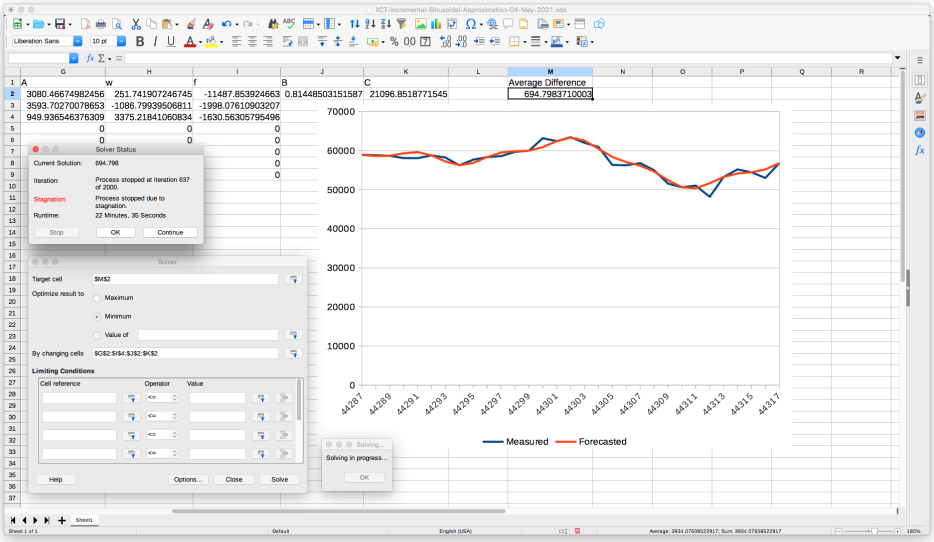
Фигура 2.13: Включване на първа синус функция



Фигура 2.14: Включване на втора синус функция

След като оптимизаторите на LibreOffice Calc достигнат ситуация, в която значително подобрение в целевата стойност (клетка $M2$) не се постига (състояние обозначено в модула Solver като stagnation), следва да се премине към следващ цикъл от инкременталната оптимизация, чрез добавяне на втора синус функция (Фиг. 2.14). За конкретните данни от примерния времеви ред, ясно се вижда, че тренд и две синус функции водят до добро напасване и имат нужните свойства за изчисляване

на прогнозни стойности. При времеви редове с повече на брой измервания и повече на брой флуктуации, броят на синус функциите би бил по-голям.



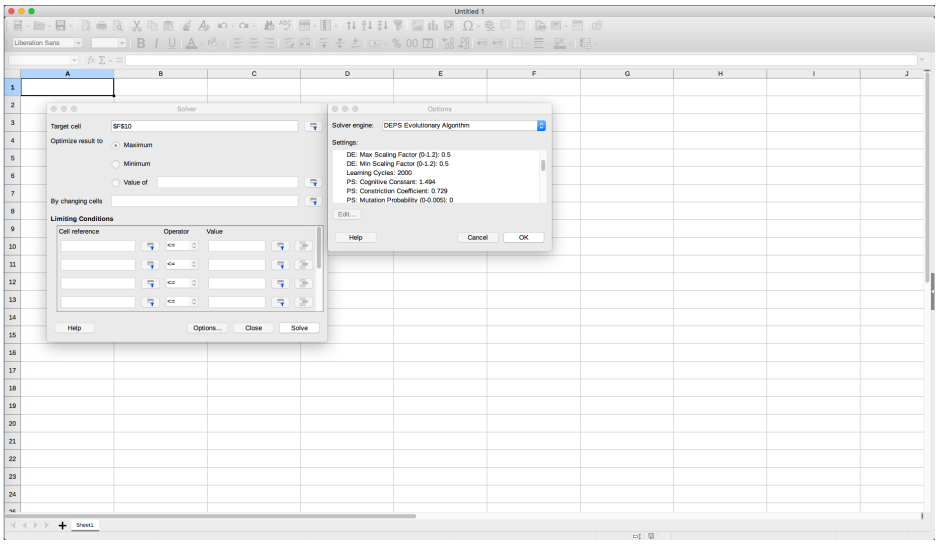
Фигура 2.15: Включване на трета синус функция

За да се подбере точния момент, в който трябва да се преустанови добавянето на синус функции, се съблюдават разликите в целевата клетка на оптимизацията (клетка $M2$). При добавянето на трета синус функция (Фиг. 2.15) се забелязва, че целевата клетка не променя драстично стойността си. Изборът до коя на брой синус функция да се прекрати инкременталната оптимизация остава изцяло субективен и от компетентността на лицето, вземащо решения на база на генерираните прогнози.

2.4 Бърз прототип на LibreOffice Calc с еволюция на разликите и оптимизация с рояк от частици

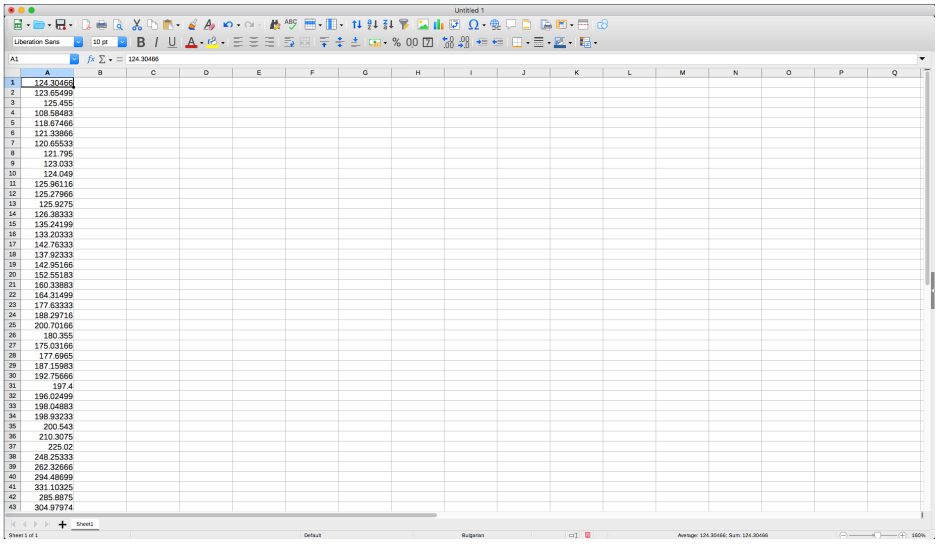
Процесът по търсенето на оптимални тегла в изкуствена невронна мрежа от тип трислоен перцептрон много нагледно може да бъде демонстриран с бърз прототип в софтуерния пакет LibreOffice Calc. За осъществяване на прототипирането, моделът на изкуствената невронна мрежа бива разгърнат в двузимерната равнина от клетки на електронната таблица. Търсенето на оптимални стойности за теглата в мрежата

се постига чрез вградения в LibreOffice Calc модул за оптимизация, наречен Solver (Фиг. 2.16).



Фигура 2.16: Модул за оптимизация в LibreOffice Calc

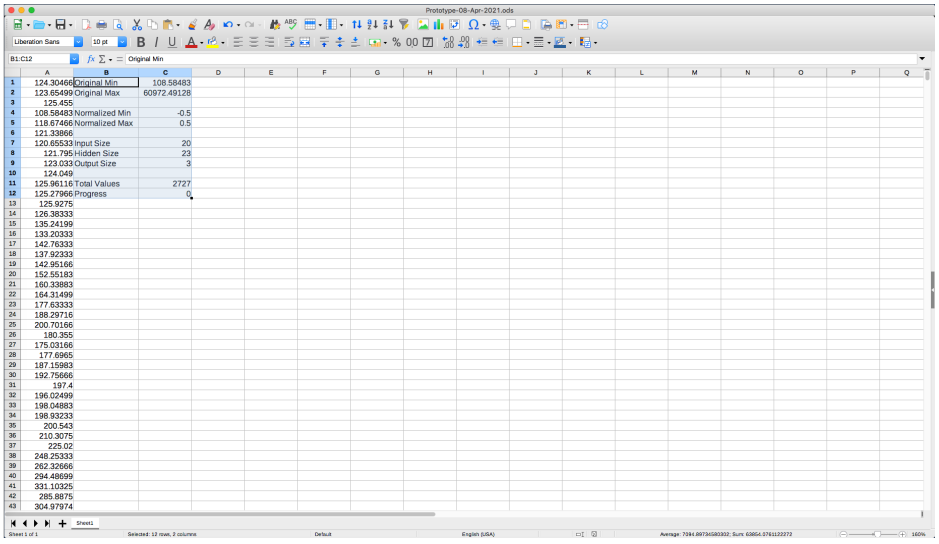
За нелинейна оптимизация, модулът прилага алгоритмите за еволюция на различките и оптимизация с рояк от частици. Двата алгоритъма се прилагат в хибридна комбинация, като с предварително дефинирана вероятност, е определено колко често ще бъде активиран всеки от тях. Модулът се настройва за клетка, чиято оптимална стойност ще бъде търсена (максимум, минимум или конкретно число). Също така, задава се и регионът от клетки, които подлежат на промяна в процеса по оптимизация. Като клетка, в която ще се търси минимум при бързото протитипиране се избира общата средно-квадратична грешка, допусната от изкуствената невронна мрежа. Регионът от клетки за оптимизация съдържа теглата, използвани в изкуствената невронна мрежа.



	A
1	124.30466
2	123.65499
3	125.455
4	108.58483
5	118.67466
6	121.33866
7	120.65533
8	121.795
9	123.033
10	124.049
11	125.96116
12	125.27966
13	125.9275
14	126.38333
15	135.24199
16	133.20333
17	142.76333
18	137.92333
19	142.95166
20	152.55183
21	160.33883
22	164.31499
23	177.63333
24	188.29716
25	200.70166
26	180.355
27	175.03166
28	177.6965
29	187.15983
30	192.75666
31	197.4
32	196.02499
33	198.04883
34	198.93233
35	200.543
36	210.3075
37	225.02
38	248.25333
39	262.32666
40	294.48699
41	331.10325
42	285.8875
43	304.97974

Фигура 2.17: Стойности на биткойн виртуалната криптовалута

Като множество данни се използват стойностите на биткойн виртуалната криптовалута (Фиг. 2.17) на дневна база, за няколко години назад. Моделът за прогнозиране се основава на нелинейна авторегресия. Това означава, че на входа на мрежата се подават мащабирани минали стойности от времевия ред, а на изхода на мрежата се очакват мащабирани прогнозни стойности.



	A	B	C
1	124.30466	Original Min	108.58483
2	123.65499	Original Max	60972.49128
3	125.455		
4	108.58483	Normalized Min	-0.5
5	118.67466	Normalized Max	0.5
6	121.33866		
7	120.65533	Input Size	20
8	121.795	Hidden Size	23
9	123.033	Output Size	3
10	124.049		
11	125.96116	Total Values	2727
12	125.27966	Progress	0
13	125.9275		
14	126.38333		
15	135.24199		
16	133.20333		
17	142.76333		
18	137.92333		
19	142.95166		
20	152.55183		
21	160.33883		
22	164.31499		
23	177.63333		
24	188.29716		
25	200.70166		
26	180.355		
27	175.03166		
28	177.6965		
29	187.15983		
30	192.75666		
31	197.4		
32	196.02499		
33	198.04883		
34	198.93233		
35	200.543		
36	210.3075		
37	225.02		
38	248.25333		
39	262.32666		
40	294.48699		
41	331.10325		
42	285.8875		
43	304.97974		

Фигура 2.18: Параметри на трислойната изкуствена невронна мрежа

След подбора на времеви ред следва избор на параметрите, с които ще бъде напревен моделът на трислойната изкуствена невронна мрежа (Фиг. 2.18). За це-

лите на линейното мащабиране първо се намират най-малката и най-голямата стойност в оригиналния времеви ред, чрез формули в LibreOffice Calc: $= MIN(A : A)$ и $= MAX(A : A)$. Тъй като приложената прагова функция е хиперболичен тангенс, диапазона за мащабирания времеви ред е избран от -0.5 до +0.5. Умишлено се избягва мащабиране до асимптотичните стойности от -1.0 до +1.0, тъй като такова мащабиране много би увеличило стойностите на междинните пресмятания, а и не би дало възможност да се прогнозират по-малки или по-големи стойности от вече известните в оригиналния времеви ред. Топологията на мрежата се избира експериментално, като изходния слой има размер, според това колко стойности в бъдещето е желателно да се предсказват. Размерът на входния слой се определя експериментално. За размера на скрития слой има различни емпирични правила. Популярни варианти са той да е сумата от входния и изходния слой или половината от тяхната сумата. Съществуват адаптивни алгоритми, които чрез проби и грешки да определят топологията на мрежата, но в това бързо прототипиране тези алгоритми не се прилагат.

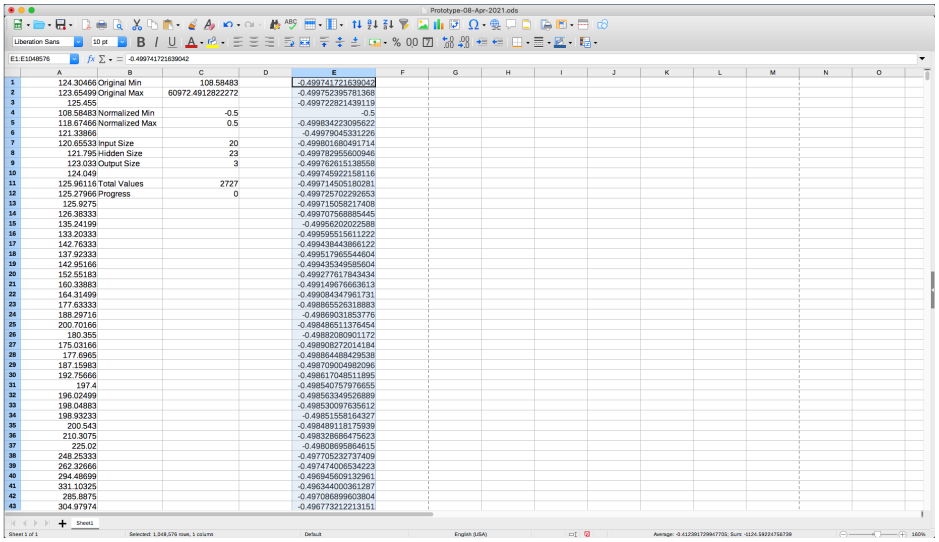
Общият брой стойности в оригиналния времеви ред се определят с формула в LibreOffice Calc: $= COUNT(A : A)$. Тъй като процеса по „разгръщането“ на модела е относително бавен, то се добавя клетка, в която да се проследява напредъка от Python скрипта в „разгръщането“. LibreOffice Calc позволява изпълнението на макроси, като се поддържат няколко програмни езика. Програмният език Python е изключително популярен в сферата на машинното самообучение и дава изключително големи възможности за частична автоматизация в програмите на LibreOffice.

След мащабирането на оригиналния времеви ред се формират тренировъчните примери, като за вход се вземат стойности преди условния момент t_0, а за очакван изход стойности след условния момент t_0 (условно разделяне на минало и бъдеще).

Листинг 2.4: Мащабиране на оригиналния времеви ред

```
''' Scale input. '''
for t in range(1, total_values + 1):
    sheet.getCellRangeByName("E" + str(t)).setValue(sheet.getCellRangeByName("$C$4").getValue() +
    (sheet.getCellRangeByName("$C$5").getValue() - sheet.getCellRangeByName("$C$4").getValue()) *
        ((sheet.getCellRangeByName("A" + str(t)).getValue() -
    sheet.getCellRangeByName("$C$1").getValue()) / (sheet.getCellRangeByName("$C$2").getValue() -
        sheet.getCellRangeByName("$C$1").getValue())))
```

В листинг 2.4 се демонстрира линейното мащабиране, като за извършване на изчисленията се използват установените минимални и максимални стойности (Фиг. 2.19).



	A	B	C	D	E
1	124.30466	Original Min	108.58483		-0.499741721639042
2	123.65499	Original Max	60972.4912822272		-0.499752395781368
3	125.455				-0.499722821439119
4	108.58483	Normalized Min	-0.5		-0.5
5	118.67466	Normalized Max	0.5		-0.499834223095622
6	121.33866				-0.49979045331226
7	120.65533	Input Size	20		-0.499801680491714
8	121.795	Hidden Size	23		-0.499782955600946
9	123.033	Output Size	3		-0.499762615138558
10	124.049				-0.499745922158116
11	125.96116	Total Values	2727		-0.499714505180281
12	125.27966	Progress	0		-0.499725702292653
13	125.9275				-0.499715058217408
14	126.38333				-0.499707568885445
15	135.24199				-0.49956202022588
16	133.20333				-0.499595515611222
17	142.76333				-0.499438443866122
18	137.92333				-0.499517965544604
19	142.95166				-0.499435349585604
20	152.55183				-0.499277617843434
21	160.33883				-0.499149676663613
22	164.31499				-0.499084347961731
23	177.63333				-0.498865526318883
24	188.29716				-0.49869031853776
25	200.70166				-0.498486511376454
26	180.355				-0.49882080901172
27	175.03166				-0.498908272014184
28	177.6965				-0.498864488429538
29	187.15983				-0.498709004982096
30	192.75666				-0.498617048511895
31	197.4				-0.498540757976655
32	196.02499				-0.498563349526889
33	198.04883				-0.498530097635612
34	198.93233				-0.49851558164327
35	200.543				-0.498489118175939
36	210.3075				-0.498328686475623
37	225.02				-0.49808695864615
38	248.25333				-0.497705232737409
39	262.32666				-0.497474006534223
40	294.48699				-0.496945609132961
41	331.10325				-0.496344000361287
42	285.8875				-0.497098689903804
43	304.97974				-0.496773212213151

Фигура 2.19: Резултат от мащабирането на времевия ред

На всеки слой в изкуствената невронна мрежа се добавя по един допълнителен неврон (Листинг 2.5), постоянно емитиращ единична стойност (отместване или от английски език bias).

Листинг 2.5: Неврони емитиращи постоянно единичен сигнал

```
''' Setup biases. '''
sheet.getCellRangeByName("G" + str(x)).setValue(1)
sheet.getCellRangeByName("G" + str(x)).CellBackColor = (255 << 16 | 255 << 8 | 0)
sheet.getCellRangeByName("H" + str(x)).setValue(1)
sheet.getCellRangeByName("H" + str(x)).CellBackColor = (255 << 16 | 255 << 8 | 0)
sheet.getCellRangeByName("I" + str(x)).setValue(1)
sheet.getCellRangeByName("I" + str(x)).CellBackColor = (255 << 16 | 255 << 8 | 0)
sheet.getCellRangeByName("J" + str(x)).setValue(1)
sheet.getCellRangeByName("J" + str(x)).CellBackColor = (255 << 16 | 255 << 8 | 0)
```

Мащабираният времеви ред бива „разбит“ условно на „минали“ стойности и „бъдещи“ стойности. Миналите стойности стават входни сигнали за изкуствената невронна мрежа (Листинг 2.6).

Листинг 2.6: Формиране на входния слой

```
''' Input data loading. '''
for i in range(1, input_size + 1):
    sheet.getCellRangeByName("G" + str(x + i)).setValue(sheet.getCellRangeByName("E" + str(t +
        i)).getValue())
    sheet.getCellRangeByName("G" + str(x + i)).CellBackColor = (255 << 16 | 0 << 8 | 0)
```

По аналогичен начин, бъдещите стойности се зареждат като очакван изход от изкуствената невронна мрежа (Листинг 2.7).

Листинг 2.7: Очакван изход от мрежата

```
''' Expected data loading. '''
for e in range(1, output_size + 1):
    sheet.getCellRangeByName("J" + str(x + e)).setValue(sheet.getCellRangeByName("E" + str(t +
        e + input_size)).getValue())
    sheet.getCellRangeByName("J" + str(x + e)).CellBackColor = (0 << 16 | 127 << 8 | 0)
```

Стойностите, във възлите на скрития слой, са резултат от пресмятане на входните сигнали и текущите стойности на теглата в мрежата (Листинг 2.8).

Листинг 2.8: Стойности на скрития слой при правия пас

```
''' Setup hidden layer. '''
wih = 1
for h in range(1, hidden_size + 1):
    sum = ""
    for i in range(0, input_size + 1):
        sum = sum + "G" + str(x + i) + "*Q" + str(wih)
        wih = wih + 1
        if i < input_size:
            sum = sum + "␣+␣"
    sheet.getCellRangeByName("H" + str(x + h)).setFormula("=TANH(␣" + sum + "␣)")
    sheet.getCellRangeByName("H" + str(x + h)).CellBackColor = (0 << 16 | 0 << 8 | 255)
```

На свой ред, стойностите в изходния слой, са резултат от пресмятане на сигналите в скрития слой и текущите стойности на теглата в мрежата (Листинг 2.9).

Листинг 2.9: Стойности на изходния слой при правия пас

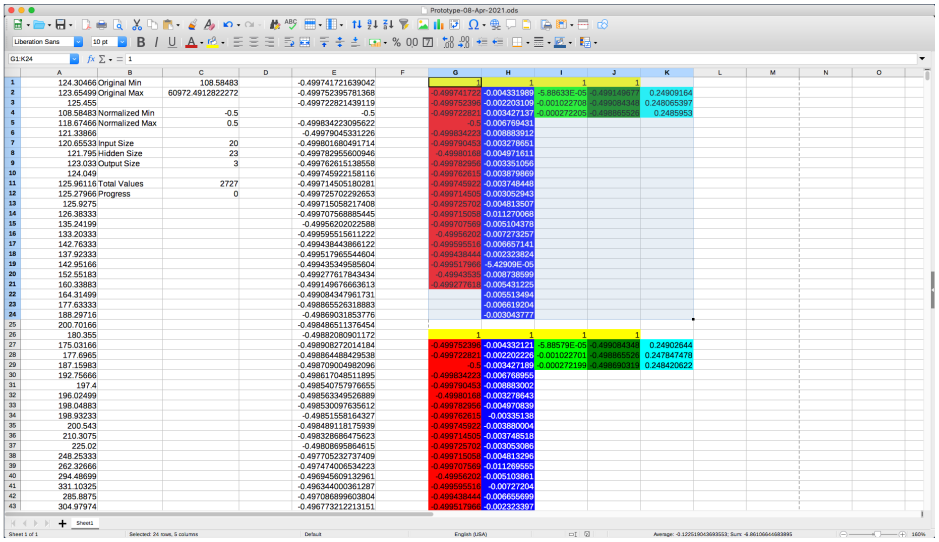
```
''' Setup output layer. '''
who = 1
for o in range(1, output_size + 1):
    sum = ""
    for h in range(0, hidden_size + 1):
        sum = sum + "H" + str(x + h) + "*S" + str(who)
        who = who + 1
        if h < hidden_size:
            sum = sum + "␣+␣"
    sheet.getCellRangeByName("I" + str(x + o)).setFormula("=TANH(␣" + sum + "␣)")
    sheet.getCellRangeByName("I" + str(x + o)).CellBackColor = (0 << 16 | 255 << 8 | 0)
```

Стойностите в изходния слой, спрямо очакваните стойности, дават грешката, която изкуствената невронна мрежа допуска за конкретния тренировъчен пример (Листинг 2.10).

Листинг 2.10: Стойност на грешката допусната от мрежата за конкретния пример

```
        ''' Network output error. '''
        for r in range(1, output_size + 1):
            sheet.getCellRangeByName("K" + str(x + r)).setFormula("=(J" + str(x + r) + "-I" + str(x +
                r) + ")*(J" + str(x + r) + "-I" + str(x + r) + ")")
            sheet.getCellRangeByName("K" + str(x + r)).CellBackColor = (0 << 16 | 255 << 8 | 255)
```

Така описаното разполагане по клетките в електронната таблица, се повтаря многократно, така че да се появи фрагмент за всеки тренировъчен пример. Фрагментът съдържа входен слой, скрит слой, изходен слой и очаквани на изхода стойности (Фиг. 2.20).



Фигура 2.20: Фрагменти за тренировъчните примери

Общата грешка, допусната от мрежата при всички тренировъчни примери, е на базата на средно-квадратична грешка (Листинг 2.11).

Листинг 2.11: Обща средно-квадратична грешка на мрежата

```
    ''' Network total error. '''
    sheet.getCellRangeByName("M1").setFormula("=SQRT(SUM(K:K)/COUNT(K:K))")
    sheet.getCellRangeByName("M1").CellBackColor = 0
```

Два региона клетки в листа на електронната таблица, се определят за стойностите на теглата в мрежата, както и обратно мащабиране към оригиналните стойности (Листинг 2.12).

Листинг 2.12: Определяне на региони за теглата на мрежата

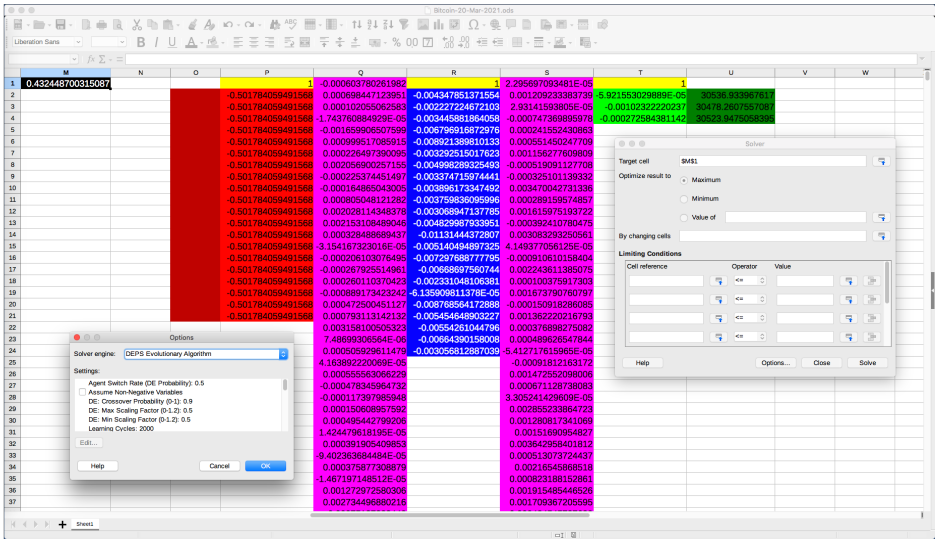
```
    ''' Setup hidden layer weights. '''
    wih = 1
    for h in range(2, hidden_size + 2):
        for i in range(1, input_size + 2):
            sheet.getCellRangeByName("Q" + str(wih)).CellBackColor = (255 << 16 | 0 << 8 | 255)
            wih = wih + 1

    ''' Setup output layer weights. '''
    who = 1
    for o in range(2, output_size + 2):
        for h in range(1, hidden_size + 2):
            sheet.getCellRangeByName("S" + str(who)).CellBackColor = (255 << 16 | 0 << 8 | 255)
            who = who + 1

        sheet.getCellRangeByName("U" + str(o)).setFormula("=$C$1 + ($C$2 - $C$1) * ((T" + str(o) + " - 
            $C$4) / ($C$5 - $C$4))")
        sheet.getCellRangeByName("U" + str(o)).CellBackColor = (0 << 16 | 127 << 8 | 0)
```

За да не се блокира графичния потребителски интерфейс, „разгръщането“ на модела на изкуствената невронна мрежа се извършва в отделна нишка (Листинг 2.13).

Листинг 2.13: Изпълнение с отделна нишка

```
def BuildAnnModel():
    thread = Thread(target=ThreadWorker, args=(XSCRIPTCONTEXT.getDesktop(),))
    thread.start()
```

Търсенето на оптимални тегла става чрез избор на клетката, за която ще се търси минимална стойност (общата допусната грешка от мрежата), както и с избор на клетките, които Solver модулът може да променя, така че да удовлетвори търсенето на минимума (стойностите на теглата в мрежата). От диалоговия прозорец за настройка на Solver модула могат да се изберат различни параметри за алгоритмите еволюция на разликите и оптимизация с рояк на частици (Фиг. 2.21).



Фигура 2.21: Избор на клетки за оптимизация и параметри на оптимизиращите алгоритми

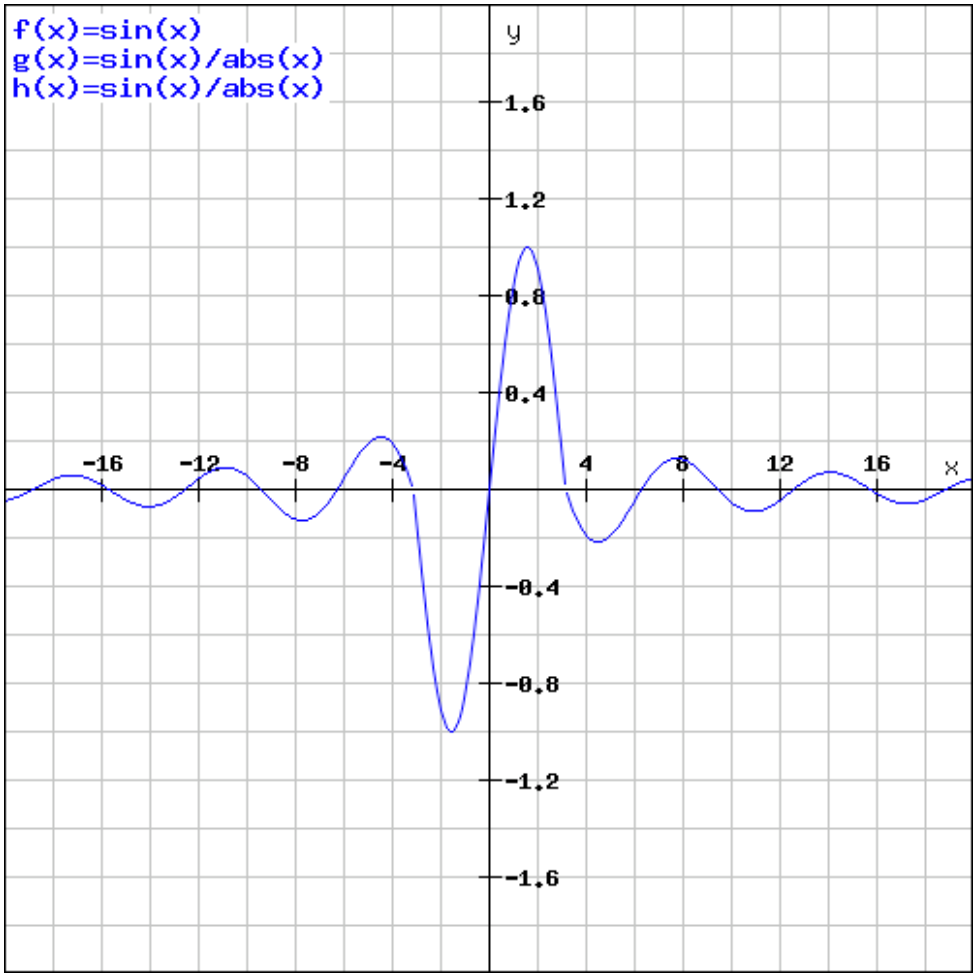
2.5 Алтернатива на производна за активационната функция в изкуствени невронни мрежи

Изкуствените невронни мрежи придобиват своята популярност в края на 20 век. По своята същност те представляват насочен тегловен граф. Мрежите работят в два режима – обучение и изпълнение. Процесът на обучение се състои в търсенето на такива стойности за теглата в графа, че мрежата да съпоставя максимално добре информация от входа към информация на изхода. При класическите трислойни мрежи, информацията се разпространява от входа към изхода, като всеки възел приема сигнали от възлите в предходния слой. Тези сигнали се получават от сумираща функция, която най-често е линейна (умножение на сигнала по тегловния коефициент за връзката между два възела). Резултатът от събраните входни сигнали се подава на прагова функция, която определя нивото на активация, което всеки възел има.

Съществуват множество активационни функции, които са наложени в практиката. Тук се разглежда затихваща функция с периодичен характер (Фиг. 2.22). Периодичният компонент е в следствие на синус компонентата. При точните числени алгоритми за обучение, какъвто е алгоритъмът с обратно разпространение на грешката,

от основно значение е производната на активационната функция. От стойностите на първата производна пряко зависи разликата, с която ще се променят теглата в процеса на обучението.

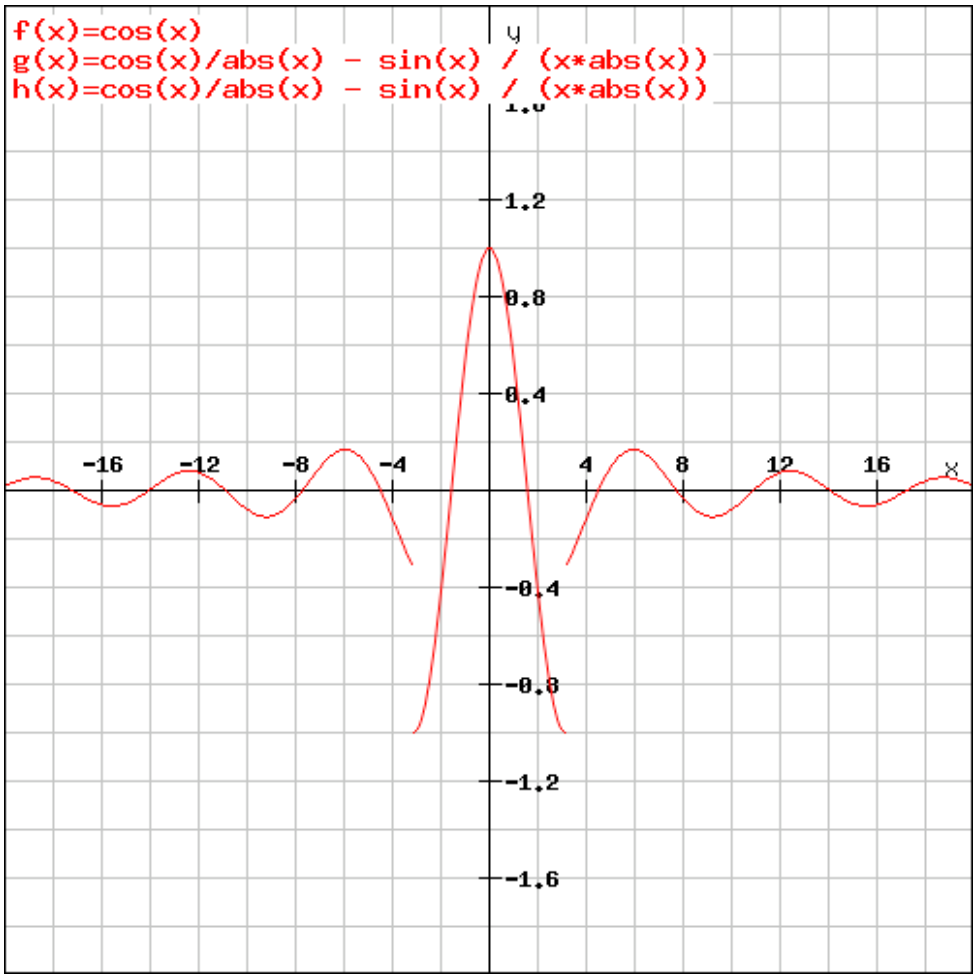
В ситуации, когато първата производна също притежава периодична компонента (Фиг. 2.23), е възможно чисто механично тази първа производна да бъде подменена. Такава подмяна е представена и е възможна благодарение на начина, по който библиотеката за работа с невронни мрежи Encog е организирана.



Фигура 2.22: Активационна функция затихваща синусоида

Всеки неврон в изкуствената невронна мрежа определя нивото на своята активност според нормираща функция, известна под термина активационна функция. Изходът на всеки неврон е важно да се нормализира в определен интервал (най-често между нула и единица или между минус единица и плюс единица), тъй като раз-

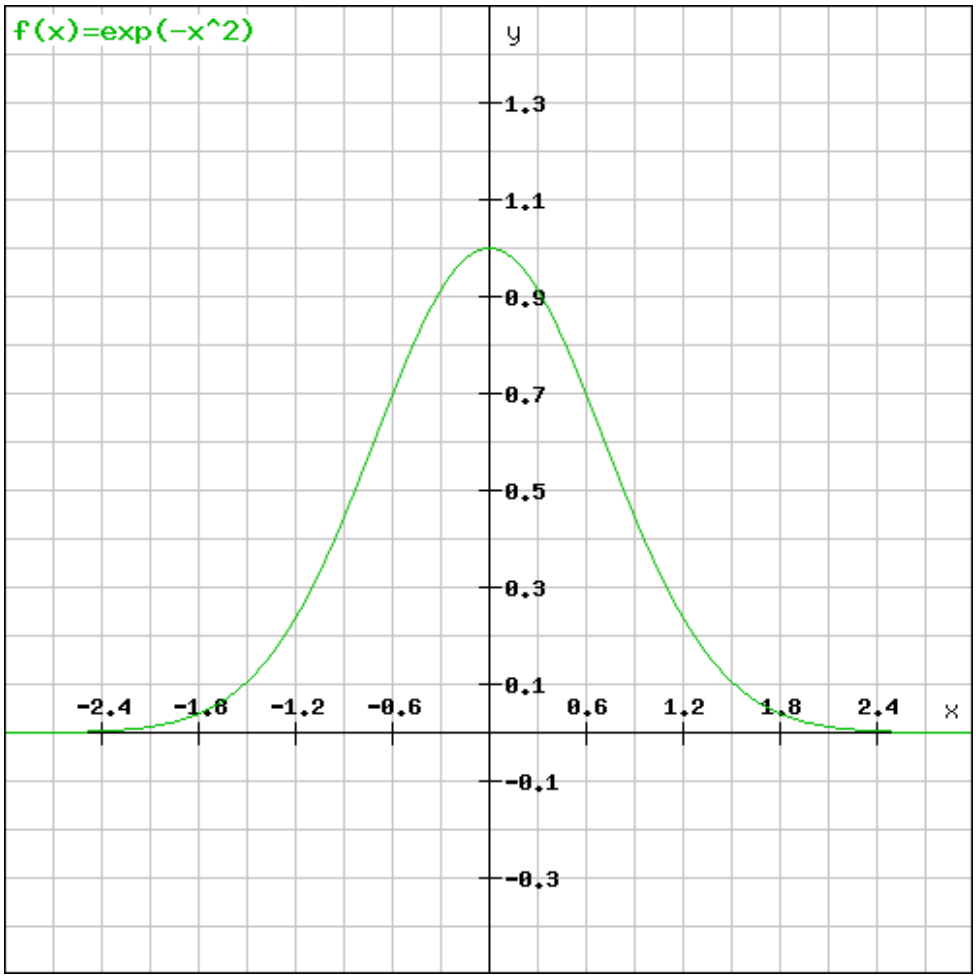
личните слоеве на изкуствените невронни мрежи имат различен брой неврони и без нормализация подаваните сигнали към следващ слой биха били несъразмерни.



Фигура 2.23: Първа производна на активационната функция от тип затихваща синусоида

Активационната функция от Фиг. 2.22 е така подбрана, че да имитира естествените процеси на насищане в природата. При положителна сума на входа, невронът реагира с положителна стойност на изхода си. При отрицателна сума на входа, невронът реагира с отрицателна стойност на изхода си. В същото време, ако входните сигнали са твърде интензивни (както в отрицателна, така и в положителна посока), симулираният процес на насищане не позволява на неврона да излъчи сигнал. Така организираната активация позволява на невроните в един слой да си разпределят отговорностите значително по-равномерно, а това от своя страна води до по-

балансирано представяне на информацията в мрежата.

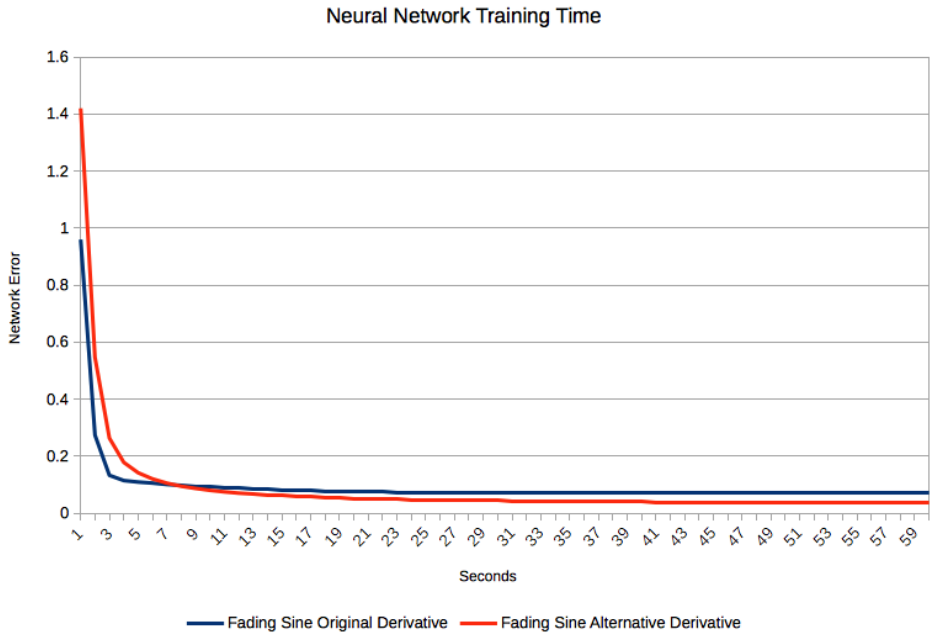


Фигура 2.24: Алтернатива за първа производна на активационната функция от тип затихваща синусоида

Когато активационната функция има периодична компонента, това дава отражение и на нейната първа производна. От една страна, се забелязва периодичният компонент на първата производна, но също така ясно се виждат и две точки на прекъсване (Фиг. 2.23). В следствие на тези две усложнения, обучението с обратно разпространение на грешката води до по-бавна сходимост на алгоритъма. Елегантен подход за ускоряване на процеса е подмяната на първата производна с функция, близка по форма, но без периодична компонента и без точки на прекъсване $f(x) = \exp(-x^2)$ (Фиг. 2.24). Както е видно, алтернативната функция не само има по-подходящи математически свойства, но и се пресмята по-бързо от оригиналната

производна.

Извършените експерименти и постигнатите резултати са осъществени с програмната библиотека Encog, предназначена за изследване на изкуствени невронни мрежи. За целите на експеримента е използвана трислойна изкуствена невронна мрежа, без обратни връзки, с топология 256-64-10. Задачата на мрежата е да извърши класификация на ръчно написани символи (цифрите от 0 до 9). Данните са взети от публично достъпно хранилище и представляват 1593 сканирани изписвания, направени от 80 човека. Сканираните изображения на цифрите са преоразмерени и подготвени като растерна матрица от 16x16 клетки.

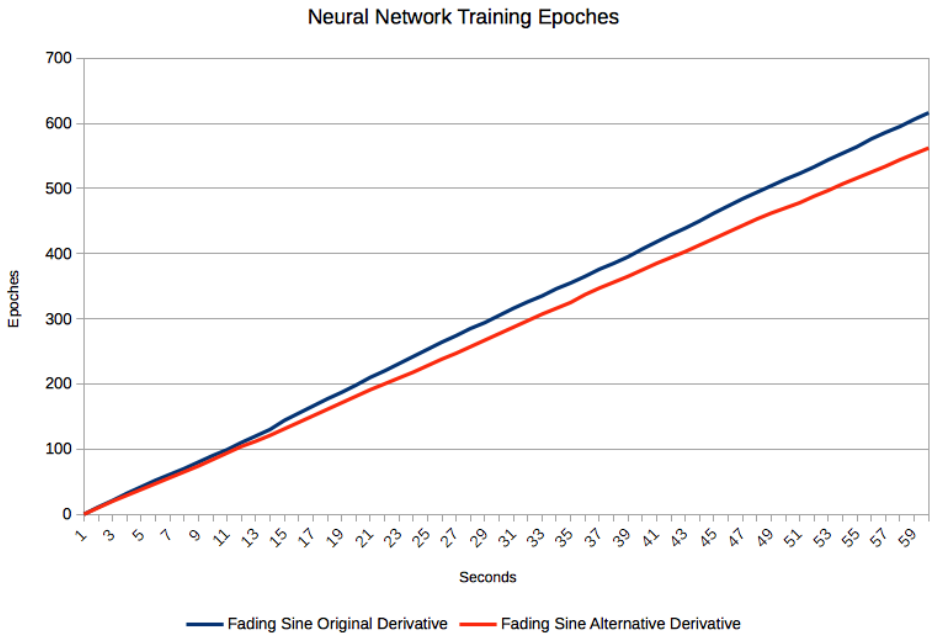


Фигура 2.25: Грешка допусната от мрежата

Експериментите са проведени на настолна компютърна конфигурация с операционна система macOS Sierra 10.12.2, процесор 2.3 GHz Intel Core i5 и 8 GB RAM памет и библиотека Encog Core v3.3.0 - Java Version, Java 8 Update 112 (64bit).

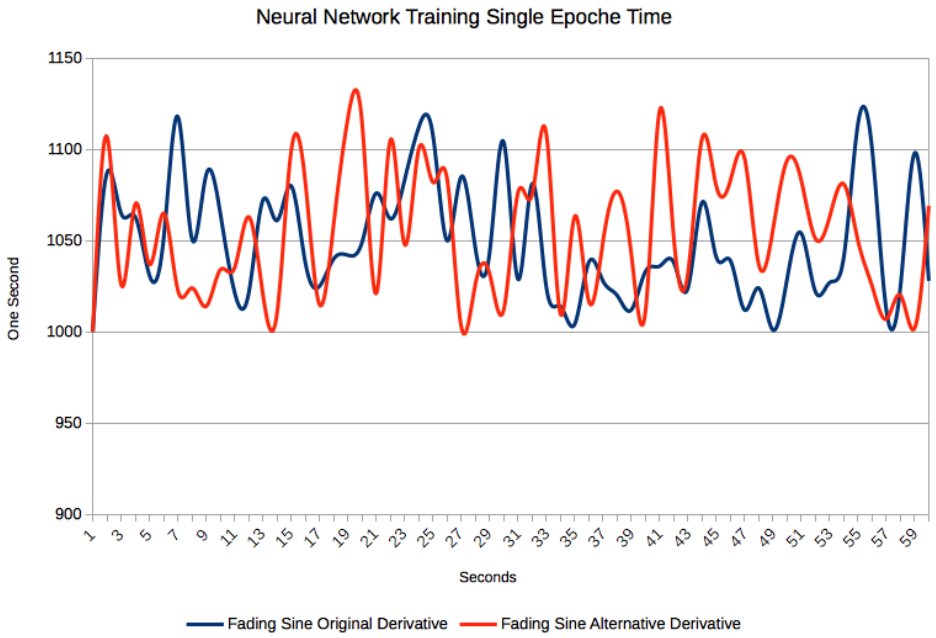
Един и същ експеримент е проведен с две различни функции, като първа производна на активационната функция. Целта е изкуствената невронна мрежа да работи като класификатор, разделяйки входните данни в десет предварително известни класа. И двата експеримента са извършени в рамките на 10 минути обучение с 600

отделни замервания. Съществената сходимост на процеса се случва в рамките на първата една минута (Фиг. 2.25). Още през първите 10 секунди се забелязва, че използването на алтернативната функция води до по-бързо намаляване на грешката, която изкуствената невронна мрежа допуска. Това подобрение е възможно дори при положение, че за един и същи период от време са проведени по-малко тренировъчни епохи с алтернативната функция (Фиг. 2.26, долната линия).



Фигура 2.26: Брой епохи

Замерване на стойностите по време на експеримента се прави на всяка секунда. Тъй като процеса на обучение не бива да бъде прекъсван, се позволява надхвърляне на времето от една секунда. Това води до флуктуации, които са показани на Фиг. 2.27. По отношение на този критерий, алтернативната функция има идентично поведение спрямо оригиналната производна.



Фигура 2.27: Отклонение от секундата

Предложената алтернатива за първата производна на синусоидално затихващата активационна функция води до по-добра сходимост по градиента на грешката, която се допуска от изкуствената невронна мрежа. Също така, бързодействието при пресмятането на алтернативната функция е по-добро, тъй като от изчислителна гледна точка нужните компютърни инструкции са по-малко и по-бързо изпълними [128].

2.6 Бавно изчислявани целеви функции

При оптимизационни проблеми от реалната практика често се налага пресмятане на целеви функции, изискващи по-голямо време за изчисления. Най-често това са комбинаторни задачи [64], задачи с Монте-Карло симулациите или задачи с пълно изчерпване [58].

В хазартната индустрия едни от най-разпространените игри са игри от тип ротативка. При тези игри на екрана се подреждат символи, които образуват печеливши комбинации, при предварително зададени печеливши шаблони. Когато се проектира хазартната игра, е необходимо в правоъгълна матрица да бъдат подредени първо-

начално заложените символи. Тази матрица се нарича виртуален барабан и по своята същност е дискретно вероятностно разпределение. Виртуалните барабани пряко определят математическото очакване за средно ниво на печалбите. Този параметър в хазартната индустрия е познат под термина Return To Player (RTP) и представлява процентно отношение на общо спечелена сума към общо загубена сума. Процентът на печалбите подлежи на държавна регулация навсякъде по света, където хазартната индустрия е легално позволена. Поради тази причина, подреждането на символите върху виртуалните барабани е комбинаторна, оптимизационна задача. Целта е да се постигне желаният заложен RTP процент. Целевата функция при проектирането на виртуални барабани най-често е Монте-Карло симулация, но в някои ситуации е възможно и използването на алгоритъм за пълно изчерпване, чрез генериране на всички възможни игрови екрани. И двата варианта за изчисление на целевата функция изискват значително време за изчисляване. За да се приложи генетичен алгоритъм в такава задача, е необходимо да се търси начин за компенсация на бавно пресмятаната целева функция.

Листинг 2.14: Алгоритъм на централния възел

```
unsigned long counter = 0;
if(rank != ROOT_NODE) {
        return;
}

GeneticAlgorithm global;
std::map<int,GeneticAlgorithm> populations;
do {
        for(int r=0; r<size; r++) {
                if(r == ROOT_NODE) {
                        continue;
                }

                if(counter == 0) {
                        GeneticAlgorithmOptimizer::addRandomReels(global, model, targets
                                        , LOCAL_POPULATION_SIZE*size);
                        GeneticAlgorithm ga;
                        global.subset(ga, LOCAL_POPULATION_SIZE);
                        populations[r] = ga;
                } else {
                        if(rand()%(NUMBER_OF_BROADCASTS/10) == 0) {
                                GeneticAlgorithm ga;
                                global.subset(ga, LOCAL_POPULATION_SIZE);
                                populations[r] = ga;
                        }
                }
                const std::string &value = populations[r].toString();
                MPI_Send(value.c_str(), value.size(), MPI_BYTE, r, DEFAULT_TAG, MPI_COMM_WORLD)
```

```
                        ;
                }

                for(int r=0; r<size; r++) {
                        if(r == ROOT_NODE) {
                                continue;
                        }

                        GeneticAlgorithm ga;
                        MPI_Recv(buffer, RECEIVE_BUFFER_SIZE, MPI_BYTE, r, DEFAULT_TAG, MPI_COMM_WORLD,
                                MPI_STATUS_IGNORE);
                        ga.fromString(buffer);
                        populations[r] = ga;
                        if(ga.getBestFitness() < global.getBestFitness()) {
                                global.setChromosome( ga.getBestChromosome() );
                        }
                }

                counter++;
        } while(counter < NUMBER_OF_BROADCASTS);
```

Тъй като генетичните алгоритми позволяват много висока степен на паралелна обработка, това ги прави идеални за реализацията под формата на паралелни пресмятания

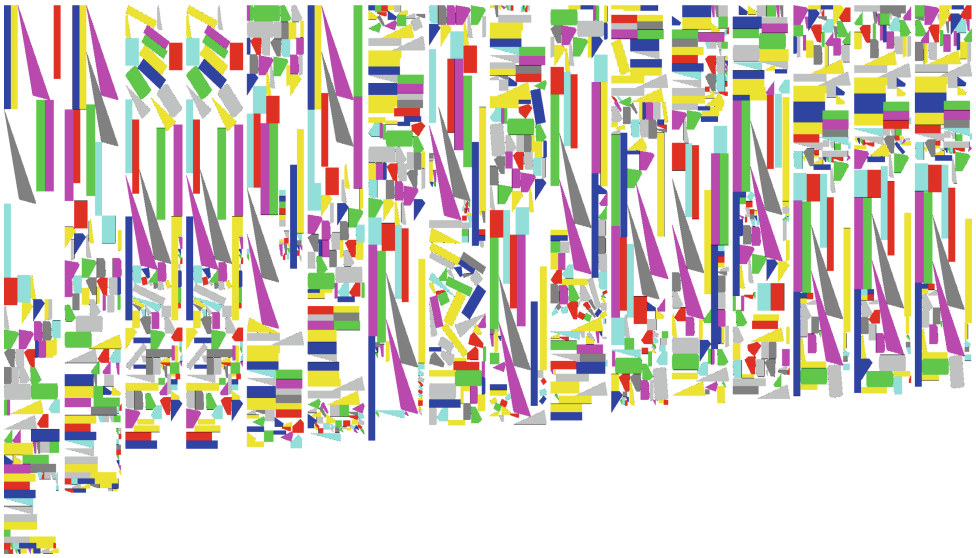
Листинг 2.15: Алгоритъм на работните възли

```
        unsigned long counter = 0;
        if(rank == ROOT_NODE) {
                return;
        }

        do {
                GeneticAlgorithm ga;
                MPI_Recv(buffer, RECEIVE_BUFFER_SIZE, MPI_BYTE, ROOT_NODE, DEFAULT_TAG, MPI_COMM_WORLD,
                        MPI_STATUS_IGNORE);
                ga.fromString(buffer);

                GeneticAlgorithmOptimizer::optimize(ga, model, targets, LOCAL_OPTIMIZATION_EPOCHES);

                std::string result = ga.toString();
                MPI_Send(result.c_str(), result.size(), MPI_BYTE, ROOT_NODE, DEFAULT_TAG, MPI_COMM_
                    WORLD);

                counter++;
        } while(counter < NUMBER_OF_BROADCASTS);
```

Всеки работен възел получава своя локална популация, извършва цикъл за оптимизация и изпраща резултати към централния възел (Листинг 2.15).

При задачите за двуизмерен разкрой, неправилни двумерни геометрични фигури трябва да се разположат оптимално , без да се пресичат, в обща двумерна плоскост. Задачата е комбинаторна и е свързана с позиции на множество геометрични фигури,

които включват координати на всяка от фигурите и ъгъл на ротация. Целта при този вид задачи е фигурите така да се уплътнят, че да използват най-малка площ от общата плоскост. Този вид задачи са свързани с намаляване на загубата от използвания материал, когато фигурите се изрязват от него. Някои материали, като благородни метали, стомана или скъпи кожи е важно да бъдат оптимално използвани, така че да се снижат производствените разходи.



Фигура 2.28: Междинни състояния при оптимизация за двуизмерен разкрой

Генетичните алгоритми намират приложение и при решаването на задачите от двуизмерния разкрой. Двумерните геометрични фигури се представят като полигони, които са елементи на хромозомите. Всяка от фигурите се характеризира с (x,y) координати и ъгъл на ротация около собствения център. При операцията по кръстосване се запазва същият набор от двуизмерни фигури, които съставят задачата. При мутацията фигурите менят координатите си и/или ъгъла на ротация

2.7 Класификация на потребителски вот от страна на сървъра в разпределени изчисления тип човек-компютър

В по-голямата си част, разпределените изчисления не изискват човешка намеса. В една по-малка част от проектите се появява нуждата човек да извърши субективна оценка на пресметнатите резултати. Такива проекти са проектите в областта на изкуството, където субективните понятия „красиво“ и „грозно“ не могат да бъдат изчислени с компютър. В същата група са и проекти, където се разчита на човешката интуиция. Точно такава е сферата и на финансовото прогнозиране. Не рядко, опитни търговци имат усет за посоката, в която ще се променят цените, без да могат да обяснят на какво се дължи това тяхно интуитивно убеждение.

Съвременните възможности на мобилните устройства с Android OS предлагат начини за събиране на потребителски вот. Тук се предлагат иновативни разпределени изчисления човек-компютър за събиране на потребителски вот. На екрана на Android устройството ежедневно се показва известие и потребителят е провокиран да гласува или надолу или нагоре за промяна на стойността на конкретна валутна двойка. Информацията, предоставена от потребителите, се събира на олекотено сървърно приложение с основа PHP/MySQL.

За да може гласът на потребителите да се използва за бъдещо прогнозиране, събраната информация трябва да бъде класифицирана според честота на гласуване за един потребител и процент на успех на всеки подаден глас. Тази задача може да бъде ефективно решена със самоорганизиращи се карти на Кохонен. Тук интерес представляват четири групи (клъстъри):

1) Потребители с ниска честота на гласуване и нисък процент на познаване;

2) Потребители с ниска честота на гласуване, но висок процент на познаване;

3) Потребители с висока честота на гласуване, но нисък процент на познаване;

4) Потребители с висока честота на гласуване и висок процент на познаване.

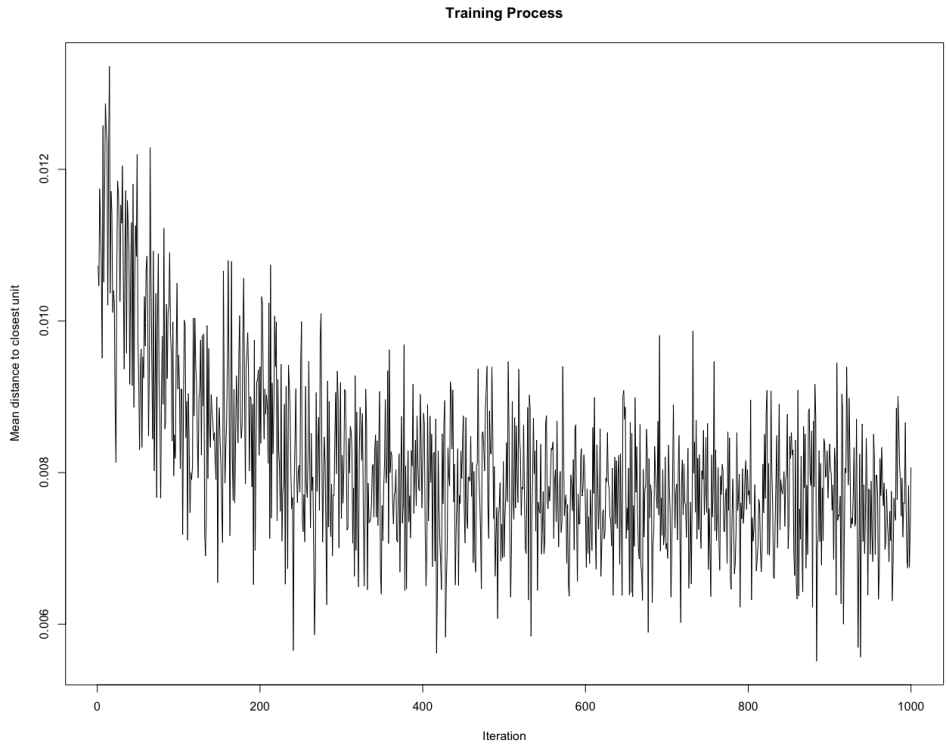
Намирането на границите на четирите групи е сложно и затова самоорганизиращите се карти са подходящ инструмент за подобна задача по групиране. Мрежата се обучава с информацията, събрана от различни потребители с различна дейност. Някои от потребителите имат по-дълъг период на използване на системата и те осигуряват по-добро разпределение на информацията във времето. Брой успешни предположения, брой на неуспешните предположения и процентът на дните с гласуване се предоставят като вход на мрежата.

Финансовото прогнозиране може да бъде много сложно, поради големия брой зависими фактори. Дори успешните потребители правят грешки в прогнозите си. За всеки потребител в системата се изчислява личен процент на успех, но когато бъдещата прогноза трябва да бъде изчислена, всеки потребител участва със своя рейтинг и корекционен коефициент, даден на една от четирите групи, в които е класиран. Чрез този процес на установяване на прогнозата, група хора предлага субективно мнение за бъдещата промяна на стойността на валутната двойка. Някои хора гласуват според знанията си, други гласуват според личните си изчисления, а трета група хора просто гласуват, вдъхновени само от интуицията си, дори когато не са в състояние да обяснят защо мислят така. В психологията е добре известно, че група хора могат да постигнат по-добри резултати, разчитайки на интуицията си, спрямо експертни решения в дадена област.

#	Participation Rate	Guess Rate	Class	#	Participation Rate	Guess Rate	Class	#	Participation Rate	Guess Rate	Class
1	0.38	0.49	1	11	0.57	0.47	3	21	0.30	0.49	4
2	0.45	0.42	3	12	0.12	0.55	2	22	0.34	0.56	4
3	0.33	0.44	1	13	0.51	0.45	3	23	0.33	0.45	1
4	0.55	0.51	3	14	0.36	0.46	1	24	0.42	0.44	1
5	0.39	0.40	1	15	0.25	0.49	2	25	0.46	0.47	3
6	0.51	0.54	3	16	0.37	0.53	4	26	0.48	0.49	3
7	0.50	0.51	3	17	0.43	0.55	3	27	0.25	0.42	2
8	0.55	0.47	3	18	0.20	0.46	2	28	0.52	0.48	3
9	0.10	0.48	2	19	0.32	0.52	4	29	0.42	0.50	3
10	0.28	0.49	4	20	0.57	0.47	3				

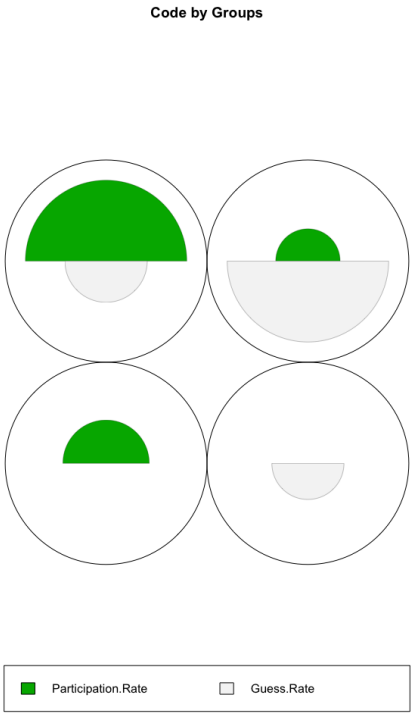
Фигура 2.29: Експериментални данни

Експерименти са направени с 29 потребители, които са гласували за движението на валутната двойка EUR/USD. За всеки участник процентът на участие се измерва чрез преброяване на дните на участие, разделени на всички дни на експеримента. Другият измерен параметър е степента на успех при прогнозите. Събраните данни са представени на Фиг. 2.29. Последната колона в набора от данни е класификационната група. Всеки участник е класиран в една от четирите групи.



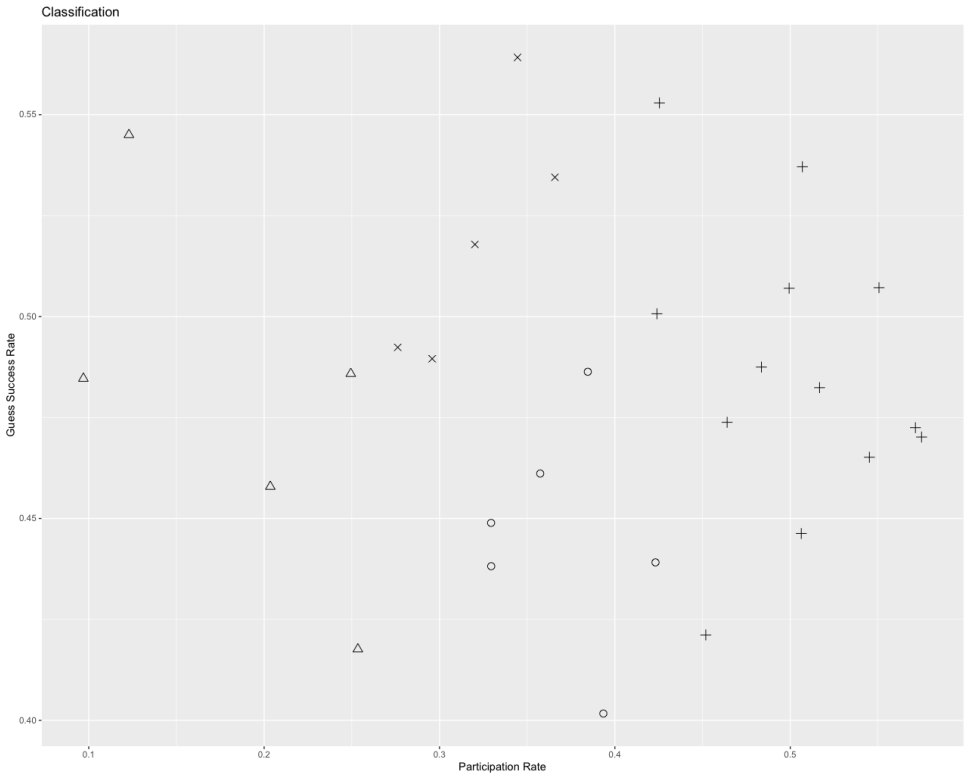
Фигура 2.30: Сходимост на обучението

Като класификатор се използва пакетът "kohonen"в R. За самоорганизираща се карта е използвана квадратна топология 2x2: somgrid(xdim = 2, ydim = 2, topo = "rectangular"). Като скорост на обучение, са избирани двата параметъра да бъдат 0.0005 и 0.0001: alpha = c(0.0005, 0.0001).



Фигура 2.31: Маркировка на групите

Обучителните итерации са избрани да бъдат 1000, както е показано на графиката за конвергенция (Фиг. 2.30). Множеството данни за обучение е сравнително малко и обучението завършва за милисекунди. В първия клас има потребители със сравнително нисък процент на участие и сравнително нисък процент на познаване. Във втория клас има потребители с нисък процент на участие и но с висок процент на познаване. В третия клас има потребители с висок процент на участие, но със сравнително нисък процент на познаване. В четвъртия клас има потребители със сравнително висок процент на участие и със сравнително добро ниво на познаване. Класовете са показани на Фиг. 2.32. Класификацията, направена от R със самоорганизиращи се карти, се визуализира като маркировъчен код на Фиг. 2.31 [108].



Фигура 2.32: Разпределение по класове

Разпределените изчисления от тип човек-компютър могат да бъдат много обещаващ инструмент за финансово прогнозиране. Търговската интуиция е нещо, което не е постижимо със съвременните компютри. Обобщеното мнение на група хора, в комбинация с техните знания и обработка на подсъзнателна информация, може да доведе до много по-надеждни прогнозни резултати. С възможностите на съвременни мобилни устройства и безжични комуникационни канали, разпределените изчисления, базирани на човек-компютър, стават широко достъпни и финансово рентабилни. Най-големият недостатък на груповото вземане на решения е, че е трудно всеки път да се докаже правилността на взетите решения и е трудно всеки път да се повтори едно и също решение.

2.8 Обобщение

Изследвани са възможностите за подобряване на алгоритъма за селекция в генетичните алгоритми. Прилага се идея за рекурсивно спускане във възли от дървовидна структура, като всеки възел се характеризира с под популация. Във всеки възел се извършва пълно изчерпване за рекомбинация на индивидите в прилежащата на възела подпопулация. Пълното изчерпване се комбинира и с локално търсене, с цел допълнително подобряване на резултатите в конкретния възел. Постигнатите резултати са представени в [107, 109]. В публикация [107] авторът на настоящия дисертационен труд има 1/3 принос, който се състои в предлагането на представената идея и написването на програмния код за извършване на експериментите, поради което е и водещ автор в публикацията. В публикация [109] авторът на настоящия дисертационен труд има 1/3 принос, който се състои в предлагането на представената идея и написването на програмния код за извършване на експериментите, поради което е и водещ автор в публикацията. Изследвани са възможностите за напасване на криви към множество точки с помощта на уравнение за права и ред от синус функции. Постигнатата крива се използва за генериране на прогноза, извън диапазона на известните измерени точки. Постигнатите резултати са представени в [115]. В публикация [115] авторът на настоящия дисертационен труд има 1/3 принос, който се състои в предлагането на представената идея. Изследвани са възможностите за бързо прототипиране на изкуствени невронни мрежи, с помощта на рояк от частици и еволюция на разликите. Постигнатите резултати са представени в [110]. Извършено е изследване на възможностите за класифициране на потребителския вот при прогнозиране на финансови времеви редове. Постигнатите резултати са представени в [108]. В публикация [108] авторът на настоящия дисертационен труд има 1/3 принос, който се състои в предлагането на представената идея и написването на програмния код за извършване на експериментите, поради което е и водещ автор в публикацията.

Глава 3

Софтуерна система за прогнозиране с ИНМ на времеви редове

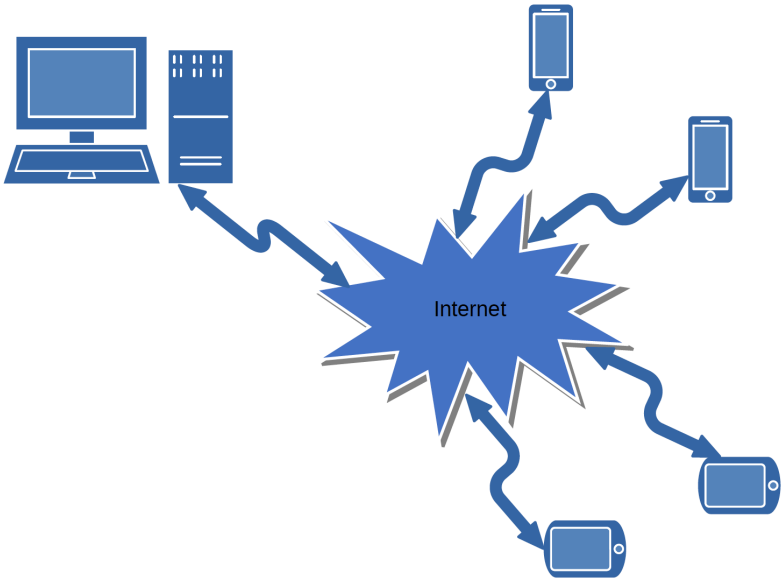
На база на разгледаните алгоритми и системи за прогнозиране, изборът за изработка на софтуерна система за прогнозиране пада върху клиент-сървър базирана архитектура, в която мобилни устройства извършват изчисления на изчислими пакети. Изчисленията се организират под формата на дарена изчислителна мощност. Информацията за времевите редове се получава от отдалечен уеб сървър. Какви конкретни времеви редове ще бъдат заложени в системата, зависи от операторите на сървърната система.

3.1 Архитектура на системата

Най-обобщеното представяне на разработената система съдържа трите най-важни компонента – сървър, комуникационна среда и мобилни устройства (Фиг. 3.1). Комуникационната среда е глобалната мрежа Интернет. Информацията, която трябва да бъде изчислена, основно се намира на специално предназначен за нуждите на системата сървър. Изчислителните възли в системата са умни мобилни устройства.

Основна цел при разработването е крайното решение да бъде максимално икономически изгодно. Това означава, че разходите за поддържане на цялата инфраструктура трябва да бъдат минимизирани. Тази цел може да бъде постигната, ако раз-

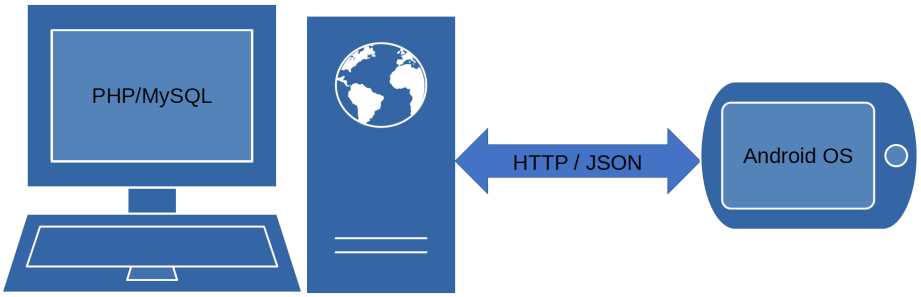
ходите за поддържането на сървър бъдат снижени възможно най-много. Системата трябва да работи с маломощен сървър, който само да синхронизира пресмятанията и да съхранява получените междинни резултати. Тъй като потребителите в системата даряват техните изчислителни ресурси, практически разходите от страна на клиента са сведени до нула. Всеки потребител сам закупува мобилното си устройство, сам заплаща сметката за изразходвания интернет трафик и сам заплаща електричеството, необходимо за опериране на устройството. Поради стремежа за минимално натоварен сървър, вместо закупуването на отделна сървър машина и разполагането ѝ в информационен център, спокойно може да се наеме уеб услуга от тип споделен хостинг (Фиг. 3.2).



Фигура 3.1: Обща организация на системата

При споделения хостинг, върху една физическа машина доставчикът стартира множество виртуални машини и в самите виртуални машини се активират отделни инстанции на уеб сървърни приложения. Най-често използваните от доставчиците на споделен уеб хостинг операционни системи са базирани на Linux. Операционната система Linux много добре се съчетава с уеб сървъра Apache. Към уеб сървъра доставчиците най-често предлагат и MySQL релационна база данни. Apache уеб сър-

вера най-често има настроена поддръжка на PHP интерпретатора. Споделен хостинг в конфигурация Linux-Apache-MySQL-PHP (XAMPP) е едно от икономически най-ефективните решения. Като алтернативи може да се ползват Windows базирани сървъри, които поддържат MS SQL Serve база данни и ASP.NET уеб страници. Възможни са също комбинации с JSP, PostgreSQL или Oracle, но никоя от изброените комбинации не може да постигне икономическата ефективност на XAMPP. Споделеният хостинг, при конфигурация XAMPP, може да се наеме за абонамент от няколко долара на месец.



Фигура 3.2: Софтуерни компоненти

Комуникацията в системата винаги се инициира от клиентското приложение, което отправя заявки към сървъра и връща резултати. Комуникацията се извършва по протокола TCP/IP, като се използват възможностите за HTTP комуникация. По аналогия с RESTful приложенията, клиентът и сървърът обменят съобщения, пакетирани с JSON. Ролята на JSON е в това, информацията да бъде предавана по мрежата в структуриран вид. Възможна е реализация и с XML, но JSON е значително по-добре оптимизиран за комуникация между машини. XML намира по-широко приложение там където е нужна човешка намеса за описване на данните. JSON има и предимството, че обемът служебна информация е значително по-малък в сравнение с XML.

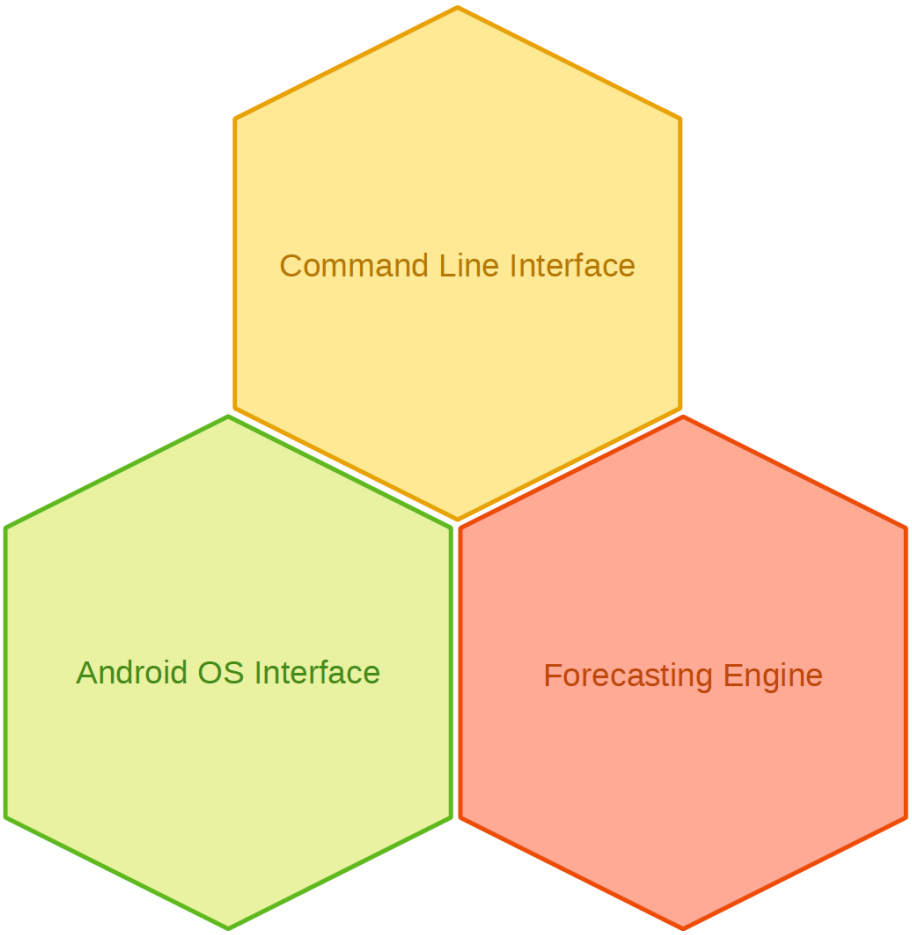
От страната на клиента стои Android OS приложение, разработено към настоящия дисертационен труд, което има основната задача да извършва разпределените изчисления и да докладва получените резултати на отдалечения сървър. Сървърно-

то приложение не е част от настоящия дисертационен труд, а се използва на база на предходна разработка в ИИКТ-БАН [4].

Изборът на мобилни устройства с операционна система Android OS е направен, тъй като тези устройства са много разпространени, със значителен пазарен дял, ядрото на операционната система е Linux базирано, а множество нейни компоненти са с отворен код. От значение е и цената на мобилните устройства, тъй като прекият конкурент iOS, на компанията Apple, поддържа значително по-високи цени. При дарените ресурси за разпределени изчисления е от съществено значение крайните потребители да могат да си позволят цената на изчислителните ресурси. Сериозно предимство на Android OS е и факта, че поддържа програмния език Java. По настоящем, Java е най-широко използваният програмен език, което го прави изключително подходящ, от гледна точка на човешки ресурси, при евентуално разрастване на проекта. В последните години се забелязва много силно наложена тенденция езикът Java да бъде подменен в Android OS с езика Kotlin, но този процес би отнел твърде много време. Предимство на програмния език Java е и това, че той широко се използва не само за програмиране на мобилни приложения, но и на множество други видове софтуер. За сравнение, езиците Objective-C и Swift, на компанията Apple, далеч не се използват толкова много, извън продуктите на самата компания. Това ограничение би довело до сериозни проблеми при търсенето на програмисти за мобилни устройства, базирани на операционната система iOS.

3.2 Модулна организация на мобилното приложение

Разработката на мобилното приложение е организирана в отделни модули. Използването на модули позволява по-гъвкава работа с различните компоненти и по-лесно тестване на написания програмен код. Модулността позволява отделни части от приложението да се използват по различен начин. На най-високото ниво за разделяне са налични три модула – изчислителен модул, Android OS потребителски интерфейс и интерфейс за терминален прозорец на настолна операционна система (Фиг. 3.3).



Фигура 3.3: Основни модули на мобилното приложение

Всички съществени пресмятания се реализират в изчислителния модул. Стремежът в този модул е той да не използва специфичен код от API-то на Android OS или на някоя от операционните системи за настолни компютри. Целта на това разделяне е изчислителният модул да има своята самостоятелност и с лекота да се интегрира в модули за графичен потребителски интерфейс или интерфейс от тип команден ред.

Модулът, реализиращ интерфейс в командния интерпретатор, е полезен при изпитанията на системата, тъй като с него може да се избегне времеемкото зареждане на Android приложение в Android емулатор. Модулът за команден интерпретатор се компилира със стандартните развойни средства на Java, до самостоятелно приложение за настолен компютър. Този начин на работа улеснява търсенето на дефекти, когато съмненията са в начина на работа на Android OS интерфейса и изчислителния

модул.

Модулът за графичен потребителски интерфейс на Android OS има основна задача да извършва пресмятанията на фонов режим (Android Services). Също така, този модул визуализира информацията за междинните резултати от пресмятанията. Модулът извършва съхраняването и зареждането на локалната информация в SQLite релационна база данни. Дава и възможности за настройки за начина, по който функционира мобилното приложение.

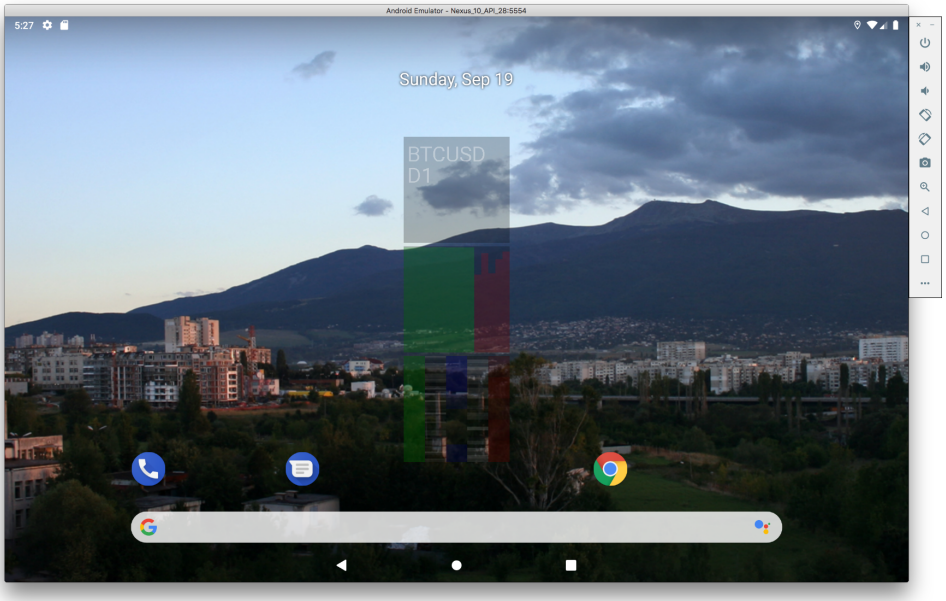
Фигура 3.4: Компоненти в GUI модула

3.2.1 Компоненти в модула за графичен интерфейс

Комуникацията на Android OS с компонентите на модула за графичен интерфейс става през няколко компонента. В Android е възприета концепцията на уеб страни-

ците, където няма единична точка за стартиране на приложението, а различните компоненти могат да бъдат извикани по отделно, точно както може да се отворят различни страници, а не задължително заглавната страница.

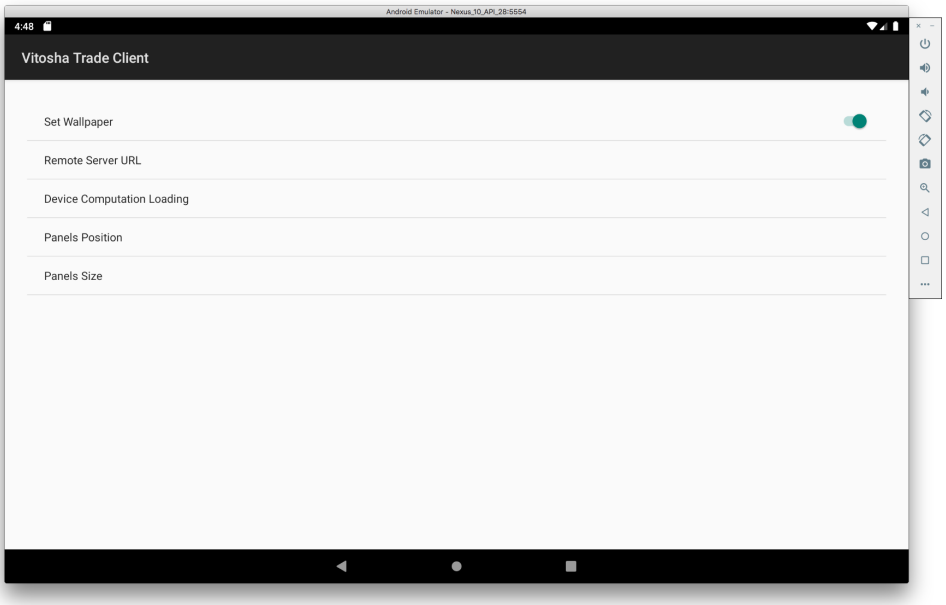
На този етап от завършеността на системата, активният тапет е основният компонент на мобилното приложение. Активните тапети задават фоново изображение, което стои зад всички компоненти на работния плот (Фиг. 3.5). Тапетите са активни, защото по своята същност не са статично изображение, а цялостна работеща програма, която позволява динамично изрисуване в растерно платно. Активният тапет се управлява от операционната система чрез услуга/демон (Android Services). Услугите в Android са програми, които не притежават свой собствен потребителски интерфейс. Услугите са аналог на демон процесите в операционните системи за настолни компютри.



Фигура 3.5: Android Live Wallpaper

Тъй като използването на устройството във фонов режим, за извършване на изчисления, може да забави сериозно работата му, то активният тапет разполага с екран за настройки (Фиг. 3.6). В този екран се определя честотата за извършване на пресмятания, както и някои опции, свързани с визуализацията на междинните

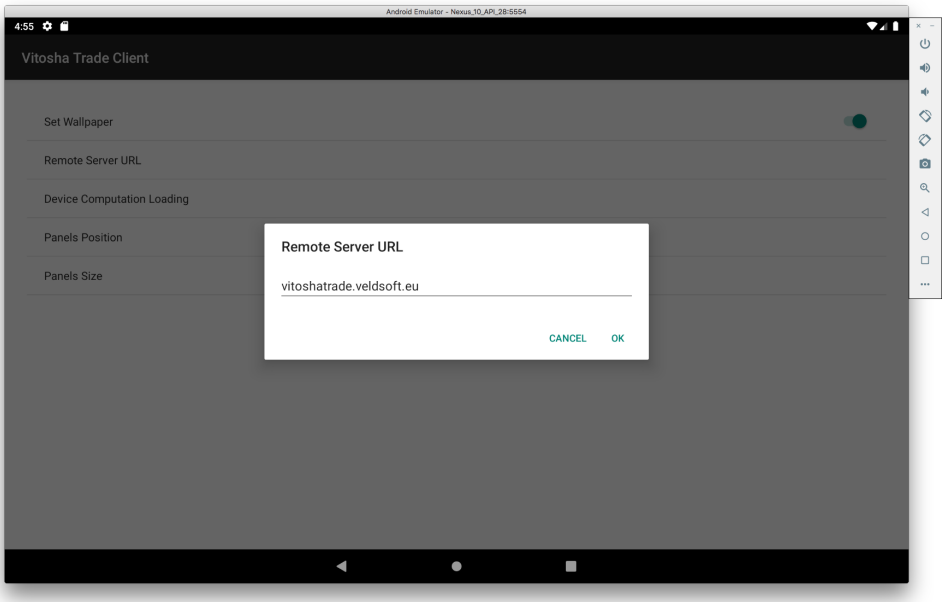
резултати.



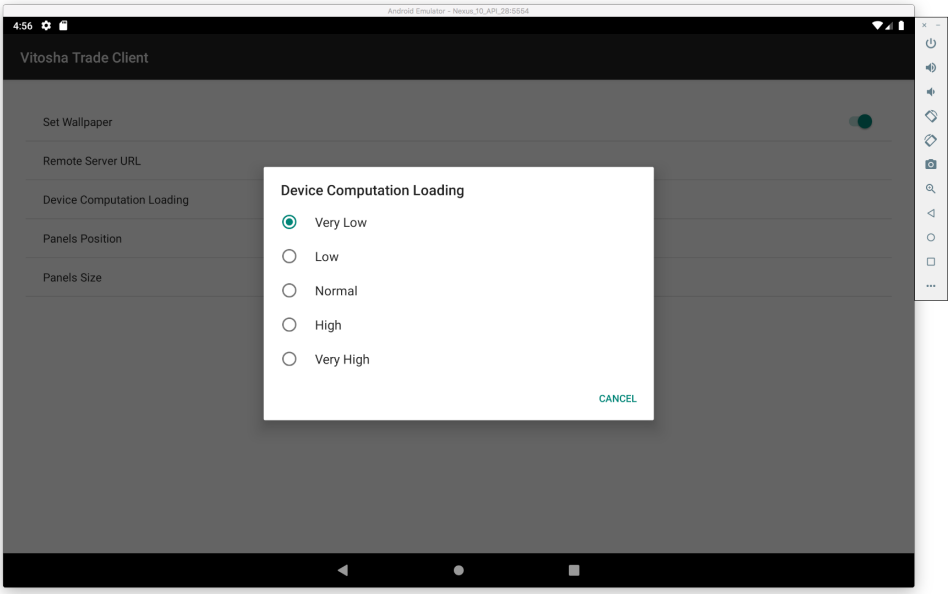
Фигура 3.6: Настройки на активния тапет

Цялата информация за извършване на изчисленията върху мобилното устройство, се получава от отдалечен сървър на конкретно установен URL адрес (Фиг. 3.7). Натоварването на мобилното устройство е организирано в пет възможни степени (Фиг. 3.8). Позициите на панелите за визуализация на междинната информация се определят от списък с възможности (Фиг. 3.9). Размерите на панелите за визуализация са ограничени в три опции (Фиг. 3.10).

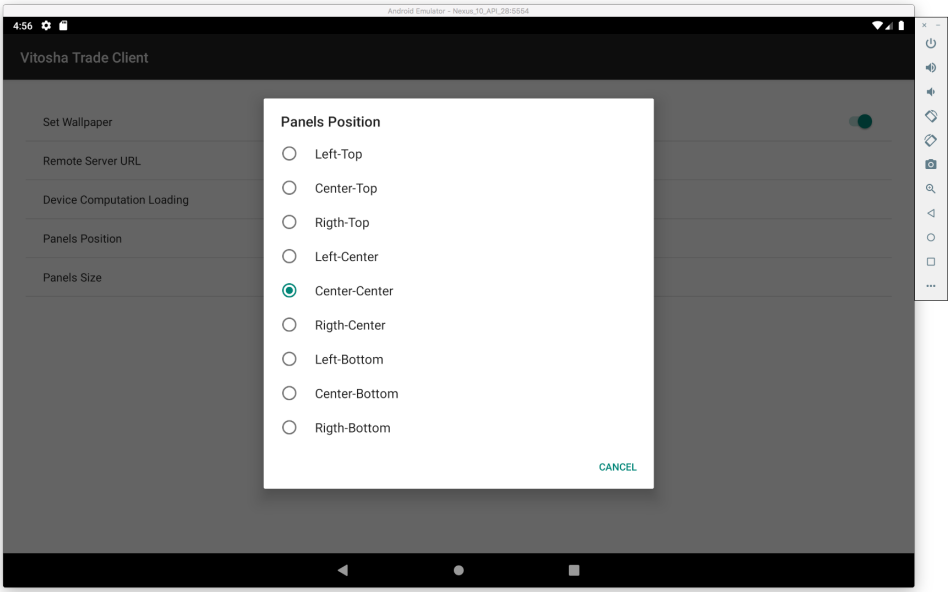
Адресът на отдалечения сървър представлява буквено-символно име, което води към компютър, който изпълнява приложение за уеб сървър и поддържа нужните скриптове за JSON обмен на съобщения с клиентското приложение.



Фигура 3.7: Адрес на отдалечения сървър



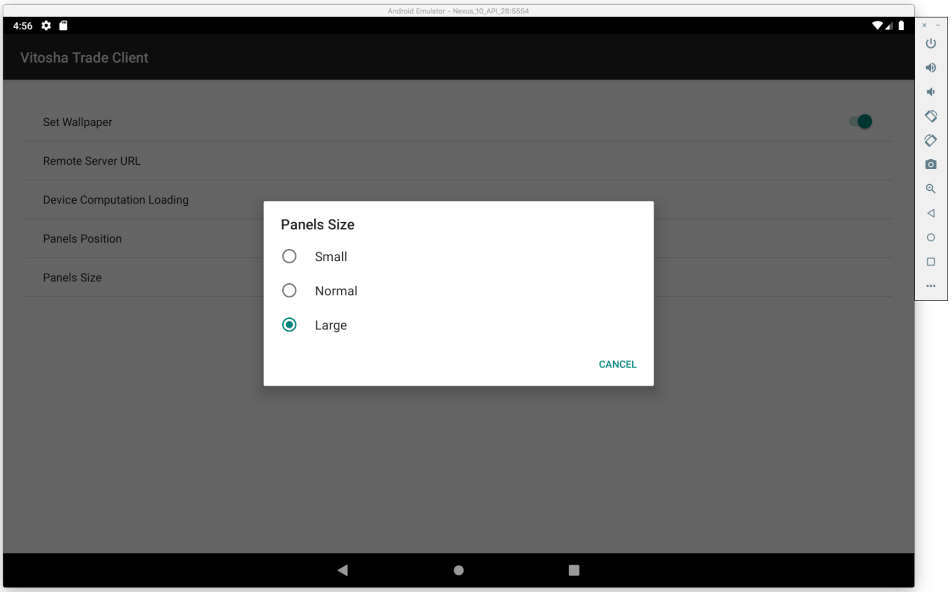
Фигура 3.8: Натоварване на мобилното устройство



Фигура 3.9: Позиция на панелите за визуализация

Натоварването на мобилното устройство се регулира с период, през който да се стартира единичен цикъл за обучение и прогнозиране. Твърде дългите интервали забавят процеса на смятане, но разтоварват мобилното устройство от консумация на енергия и забавяне на общата производителност.

Според организацията на работния плот, всеки потребител има различни предпочитания в коя част на екрана му е удобно да получава фонова визуална информация. Чрез възможността за промяна на позициите, панелите за визуализация могат да са максимално информативни за потребителя. Разликата в размерите на различните екрани води до допълнителна необходимост за размерите на панелите за визуализация. При устройства с големи екрани е рационално панелите да са по-големи, а при устройства с малки екрани е рационално панелите да са с по-малки размери.



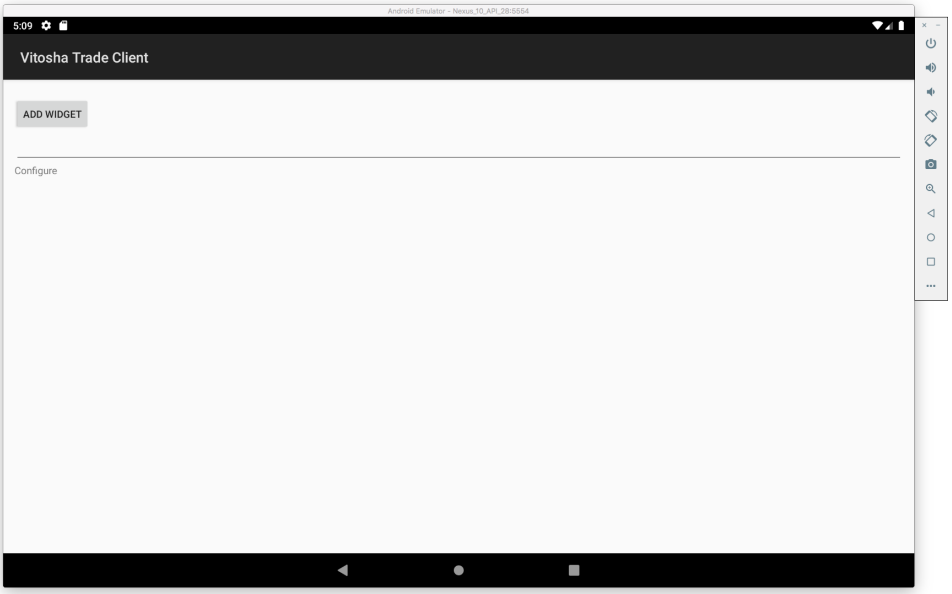
Фигура 3.10: Размери на панелите за визуализация

Фигура 3.11: Компонент за гласуване

Графичният потребителски интерфейс, освен статична визуализация на междинни резултати, позволява и събиране на субективното мнение на потребителя. Това става с помощта на widget за гласуване (Фиг. 3.11). Потребителят избира дали цената на определен актив ще се повиши или ще спадне. Целта е тази информация да

се използва за разпределени изчисления по схемата човек-компютър. Всеки потребител има свое собствено мнение за движението на цената и това мнение най-често е интуитивно. Компонентът за гласуване също има свой екран за настройки (Фиг. 3.12).

При гласуване, в локалната SQLite база данни, се записват точният час на гласуване, за коя валутна двойка е подаден гласът, посоката за промяна на цената, която потребителят е избрал, и уникален идентификатор на потребителя. При наличие на интернет свързаност, същата информация се изпраща и до отдалечения сървър.

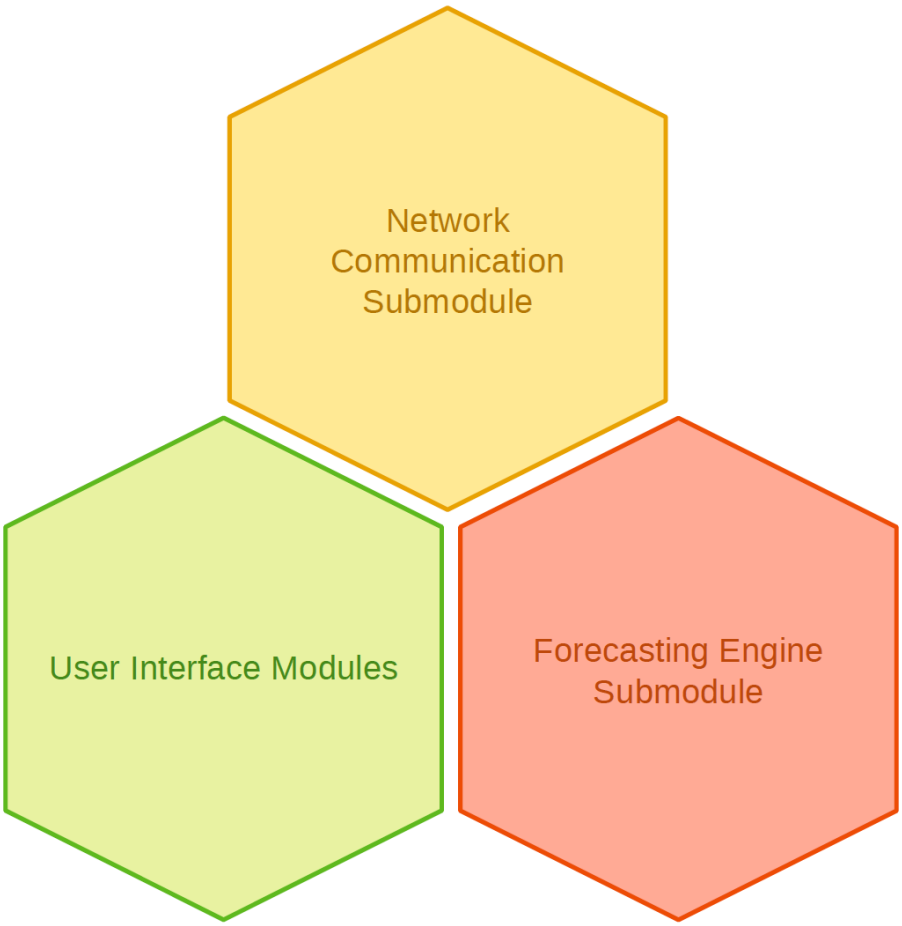


Фигура 3.12: Настройки на компонента за гласуване

Възможностите за съхраняване на информация в паметта на мобилното устройство, чрез SQLite, е функционалност предоставена от самата операционна система Android. Поради тази причина, в модула за Android е добавен и помощен, междинен компонент за работа с релационната база данни. Терминът за такъв вид компоненти е Database Helper.

3.2.2 Компоненти в модула за изчисление

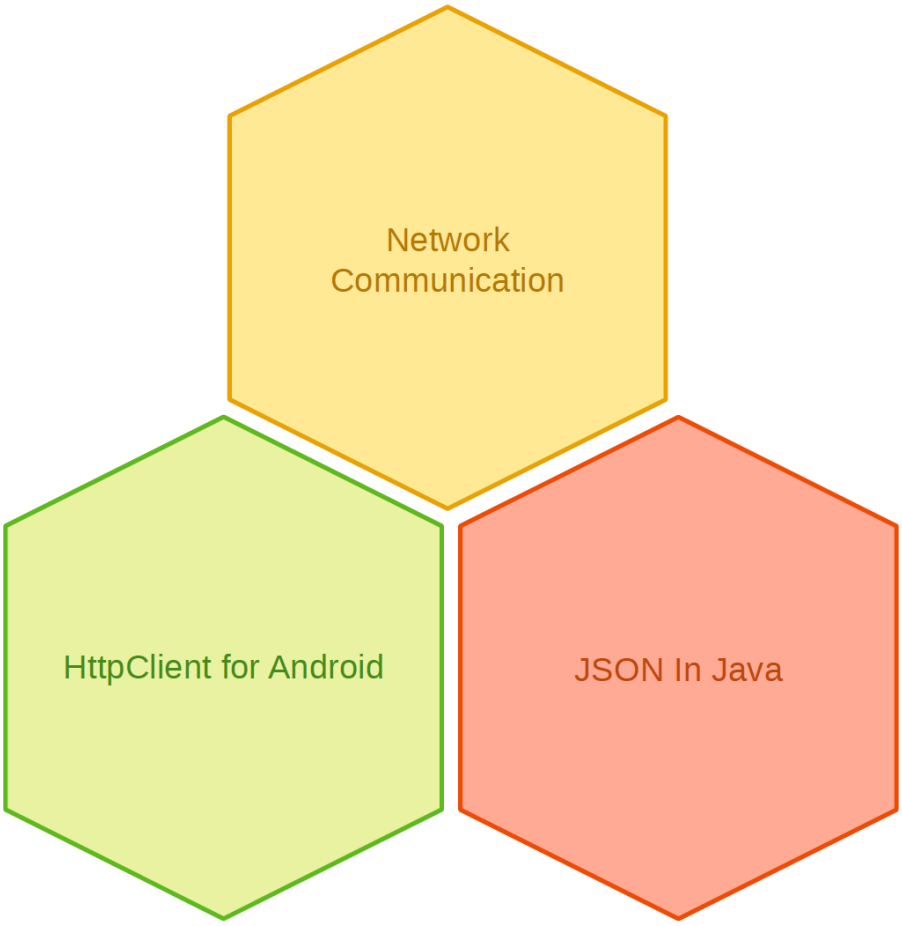
Изчислителният модул е организиран в два компонента – компонент за мрежова комуникация и компонент за обучение/прогнозиране (Фиг. 3.13). Мрежовата комуникация е изнесена извън потребителския интерфейс, защото комуникацията с отдалечения сървър е необходима, без значение дали приложението работи в Android OS средата или в терминалния прозорец на настолна компютърна конфигурация.



Фигура 3.13: Структура на изчислителния модул

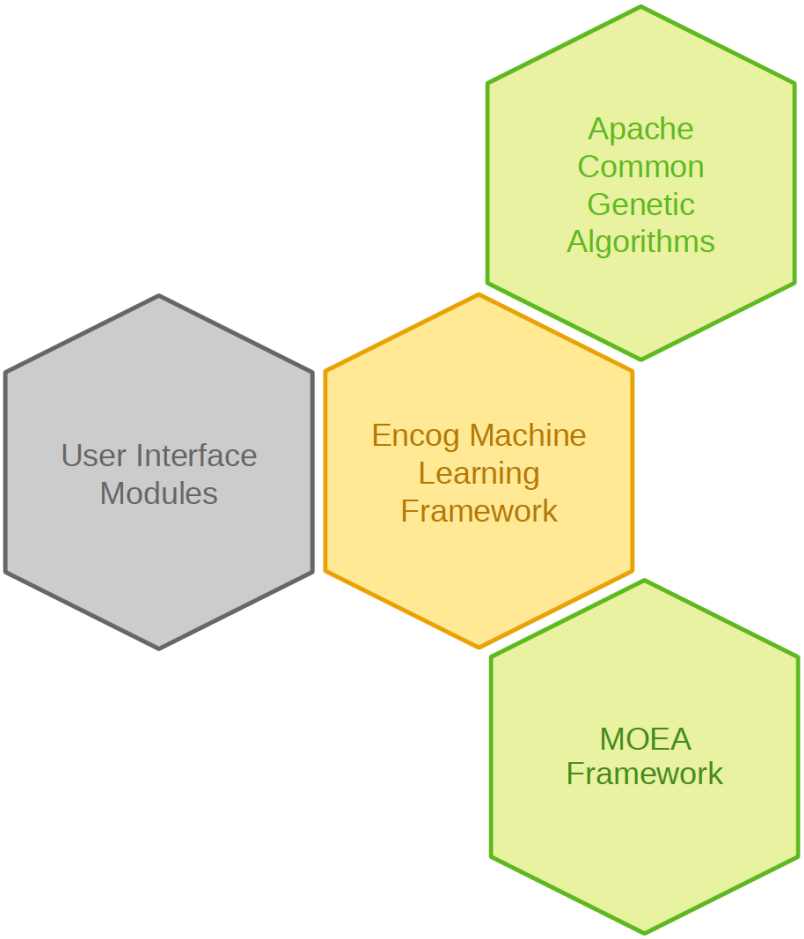
Комуникацията с отдалечения сървър е организирана под формата на отделен Java клас, който изпълнява функцията на HTTP Helper (Фиг. 3.14). Обменът на данните по HTTP протоколът е възможен, благодарение на външна програмна библиотека (HttpClient for Android). Тъй като обменяните съобщения с отдалечения

сървър са организирани под формата на JSON пакети, за обработката им се използва също външна библиотека (JSON In Java).



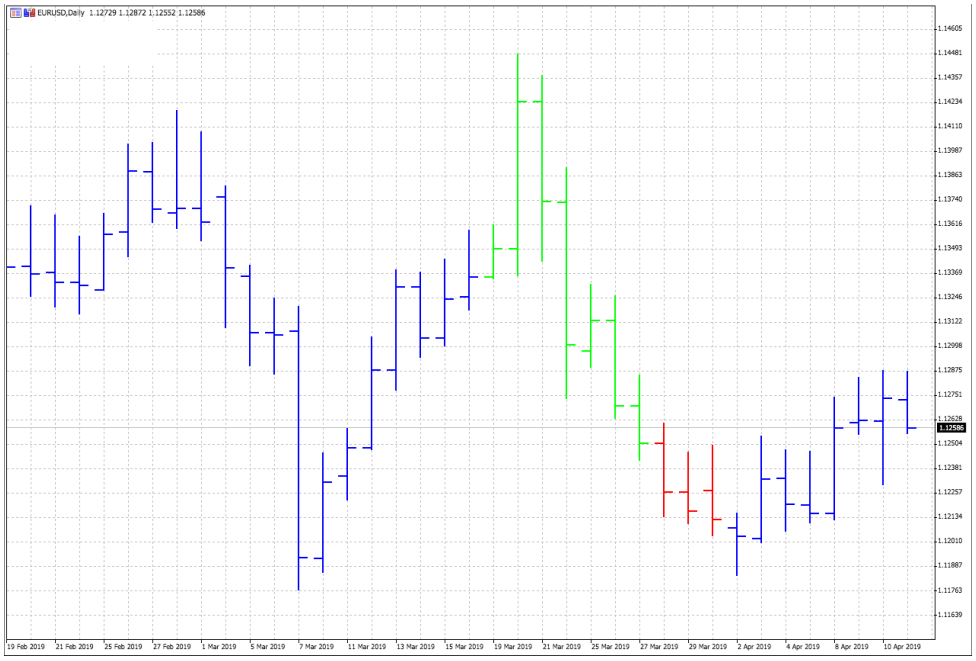
Фигура 3.14: Организация на мрежовата комуникация

Конструирането на изкуствената невронна мрежа, организацията на данните на входа и на изхода ѝ, както и евристичните алгоритми за обучение, са осъществени с помощта на външни програмни библиотеки.



Фигура 3.15: Организация на обучението/прогнозирането

Изкуствената невронна мрежа се изгражда с помощта на библиотеката Encog Machine Learning Framework. В самата библиотека е реализирано обучение с обратно разпространение на грешката. Допълнително две библиотеки реализират глобална евристична оптимизация – Apache Common Genetic Algorithms и MOEA Framework.



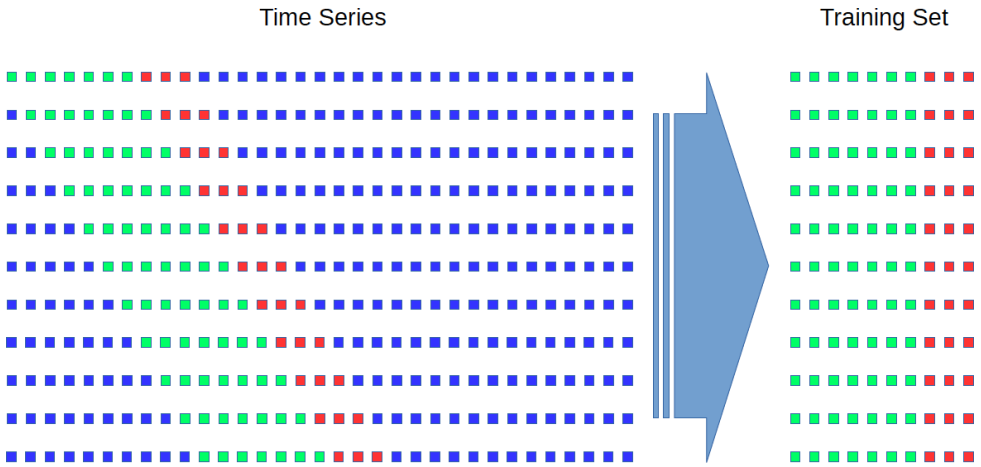
Фигура 3.16: Съотношение между цената EUR/USD

Преди да постъпят в системата, входните данни от финансовия времеви ред имат определени измерени стойности (Фиг. 3.16). Тези стойности варират в различни диапазони за различните финансови инструменти. Тази разлика е една от причините времевия ред да бъде преоразмерен (Листинг 3.1).

Листинг 3.1: Преоразмеряване на входните данни

```
double input[][][] = new double[values.length -
                (inputSize + outputSize)][inputSize];
double target[][][] = new double[values.length -
                (inputSize + outputSize)][outputSize];
for (int i = 0; i < values.length - (inputSize + outputSize); i++) {
        for (int j = 0; j < inputSize; j++) {
                input[i][j] = range[0] + (range[1] - range[0]) *
                                        (values[i + j] - min) / (max - min);
        }
        for (int j = 0; j < outputSize; j++) {
                target[i][j] = range[0] + (range[1] - range[0]) *
                                        (values[i + inputSize + j] - min) / (max - min);
        }
}
examples = new BasicMLDataSet(input, target);
```

След преоразмеряването на входните данни, те биват оформени на групи тренировъчни и тестови примери. Един пример се състои от входни данни и очаквани изходни резултати (Фиг. 3.17).



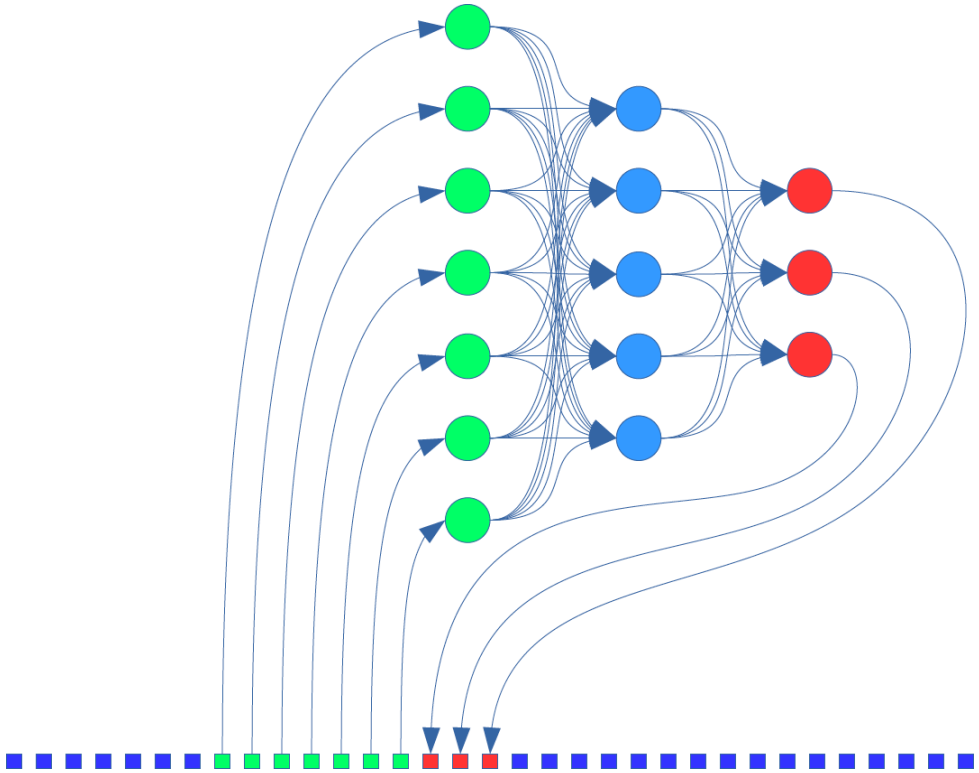
Фигура 3.17: Оформяне на тренировъчни и тестови примери

Типът на изкуствената невронна мрежа е трислоен перцептрон (Фиг. 3.18), а параметрите на самата топология се зареждат от отдалечения сървър. Топологиите на различните мрежи се задават от оператор на сървъра. Като функция за активация се използва хиперболичен тангенс, тъй като той е симетричен спрямо абцисната ос (Листинг 3.2).

Листинг 3.2: Конструиране на изкуствена невронна мрежа

```
Map<NeuronType, Integer> counters = new HashMap<NeuronType, Integer>();
counters.put(NeuronType.REGULAR, 0);
counters.put(NeuronType.BIAS, 0);
counters.put(NeuronType.INPUT, 0);
counters.put(NeuronType.OUTPUT, 0);

for (int type : InputData.NEURONS) {
        counters.put(NeuronType.valueOf(type),
                        counters.get(NeuronType.valueOf(type)) + 1);
}

int inputSize = counters.get(NeuronType.INPUT);
int hiddenSize = counters.get(NeuronType.REGULAR);
int outputSize = counters.get(NeuronType.OUTPUT);

network.addLayer(new BasicLayer(null,
                true, inputSize));
network.addLayer(new BasicLayer(new ActivationTANH(),
                true, hiddenSize));
network.addLayer(new BasicLayer(new ActivationTANH(),
                false, outputSize));
network.getStructure().finalizeStructure();
network.reset();
```

Обучението на изкуствената невронна мрежа се извърша чрез превключване между точни числени и евристични алгоритми на случаен принцип.



Фигура 3.18: Трислоен перцептрон

Точните числени алгоритми са реализирани чрез библиотека Encog Machine Learning Framework (Листинг 3.3).

Листинг 3.3: Обучение с устойчиво разпространение на грешката

```
train = new ResilientPropagation(network, examples);
train.iteration();
train.finishTraining();
```

Обучението с алгоритъма за еволюция на разликите и генетичните алгоритми е организирано под формата на полиморфизъм в езика Java (Листинг 3.4).

Листинг 3.4: Обучение с еволюционни евристични алгоритми

```
List<Double> weights = new ArrayList<Double>();
for (int layer = 0; layer < network.getLayerCount() - 1; layer++) {
        int bias = network.isLayerBiased(layer) ? 1 : 0;
        for (int from = 0; from < network.getLayerNeuronCount(layer) + bias; from++) {
                for (int to = 0; to < network.getLayerNeuronCount(layer + 1); to++) {
                        weights.add(network.getWeight(layer, from, to));
                }
        }
}

Optimizer optimizer = new MoeaOptimizer(optimizationTimeout, network, train, populationSize,
    crossoverRate, scalingFactor);
```

```
weights = optimizer.optimize(weights);

for (int layer = 0, index = 0; layer < network.getLayerCount() - 1; layer++) {
        int bias = network.isLayerBiased(layer) ? 1 : 0;
        for (int from = 0; from < network.getLayerNeuronCount(layer) + bias; from++) {
                for (int to = 0; to < network.getLayerNeuronCount(layer + 1); to++, index++) {
                        network.setWeight(layer, from, to, weights.get(index));
                }
        }
}
```

Еволюционните алгоритми в своята популация съдържат комплекти от тегла на изкуствената невронна мрежа. Комплектите тегла се зареждат последователно в структурата на изкуствената невронна мрежа и се оценява общата грешка, която мрежата допуска. Тази обща грешка на мрежата е жизнената стойност на всеки комплект тегла в популацията. Колкото по-малка е общата грешка, толкова по-добре се представя съответния комплект тегла. Най-добрите намерени решения се изпращат за съхранение на отдалечения сървър.

3.3 Обобщение

Извършена е разработка на софтуерно решение за мобилни устройства, под операционната система Android OS. Разработеният програмен код е представен като подсистема от тип клиент-сървър. От своя страна, мобилното приложение е представено с модулна архитектура за максимална конфигорируемост. Направено е представяне и на разработения графичен потребителски интерфейс. Представени са външните програмни библиотеки, включени в процесите по пресмятането на прогнозите. Получените резултати са представени в [104, 105, 106, 127]. В публикация [104] авторът на настоящия дисертационен труд има 1/3 принос, който се състои в предлагането на представената идея и написването на програмния код за извършване на експериментите, поради което е и водещ автор в публикацията. В публикация [105] авторът на настоящия дисертационен труд има 1/3 принос, който се състои в предлагането на представената идея и написването на програмния код за извършване на експериментите, поради което е и водещ автор в публикацията. В публикация [106] авторът на настоящия дисертационен труд има 1/3 принос, който се състои в предлагането

на представената идея и написването на програмния код за извършване на експериментите, поради което е и водещ автор в публикацията. В публикация [127] авторът на настоящия дисертационен труд има 1/3 принос, който се състои в написването на програмния код за извършване на експериментите.

Глава 4

Числени тестове на алгоритмите в системата за прогнозиране

Програмният модул, използван за извършването на представените експерименти е изваден, като изолиран подмодул на софтуерната разработка в настоящия дисертационен труд. Кодът се тества самостоятелно, за да се отстранят ефектите от асинхронно изпълнение, във фонов режим, породен от организацията на изпълнението в операционната система Android OS. Работата на изолираните подмодули в цялостната реализация на софтуера не би позволила ефективно измерване на ангажираните изчислителни ресурси и използваното изчислително време. Също така, измерване на производителността върху мобилни устройства е значително по-ненадеждно, отколкото при настолни компютърни системи.

Прогнозирането в мобилното приложение от страната на клиента се извършва с трислоен перцептрон. Времевият ред се разбива на двойки минало-бъдеще, които съставляват обучаващите примери. Разделението на минал и бъдещ период е условно. Практически, цялата информация е от данни в миналото. С пълзящ прозорец, условното разделяне се получава за минали стойности и стойности в бъдещия момент, спрямо условния разделител. Разполагайки с тренировъчни примери, процесът по обучението на трислойния перцептрон е свързан с търсене на тегла, водещи до минимизирането на общата грешка, която мрежата допуска. В системата са реали-

зирани два начина за търсене на оптимални тегла – точни градиентни и евристични. Двата начина се активират на случаен принцип с равна вероятност (Листинг 4.1).

Листинг 4.1: Превключване на алгоритмите за обучение

```
/* Switch between gradient and evolutionry algorithm. */
if (PRNG.nextBoolean() == true) {
        /* Gradient-based optimization. */
} else {
        /* Evolutionary optimization. */
}
```

Ефективността на двата начина за оптимизиране на теглата се подлага на изследване, като алгоритмите се изолират да работят самостоятелно, извън общата работа на системата.

4.1 Точни числени алгоритми

Encog Machine Learning Framework поддържа множество алгоритми за обучение на изкуствени невронни мрежи, като от тях са избрани пет точни числени алгоритъма (Листинг 4.2).

Листинг 4.2: Набор от точни числени алгоритми

```
/* Selection of gradient-based training. */
Propagation[] propagations = {
        new Backpropagation((BasicNetwork) network.clone(), examples),
        new ResilientPropagation((BasicNetwork) network.clone(), examples),
        new QuickPropagation((BasicNetwork) network.clone(), examples),
        new ScaledConjugateGradient((BasicNetwork) network.clone(), examples),
        new ManhattanPropagation((BasicNetwork) network.clone(), examples, PRNG.nextDouble())
};
```

Класът Backpropagation реализира традиционният алгоритъм за обратно разпространение на грешката, който разчита на частни производни. При алгоритъма за обратно разпространение на грешката съществува проблем с магнитута на производните, когато те дават твърде големи или твърде малки стойности. Втори недостатък на алгоритъма за обратно разпространение на грешката е, че параметърът за научаване (learning rate) е една единствена стойност за цялата мрежа.

Класът ResilientPropagation реализира подобрение на алгоритъма за обратно разпространение на грешката, като въвежда коефициент за обновление (update value)

на всяка връзка между два неврона. Значително предимство е, че конкретните стойности за коефициента на обновление се определят автоматично и не е нужна човешка намеса. Това е водеща разлика от класическия алгоритъм за обратно разпространение на грешката, където параметъра за научаване е предварително дефиниран, и то ръчно.

Класът QuickPropagation внася подобрение на алгоритъма за обратно разпространение на грешката, като използва Нютон методите. Състои се в квадратична апроксимация на последващите стъпки в градиента. По този начин общо допуснатата от изкуствената невронна мрежа грешка се представя като парабола, чието дъно би било минимум за общата допусната грешка. Недостатък при тази модификация на алгоритъма за обратно разпространение на грешката е, че поведението на изкуствената невронна мрежа може да се окаже хаотично, при големите стъпки за обновление на теглата.

ScaledConjugateGradient класът реализира обучение на принципа на линейните посоки, като не прави всеки път линейно търсене. Ако бъде правено линейно търсене на всяка итерация, това би направило обучението твърде неефективно, по отношение на изчислително време. Алгоритъмът се прилага за изкуствени невронни мрежи, в които функциите имат дефинирани производни.

ManhattanPropagation класът се опитва да въведе подобрение в класическото обратно разпространение на грешката, като единствено използва знака на производната, но не и нейния магнитут. Корекцията на теглата се извършва с предварително зададена стойност. Тази предварително зададена стойност се определя експериментално. Манхатън обучението може да се смята за опростен вариант на еластиочното (Resilient) обучение.

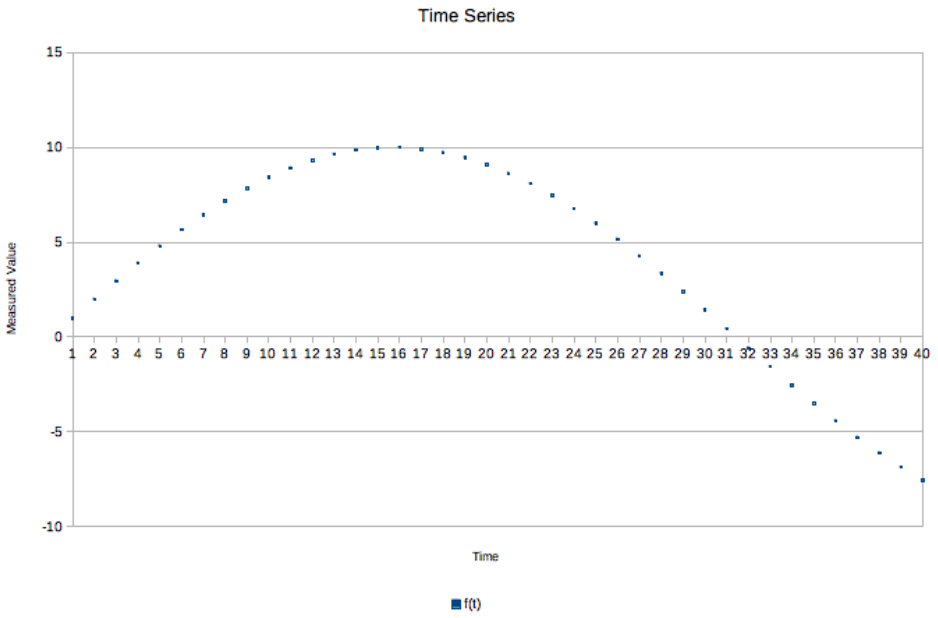
При равни други условия се стартира обучение с всеки един от алгоритмите, като се отчитат броя оптимизационни цикли и обща допусната грешка от изкуствената невронна мрежа (Листинг 4.3).

Листинг 4.3: Експериментална проверка на точните числени алгоритми

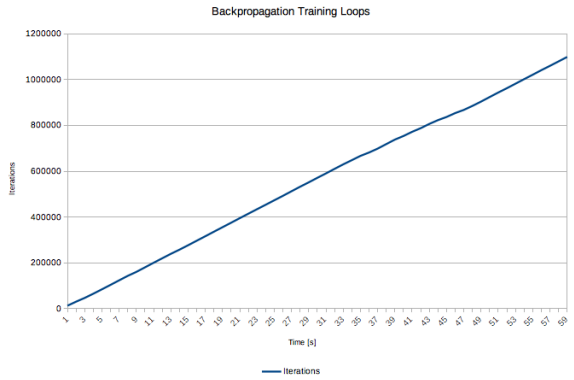
```
System.out.println("Experiment start.");
for (Propagation p : propagations) {
    System.out.println("" + p.getClass().getName());
    long loop = 0;
    long second = 0;
    long time = System.currentTimeMillis();
    long start = System.currentTimeMillis();
    while (p.isTrainingDone() == false && System.currentTimeMillis()-start < 1*60*1000) {
        loop++;
        p.iteration();
        if(System.currentTimeMillis()-time > 1000) {
            second++;
            System.out.println(second + "\t" + loop + "\t" + p.getError());
            time = System.currentTimeMillis();
        }
    }
    p.finishTraining();
}
System.out.println("Experiment end.");
```

Експериментите са извършени с два комплекта данни – базова форма на времеви ред, следваща синус функция (Фиг. 4.1) и цена на дигиталната валута биткойн в щатски долари (Фиг. 4.7).

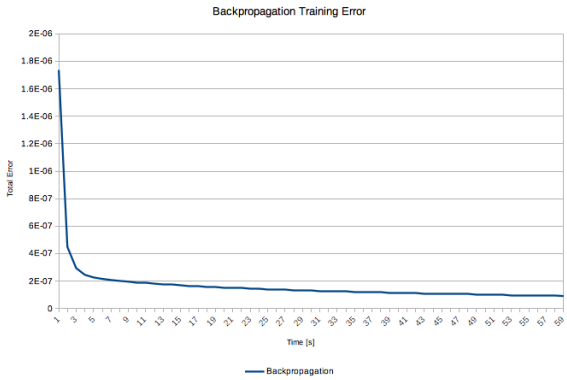
4.1.1 Синус функция



Фигура 4.1: Времеви ред следващ синус функция

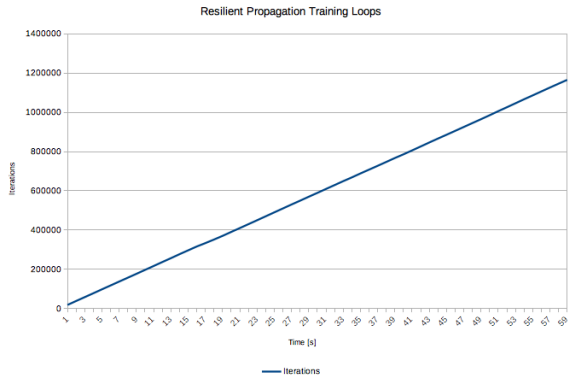


(а) Брой тренировъчни цикли

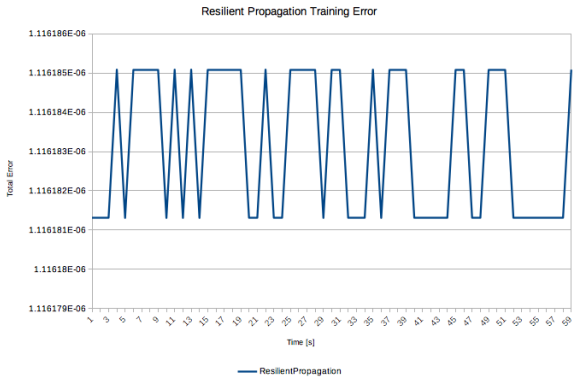


(б) Обща грешка допусната от ИНМ

Фигура 4.2: Backpropagation

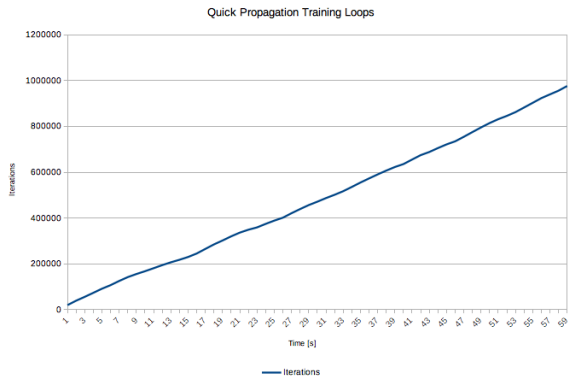


(а) Брой тренировъчни цикли

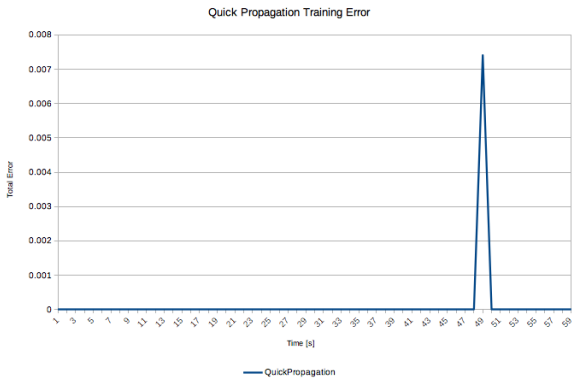


(б) Обща грешка допусната от ИНМ

Фигура 4.3: ResilientPropagation

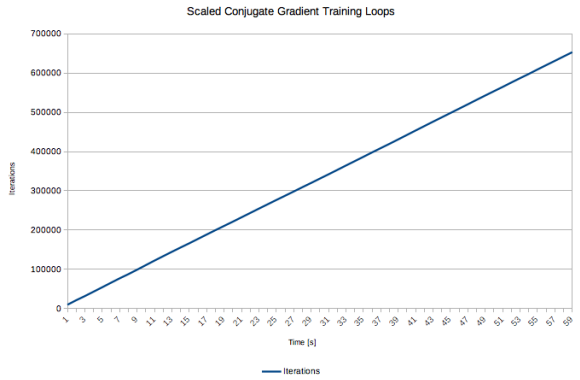


(а) Брой тренировъчни цикли

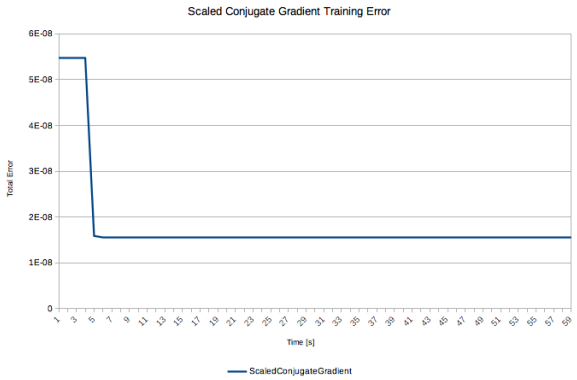


(б) Обща грешка допусната от ИНМ

Фигура 4.4: QuickPropagation

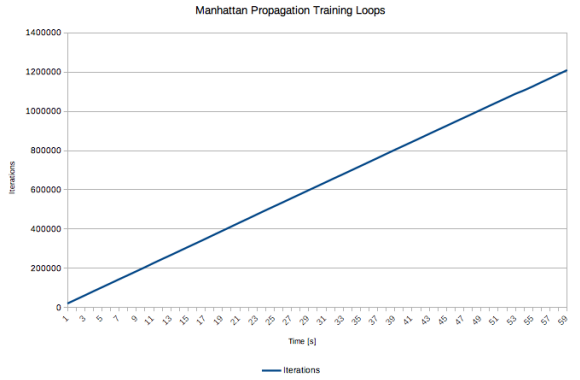


(а) Брой тренировъчни цикли

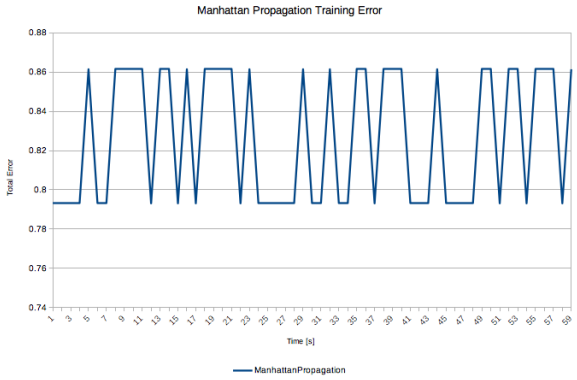


(б) Обща грешка допусната от ИНМ

Фигура 4.5: ScaledConjugateGradient



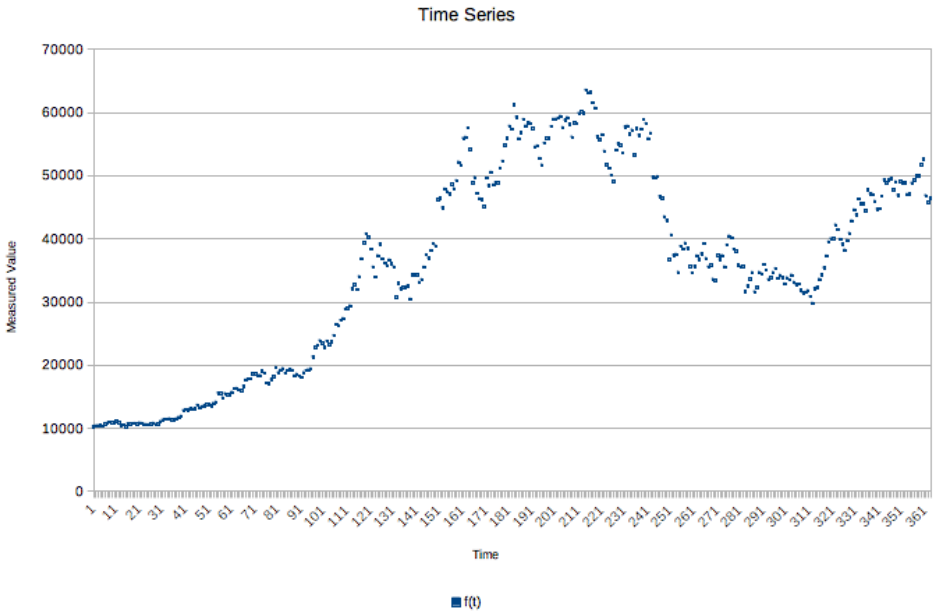
(а) Брой тренировъчни цикли



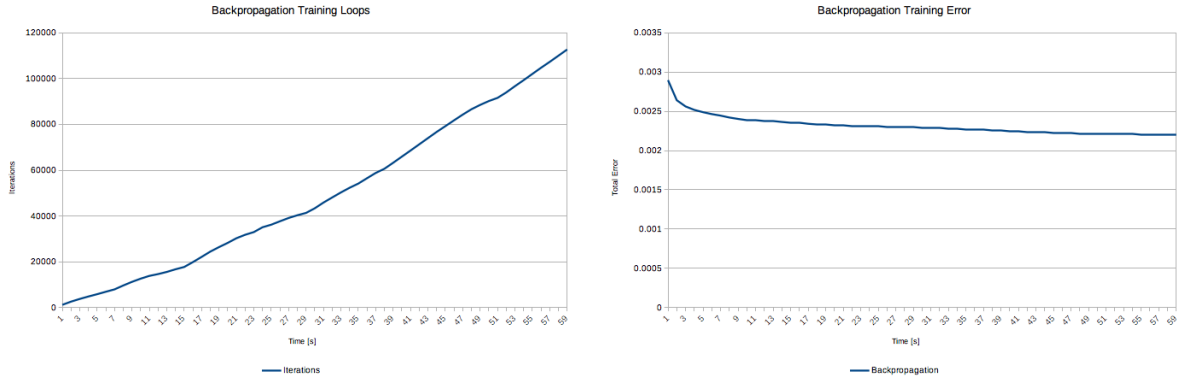
(б) Обща грешка допусната от ИНМ

Фигура 4.6: ManhattanPropagation

4.1.2 Цена на биткойн

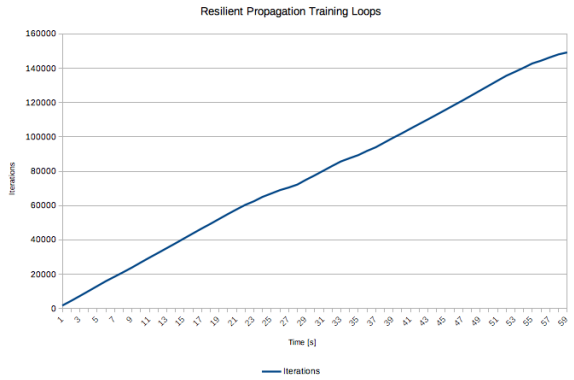


Фигура 4.7: Времеви ред с цената на биткойн

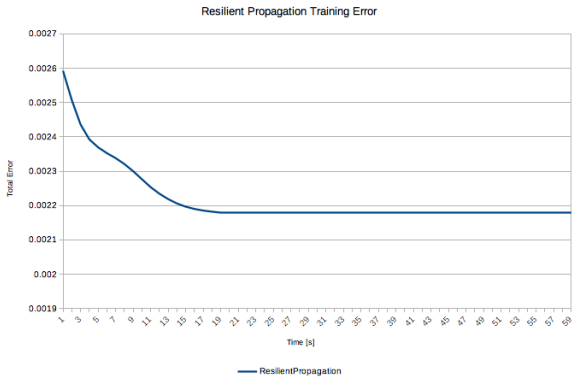


(а) Брой тренировъчни цикли (б) Обща грешка допусната от ИНМ

Фигура 4.8: Backpropagation

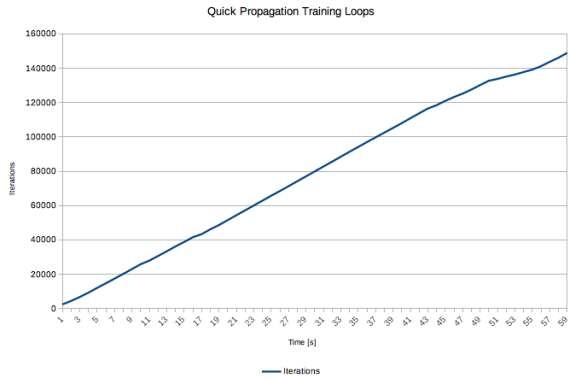


(а) Брой тренировъчни цикли

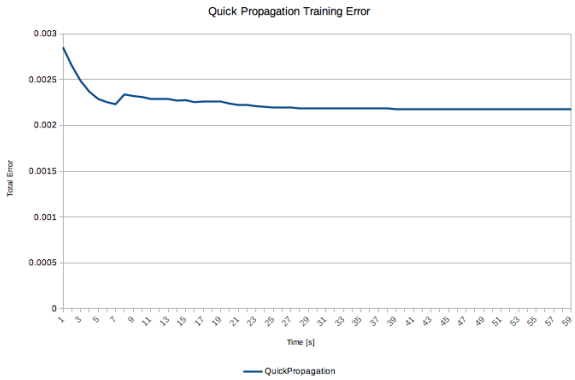


(б) Обща грешка допусната от ИНМ

Фигура 4.9: ResilientPropagation

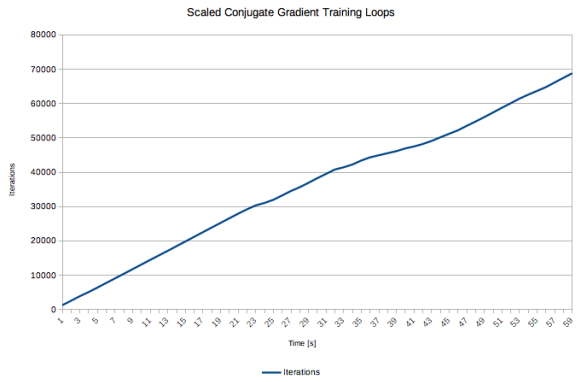


(а) Брой тренировъчни цикли

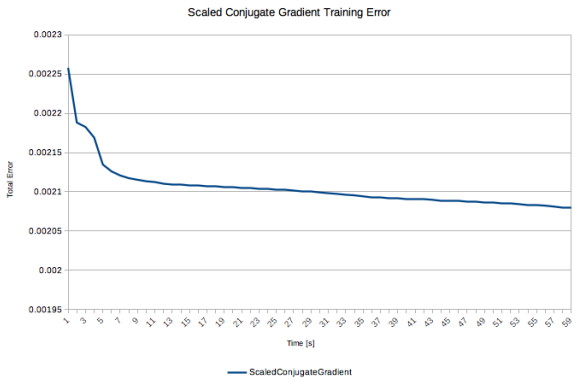


(б) Обща грешка допусната от ИНМ

Фигура 4.10: QuickPropagation

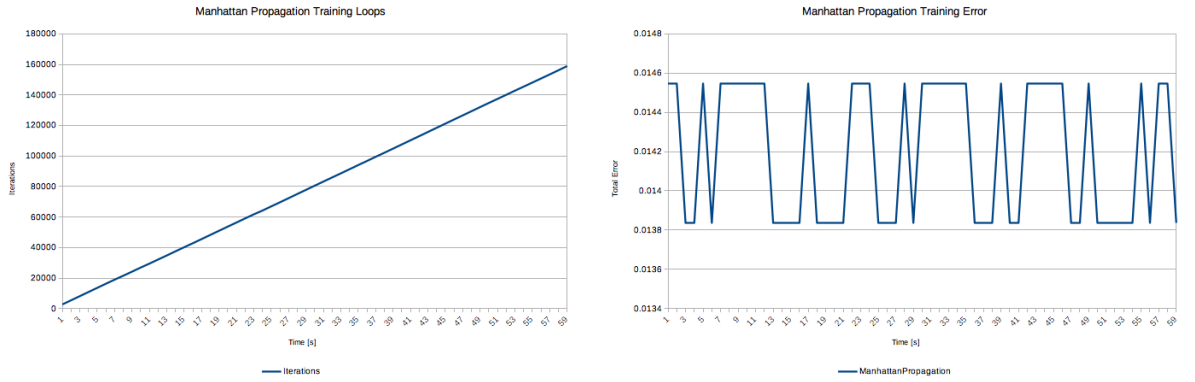


(а) Брой тренировъчни цикли



(б) Обща грешка допусната от ИНМ

Фигура 4.11: ScaledConjugateGradient



(а) Брой тренировъчни цикли (б) Обща грешка допусната от ИНМ

Фигура 4.12: ManhattanPropagation

4.1.3 Анализ на ефективността

Резултатите (Фиг. 4.2а-4.6б) от проведените експерименти показват, че при точните числени алгоритми класическият алгоритъмът с обратно разпространение на грешката дава най-добра ефективност, за времеви ред с относително опростена структура. При времеви ред със значително по-сложна структура, еластичното (Resilient) обучение е с най-добра ефективност (Фиг. 4.8а-4.12б). При точни числени алгоритми и времеви ред с проста и сложна структура, Манхатън обучението показва най-слаба ефективност. Допуснатата грешка при този вид обучение е поне десет пъти по-голяма в сравнение с най-добре представящите се алгоритми.

4.2 Еволюционни алгоритми

MOEA Framework поддържа множество евристични оптимизационни алгоритми, като основното предназначение на тази софтуерна библиотека е многокритериална оптимизация. В добавка към многокритериалната оптимизация, библиотеката поддържа и малка група еднокритериални алгоритми. От еднокритериалните алгоритми са избрани еволюционна стратегия (Evolution Strategy), генетичен алгоритъм (Genetic Algorithm) и еволюция на разликите (Differential Evolution). И трите алгоритъма организират търсенето на решение под формата на популация. Популацията от решения еволюира и във всяко следващо поколение очакването е да се появяват ре-

шения все по-близки до глобален оптимум (единственият или някой от оптимумите, ако са повече от един). Трите алгоритъма отново са изпълнени за двата комплекта данни - базова форма на времеви ред, следваща синус функция (Фиг. 4.1) и цена на дигиталната валута биткойн в щатски долари (Фиг. 4.7).

Използва се същата топология на изкуствена невронна мрежа, но вместо градиентен алгоритми, теглата на мрежата се ползват за инициализиране на начална популация на евритстичните алгоритми. Всеки индивид в популацията представлява вектор от реални числа, които точно съответстват на тегловните коефициенти между невроните. При оценката на отделните индивиди, теглата се зареждат в структурата на изкуствената невронна мрежа и се проверява общата грешка, която се допуска при прогнозиране.

Листинг 4.4: Евристични алгоритми

```
AbstractAlgorithm[] algorithms = {
        new EvolutionStrategy(problem, comparator, initialization,
                new SelfAdaptiveNormalVariation()),
        new GeneticAlgorithm(problem, comparator, initialization,
                selections[PRNG.nextInt(selections.length)],
                new GAVariation(new UniformCrossover(crossoverRate), new Insertion(mutationRate))),
        new DifferentialEvolution(problem, comparator, initialization,
                new DifferentialEvolutionSelection(),
                new DifferentialEvolutionVariation(crossoverRate, scalingFactor))
};
```

Трите евристични алгоритъма се изпълняват с параметри по подразбиране (Листинг 4.4). При реален работен режим, един от трите алгоритъма се избира на случаен принцип. Целта е да се даде разнообразие на различните начини за търсене на по-добро решение.

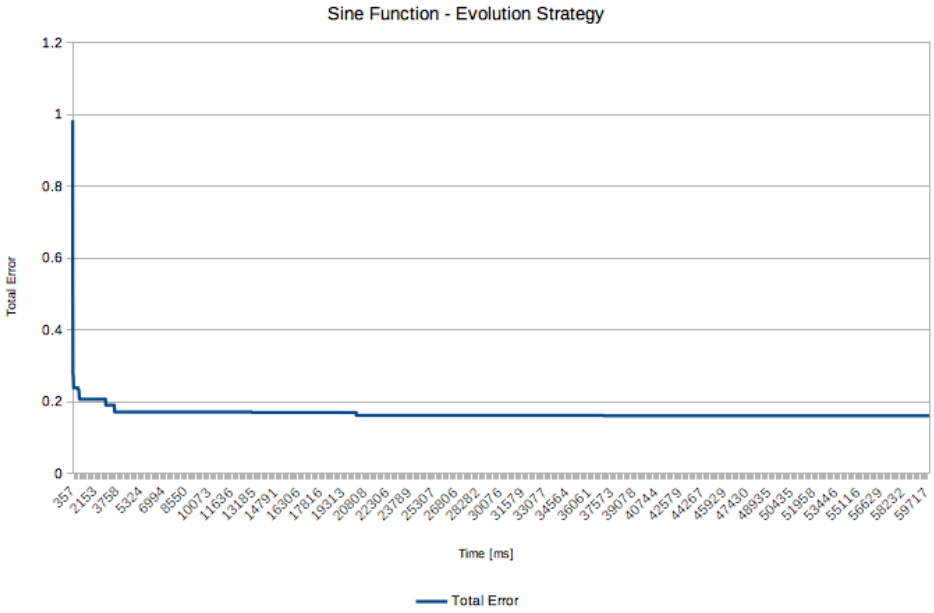
Листинг 4.5: Отчитане на междинните стойности в процеса по оптимизация

```
System.out.println(InputData.SYMBOL);
System.out.println(algorithm.getClass().getName());

long stop = System.currentTimeMillis() + optimizationTimeout;
while (System.currentTimeMillis() < stop) {
        algorithm.step();
        System.out.print(System.currentTimeMillis());
        System.out.print("\t");
        System.out.print(algorithm.getResult().get(0).getObjective(0));
        System.out.println();
}
```

Всеки от алгоритмите се стартира за една минута, като при всеки оптимизационен

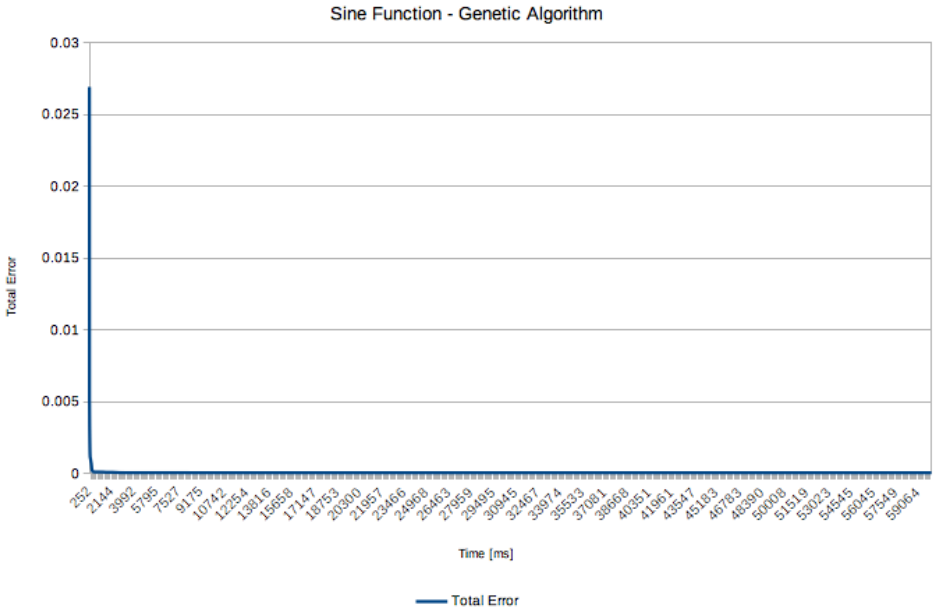
цикъл се снема най-добрата постигната стойност за жизненост на индивида (Листинг 4.5).

4.2.1 Синус функция



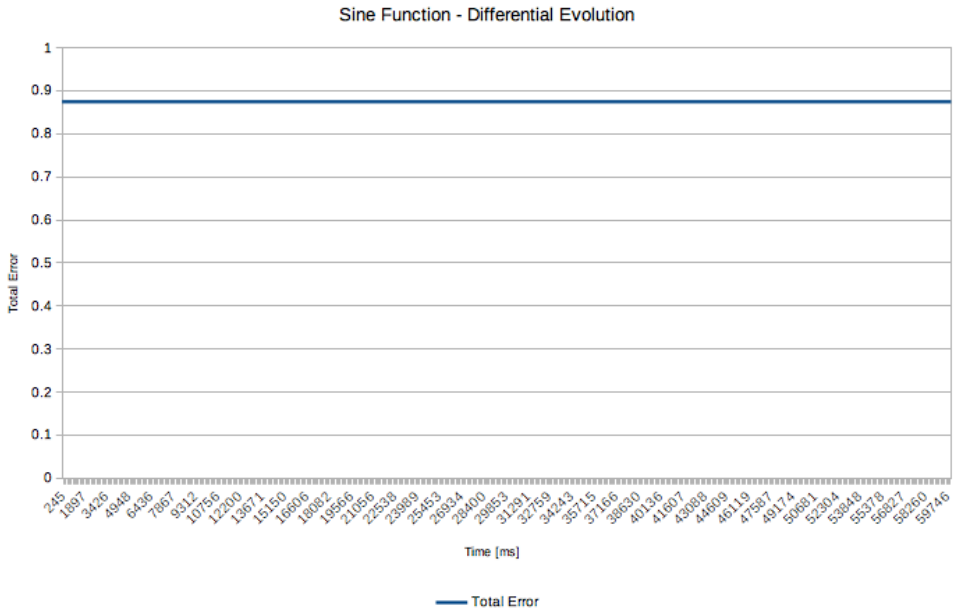
Фигура 4.13: Еволюционна стратегия върху синус функция

Синус функцията дава относително опростен времеви ред и резултатите показват сходимост на процеса по обучението на изкуствената невронна мрежа с алгоритъма за еволюционна стратегия (Фиг. 4.13).



Фигура 4.14: Генетичен алгоритъм върху синус функция

Генетичният алгоритъм също дава сходимост при първите итерации от обучението на изкуствената невронна мрежа (Фиг. 4.14).

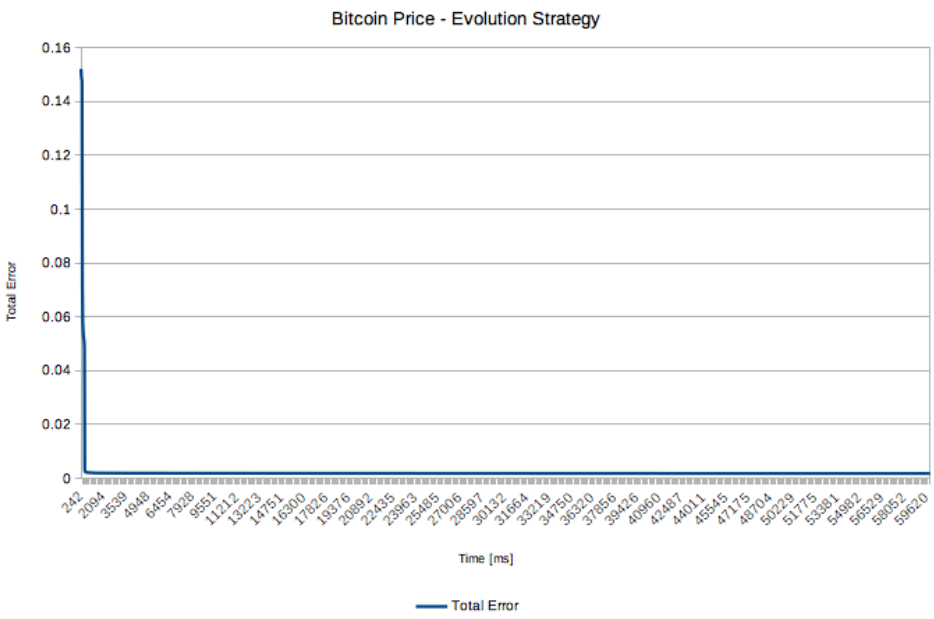


Фигура 4.15: Еволюция на разликите върху синус функция

При еволюцията на разликите не се забелязва подобрение в резултатите от оптимизационния процес (Фиг. 4.15). Това се дължи на факта, че алгоритъмът не успява

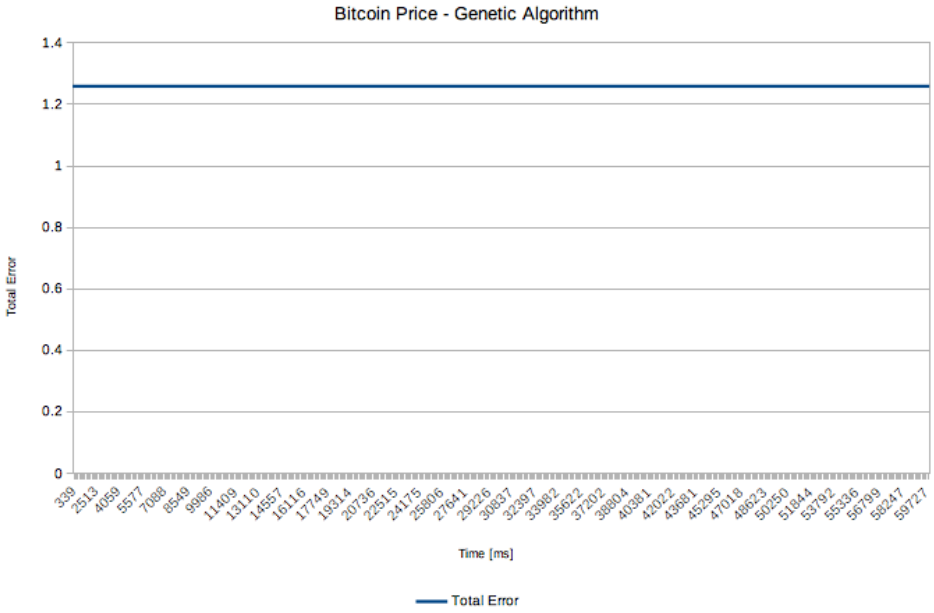
да излезе от локалния оптимум, в който попада още след първата оптимизационна итерация.

4.2.2 Цена на биткойн



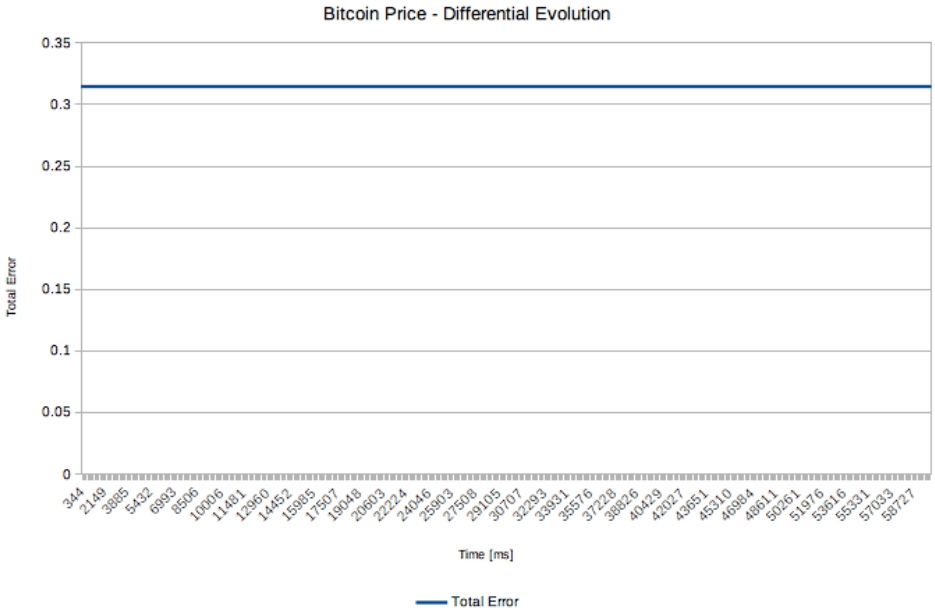
Фигура 4.16: Еволюционна стратегия върху цена на биткойн

Времевият ред с цената на дигиталната валута биткойн, отразена в щатски долари, е значително по-сложен като структура. От една страна, има много повече отчетени стойности, но също така формата е доста по-сложна. Въпреки това, еволюционната стратегия показва ясна сходимост (Фиг. 4.16).



Фигура 4.17: Генетичен алгоритъм върху цена на биткойн

При по-сложен времеви ред, генетичният алгоритъм изпада в локален опитимум още след първата итерация (Фиг. 4.17), по аналогия с еволюцията на разликите в по-опростения времеви ред.



Фигура 4.18: Еволюция на разликите върху цена на биткойн

Еволюцията на разликите показва същото поведение, което бе отчетено и при

по-опростения времеви ред (Фиг. 4.18).

4.2.3 Анализ на ефективността

От резултатите след проведените експерименти, ясно се вижда, че алгоритъмът за еволюционна стратегия работи ефективно, както при по-опростени времеви редове, така и при по-сложни времеви редове. При по-опростени времеви редове, генетичните алгоритми показват сходимост, но при по-сложни времеви редове са недостатъчно ефективни. Еволюцията на разликите не показва признаци за сходимост, както при по-опростени времеви редове, така и при по-сложни. Определена ефективност на генетичните алгоритми и еволюцията на разликите може да се очаква при по-продължително време за работа и в ситуации, когато вече има попадане в локален опитимум.

При проведените експерименти, параметрите на различните евристични алгоритми са избрани със стойности по подразбиране, но в реална работна среда, параметрите се избират с помощта на случайно търсене. Възможно е да се доразработи адаптивна стратегия за подбор на най-ефективните стойности за параметрите на евристичните алгоритми, но това изследване попада извън обхвата на настоящия труд и е въпрос на бъдещи изследвания.

4.3 Обобщение

Изпълнен е сравнителен анализ на подбраните точни числени и евристични алгоритми за обучение на изкуствени невронни мрежи. При точните числени алгоритми ясно се откроява обучението с обратно разпространение на грешката. Получените резултати са представени в [111]. При евристичните алгоритми добри резултати показва обучението с еволюционна стратегия.

Листинг 4.6: Случаен избор на точните числени алгоритми

```
propagation = propagations[PRNG.nextInt(propagations.length)];
```

Макар и някои от използваните алгоритми за обучение да не дават твърде голяма

ефективност, те могат да дадат допълнително разнообразие в генетичния фонд на еволюционните алгоритми, които са в хибридна употреба с точните числени алгоритми. Всеки от петте точни числени алгоритъма бива избиран на случаен принцип, равно вероятно, за участие в обучението (Листинг 4.2 и 4.6). Обучението на изкуствената невронна мрежа не започва от случайно подбрани тегла, а продължава от там, до където е стигнато при предишна обучаваща сесия.

Заключение - резюме на получените резултати

В почти всички области на човешката дейност се наблюдават времеви редове. Времевите редове намират различни приложения в икономическото прогнозиране, прогнозирането на продажби, бюджетен анализ, процеса на контрол по качеството, анализ на фондовите пазари и др. В тази връзка са разработени много начини за прогнозиране на финансови времеви редове, но като един от най-обещаващите, се откроява прогнозирането с изкуствени невронни мрежи. Характерно за изкуствените невронни мрежи е, че те са много ефективен инструмент, след като веднъж са обучени. Процесът на обучение, от своя страна, често отнема твърде дълго време и се нуждае от голямо количество изчислителни ресурси. Това налага търсеното на нови подходи, свързани с намаляване на времето за обучение на изкуствените невронни мрежи. Поради това, разработването на хибридни алгоритми, целящи ускоряване на обучението при изкуствени невронни мрежи от тип многослоен перцептрон, за целите на прогнозирането на времеви редове, е актуално научноизследователско направление.

Предложени са хибридни алгоритми, които комбинират точни числени алгоритми с евристични алгоритми за оптимизация на теглата в изкуствени невронни мрежи. Възможността за паралелна обработка при еволюционните алгоритми, от своя страна позволява обучението на изкуствените невронни мрежи да се организира в разпределена среда от мобилни устройства.

В процеса на обучение на ИНМ, при определяне на теглата е предложено да се

използва генетичен алгоритъм. В него се прилагат операциите – селекция, кръстосване и мутация. Предложен и анализиран е нов оператор за селекция, основан на създаването на рекурсивни поколения при процедура за рекурсивно спускане. На всяко ниво на рекурсията всички индивиди от популацията се чифтосват помежду си. Това чифтосване се прави по алгоритъма на грубата сила. Този подход добре се комбинира с локално търсене. Кръстосването се прилага, докато има излъчен по-добър индивид в поколението. Така описания хибриден подход, съчетаващ алгоритмите локално търсене и грубата сила води до по-добри резултати за сметка на по-дългото време за изчисление. Разгледани са и възможностите за апроксимиране на криви към множество точки. За инкрементална апроксимация на времеви редове се използват уравнение на права и ред от синус функции. Предложен е подход за пресмятане на коефициентите на синус функциите с оптимизатор, базиран на еволюция на разликите и рояк от частици. Използваните алгоритми се прилагат на хибриден принцип. Построената крива се използва за генериране на прогноза, извън диапазона на известните измерени точки.

Предложена е алтернатива на производна за активационната функция в изкуствени невронни мрежи. Направено е сравнение между предложената алтернативна функция и първа производна на периодична затихваща активационна функция. Алтернативната функция дава по-добри резултати по отношение на бързодействие и допусната грешка. Предложен е и алгоритъм за обучение на изкуствени невронни мрежи от тип многослоен перцептрон в разпределена среда. Като средство за оптимизация на бавно изчислими целеви функции е предложен генетичен алгоритъм. Той лесно се подлага на паралелна обработка. За един и същи период от време могат да бъдат получени множество решения на бавно изчислимите целеви функции.

Направена е програмна реализация за хибридно използване на градиентни числени и евристични алгоритми за оптимизация на теглата в изкуствени невронни мрежи. Реализираната система е базирана на мобилни разпределени изчисления за прогнозиране на финансови времеви редове. Разработено е мобилно приложение за Android операционна система. То дава възможност на потребителя да даде своя вот, затова как очаква да се промени курса за конкретна валутна двойка, за визуализация на

процеса по обучение и представя прогнозната стойност от обучената на устройството невронна мрежа.

Направен е сравнителен анализ на точни числени и евристични алгоритми. Като входни данни са използвани базова форма на времеви ред, следващ синус функция и сложна форма на времеви ред от цената на дигиталната валута биткойн. От проведените експерименти и получените резултати е установено, че при точните числени алгоритми с проста структура на времеви ред, обучението с обратно разпространение на грешката има най-добра ефективност. При сложна входна структура най-добра ефективност дава еластичното (resilient) обучение. При евристичните алгоритми с простра форма на времеви ред най-добра ефективност дава генетичният алгоритъм. При сложна структура добра ефективност показва алгоритъма еволюционна стратегия.

Получените резултати, описани в настоящия дисертационен труд, могат да се обобщят в следните научни и научно-приложни приноси/резултати:

1. Направен е обзорен анализ и класификация на алгоритмите за обучение на изкуствени невронни мрежи от тип многослоен перцептрон. Разгледани са най-използваните начини за прогнозиране на времеви редове и по какъв начин машинното самообучение се прилага в тази проблемна област. Установено е, че алгоритъмът с обратно разпространение на грешката, който е от групата на точните числени, градиентни алгоритми, е най-често използвания за обучение на изкуствени невронни мрежи. Установено е също така, че този алгоритъмът много добре се допълва с евристичните, еволюционни алгоритми за глобална оптимизация, които от своя страна се поддават на изключително висока степен за паралелна обработка;

2. За целите на процеса на обучение на ИНМ, при определяне на теглата, е предложено да се използва генетичен алгоритъм, който разчита на операциите като селекция, кръстосване и мутация. Поради това е предложен нов оператор за селекция, основан на създаването на поколения при процедура за рекурсивно спускане, което осигурява търсеното бързодействие на използваните евристич-

ни алгоритми;

3. Инкрементaлната апроксимация на времевите редове най-често се реализира от уравнение на права и ред от синус функции. За постигане на по-добро апроксимиране, е предложен е подход за пресмятане на коефициентите на синус функциите с оптимизатор, базиран на еволюция на разликите и рояк от частици;

4. Предложена е алтернатива на производна за активационната функция в изкуствени невронни мрежи. Получените резултати показват по-добро бързодействие и допусната грешка в полза на предложената алтернативна функция отколкото при използването на първа производна на периодична затихваща активационна функция;

5. Предложен е генетичен алгоритъм за обучение на изкуствени невронни мрежи от тип многослоен перцептрон в разпределена среда, което прави възможно използването му при паралелна обработка;

6. Предложена е софтуерна архитектура, позволяваща реализиране на мобилни разпределени изчисления, базираща се на предложените хибридни алгоритми. Програмната реализация за хибридно използване на градиентни числени и евристични алгоритми за оптимизация на теглата в изкуствени невронни мрежи е реализирана в мобилно приложение за Android.

Насоки за бъдещи изследвания

Постигнатото в операционната система Android би могло да бъде сходна цел за постигане в операционната система iOS, на компанията Apple. За разлика от Android, iOS е изцяло затворена операционна система. Моделът за разпространение на приложения под iOS също би довел до допълнителни затруднения. Друга насока за развитие на идеите от дисертационния труд е ориентирана към операционната система KaiOS. Тази операционна система все още няма голямо разпространение и е

насочена предимно към по-маломощни мобилни устройства, но може да се окаже изключително продуктивна посока за допълнителни научни изследвания.

По отношение на използваните алгоритми и софтуерни библиотеки, съ-ществуват множество възможности за подобряване на наличните алгоритми и добавяне на нови такива. Използваните софтуерни библиотеки са с отворен код, което дава огромна свобода за изследване на програмния код, неговото дописване и оптимизиране, както и разширяването му с нови алгоритми.

Някои от областите с потенциално приложение на изкуствените невронни мрежи са управление на автономни системи, прогнозиране на работното натоварване на служители, прогнозиране на присъствието на служителите в офиса, при извършване на преброяване на населението и др. Към някои от тези потенциални приложения са насочени и част от бъдещите изследвания.

Публикации по темата на дисертационния труд

1. Mateeva, G., Tomov, P., Parvanov, D., Petrov, P., Kostadinov, G., Balabanov, T.: Some Capabilities of Android OS for Distributed Computing. Proceedings of Big Data, Knowledge and Control Systems Engineering BdKCSE'21, 2021, 1-6, ISBN 978-1-6654-1043-4.

2. Tomov, P.: Encog Gradient Training Algorithms Evaluation. Problems of Engineering Cybernetics and Robotics, vol. 77, 2021, 11-19, ISSN 2738-7356.

3. Tomov, P.: Multilayer Perceptron Fast Prototyping with Differential Evolution and Particle Swarm Optimization in LibreOffice Calc. Problems of Engineering Cybernetics and Robotics, vol. 75, 2021, 5-14, ISSN 2738-7356.

4. Tomov, P., Zankinski, I., Balabanov, T.: Training of Artificial Neural Networks for Financial Time Series Forecasting in Android Service and Widgets. Problems of Engineering Cybernetics and Robotics, no. 71, Institute of Information and Communication Technologies - Bulgarian Academy of Sciences, 2019, 50-56, ISSN 1314-409X.

5. Tomov, P., Zankinski, I., Balabanov, T.: Server Side Vote Clustering in Human-Computer Distributed Computing. Information Technologies and Control, no. 2, John Atanasoff Society of Automatics and Informatics, 2019, 15-19, ISSN 2367-5357.

6. Tomov, P., Zankinski, I., Barova, M.: Artificial Neural Networks Time Series Forecasting with Android Live Wallpaper Technology. Proceedings of the International Conference Numerical Methods for Scientific Computations and Advanced Applications NMSCAA'18, May 28-31, 2018, Hissarya, Fastumprint, 2018, 76-79, ISBN 978-954-91700-7-8.

7. Tomov, P., Zankinski, I., Barova, M.: Mobile Alternative of the Moneybee Project For Financial Forecasting. Proceedings of the Annual University Scientific Conference of the National Military University Vasil Levski, June 14-15, 2018, Veliko Tarnovo, Innovetion and Sustainability Academy – ISA, 2018, 1085-1089, ISSN 2367-7481.

8. Zankinski, I., Barova, M., Tomov, P.: Hybrid Approach Based on Combination of Backpropagation and Evolutionary Algorithms for Artificial Neural Networks Training by Using Mobile Devices in Distributed Computing Environment. Proceedings of 11th International Conference on Large-Scale Scientific Computations LSSC'17, June 5-9, 2017, Sozopol, Bulgaria, 2017, 425-434, ISBN 978-3-319-73440-8.

9. Tomov, P., Monov, V.: Modeling and Analysis of Time Series. Proceedings of International Scientific Conference UniTech'17, Gabrovo, November 17-18, 2017, vol. II, University publishing house Vasil Aprilov, 2017, 404-409, ISSN 1313-230X.

10. Tomov, P., Monov, V.: Artificial Neural Networks and Differential Evolution Used for Time Series Forecasting in Distributed Environment. Proceedings of the International Conference Automatics and informatics, October 4-5, 2016, Sofia, Bulgaria, Federation of the scientific engineering unions, John Atanasoff Society of Automatics and Informatics, 2016, 129-132, ISSN 1313-1850.

11. Keremedchiev, D., Barova, M., Tomov, P.: Mobile Application as Distributed Computing System for Artificial Neural Networks Training Used in Perfect Information Games, International Scientific Conference UniTech'16, Gabrovo, University publishing house Vasil Aprilov, 2016, 389-393, ISSN 1313-230X.

Забелязани цитирания

- Tomov, P., Zankinski, I., Balabanov, T.: Training of Artificial Neural Networks for Financial Time Series Forecasting in Android Service and Widgets. Problems of Engineering Cybernetics and Robotics, no. 71, Institute of Information and Communication Technologies - Bulgarian Academy of Sciences, 50–56, 2019. ISSN 1314-409X

    1. Borissova, D., Dimitrova, Z., Dimitrov, V.: How to Support Teams to be Remote and Productive: Group Decision-Making for Distance Collaboration Software Tools. Information & Security: An International Journal, vol. 46, no. 1, 36–52, 2020. ISSN 0861-5160 DOI 10.11610/isij.4603

    2. Terzieva, M., Karastoyanov, D.: ICT for Innovation in Advanced Banking. Problems of Engineering Cybernetics and Robotics, vol. 73, 47–54, 2020. ISSN 2738-7356 DOI 10.7546/PECR.73.20.05

    3. Borissova, D., Dimitrova, Z. Garvanova, M., Garvanov, I., Cvetkova, P., Dimitrov, V., Pandulis, A.: Two-stage Decision-Making Approach to Survey the Excessive Usage of Smart Technologies, Problems of Engineering Cybernetics and Robotics, vol. 73, 3–16, 2020. ISSN 2738-7356 DOI 10.7546/PECR.73.20.01

- Zankinski, I., Barova, M., Tomov, P.: Hybrid Approach Based on Combination of Backpropagation and Evolutionary Algorithms for Artificial Neural Networks Training by Using Mobile Devices in Distributed Computing Environment. Proceedings of 11th International Conference on Large-Scale Scientific Computations LSSC'17, June 5-9, 2017, Sozopol, Bulgaria, 425–434, 2017. ISBN 978-3-319-73440-8 DOI 10.1007/978-3-319-73441-5_46

    4. Koprinkova-Hristova, P.: Research on Artificial Neural Networks in Bulgarian Academy of Sciences. Research in Computer Science in the Bulgarian Academy of Sciences, vol. 934, 287–304, 2021. ISBN 978-3-030-72283-8 DOI 10.1007/978-3-030-72284-5_14

- Tomov, P., Monov, V.: Artificial Neural Networks and Differential Evolution Used for Time Series Forecasting in Distributed Environment. Proceedings of the International Conference Automatics and informatics, October 4-5, 2016, Sofia, Bulgaria, Federation of the scientific engineering unions, John Atanasoff Society of Automatics and Informatics, 129–132, 2016. ISSN 1313-1850

5. Balabanov, T.: Long Short Term Memory in MLP Pair. Proceedings of the International Scientific Conference UniTech, vol. II, 375–379, 2017. ISSN:1313-230X

6. Balabanov, T., Atanasova, T., Blagoev, I.: Activation Function Permutation for Multilayer Perceptron Training. Proceedings of International Conference on Big Data, Knowledge and Control Systems Engineering, 9–14 , 2018. ISSN 2367-6450.

7. Balabanov, T., Zankinski, I., Ketipov, R.: Weights Permutation in Multilayer Perceptron. Proceedings of International Conference on Big Data, Knowledge and Control Systems Engineering, 23–27 , 2018. ISSN 2367-6450

8. Blagoev, I., Sevova, J., Kolev, K.: Dual MLP Pairs with Hidden Layer Sharing. Proceedings of 32nd International Conference on Information Technologies, 81–86, 2018. ISSN 1314-1023

9. Blagoev, I., Sevova, J., Kolev, K.: Artificial Neural Network Activation Function Optimization with Genetic Algorithms. Proceedings of the International Conference Numerical Methods for Scientific Computations and Advanced Applications, 16–19, 2018. ISBN 978-954-91700-7-8

- Keremedchiev, D., Barova, M., Tomov, P.: Mobile Application as Distributed Computing System for Artificial Neural Networks Training Used in Perfect Information Games. International Scientific Conference UniTech'16, Gabrovo, University publishing house Vasil Aprilov, 389–393, 2016. ISSN 1313-230X

10. Balabanov, T.: Long Short Term Memory in MLP Pair. Proceedings of the

International Scientific Conference UniTech, vol. II, 375–379, 2017. ISSN:1313-230X

11. Blagoev, I., Sevova, J., Kolev, K.: Dual MLP Pairs with Hidden Layer Sharing. Proceedings of 32nd International Conference on Information Technologies, 81–86, 2018. ISSN 1314-1023

12. Blagoev, I., Sevova, J., Kolev, K.: Artificial Neural Network Activation Function Optimization with Genetic Algorithms. Proceedings of the International Conference Numerical Methods for Scientific Computations and Advanced Applications, 16–19, 2018. ISBN 978-954-91700-7-8

Награди

Награда в състезание за глобална скалируема оптимизация, провело се като част от Международната конференция за високопроизводителни изчисления, 2019 година.

Декларация за оригиналност на резултатите

Декларирам, че настоящата дисертация съдържа оригинални резултати, получени при проведени от мен научни изследвания, с подкрепата и съдействието на научния ми ръководител. Резултатите, които са получени, описани и/или публикувани от други учени, са надлежно и подробно цитирани в библиографията.

Настоящата дисертация не е прилагана за придобиване на научна степен в друго висше училище, университет или научен институт.

Дата: ..... Подпис: .....

гр. София / Петър Томов /

Библиография

[1] Ahmad, F., Abidin, S.: Applications of Genetic Algorithm in Distributed Computing. International Refereed Journal of Engineering and Science, vol. 8, no. 2, 9–12, (2018). ISSN 2319-1821 DOI 10.183x/B0802010912

[2] Alba, E., Tomassini, M.: Parallelism and evolutionary algorithms. IEEE Transactions on Evolutionary Computation, vol. 6, no. 5, 443–462, (2002). ISSN 1089-778X DOI 10.1109/TEVC.2002.800880

[3] Altinoz, O., Deb, K.: Late parallelization and feedback approaches for distributed computation of evolutionary multi-objective optimization algorithms. Neural Computing and Applications, vol. 30, 723–733, (2018). ISSN 0941-0643 DOI 10.1007/s00521-016-2573-4

[4] Balabanov, T., Zankinski, I., Barova, M.: Distributed Evolutionary Computing Migration Strategy by Incident Node Participation. Proceedings of Large-Scale Scientific Computing, vol. 9374, 203–209, (2015). ISBN 978-3-319-26519-3 DOI 10.1007/978-3-319-26520-9_21

[5] Basheer, I., Hajmeer, M.: Artificial neural networks: fundamentals, computing, design, and application. Journal of Microbiological Methods, vol. 43, no. 1, 3–31, (2000). ISSN 0167-7012 DOI 10.1016/S0167-7012(00)00201-3

[6] Beyer, H.: Evolutionary algorithms in noisy environments: theoretical issues and guidelines for practice. Computer Methods in Applied Mechanics and Engineering, vol. 186, no. 2-4, 239–267, (2000). ISSN 0045-7825 DOI 10.1016/S0045-7825(99)00386-2

[7] Beyer, H., Schwefel, H., Wegener, I.: How to analyse evolutionary algorithms. Theoretical Computer Science, vol. 287, no. 1, 101–130, (2002). ISSN 0304-3975 DOI 10.1016/S0304-3975(02)00137-8

[8] Bilbao, I., Bilbao, J.: Overfitting problem and the over-training in the era of data: Particularly for Artificial Neural Networks. Proceedings of Eighth International Conference on Intelligent Computing and Information Systems, 173–177, (2017). ISBN 978-1-5386-0822-7 DOI 10.1109/INTELCIS.2017.8260032

[9] Bing, Y., Hao, J., Zhang, S.: Stock Market Prediction Using Artificial Neural Networks. Advanced Engineering Forum, vol. 6-7, 1055–1060, (2012). ISSN 2234-991X DOI 10.4028/www.scientific.net/AEF.6-7.1055

[10] Bontempi, G., Ben Taieb, S., Le Borgne Y.: Machine Learning Strategies for Time Series Forecasting. Lecture Notes in Business Information Processing, vol 138, 62–77, (2013). ISBN 978-3-642-36317-7 DOI 10.1007/978-3-642-36318-4_3

[11] Cao, L., Tay, F.: Support vector machine with adaptive parameters in financial time series forecasting. IEEE Transactions on Neural Networks, vol. 14, no. 6, 1506–1518, (2003). ISSN 1045-9227 DOI 10.1109/TNN.2003.820556

[12] Cao, J., Li, Z., Li, J.: Financial time series forecasting model based on CEEMDAN and LSTM. Physica A: Statistical Mechanics and its Applications, vol. 519, 127–139, (2019). ISSN 0378-4371 DOI 10.1016/j.physa.2018.11.061

[13] Caponetto, R., Fortuna, L., Fazzino, S., Xibilia, M.: Chaotic sequences to improve the performance of evolutionary algorithms. IEEE Transactions on Evolutionary Computation, vol. 7, no. 3, 289–304, (2003). DOI 10.1109/TEVC.2003.810069

[14] Castillo, P., Garcia-Sanchez, P., Arenas, M., Bernier, J., Merelo, J.: Distributed Evolutionary Computation Using SOAP and REST Web Services. Advances in Intelligent Modelling and Simulation, vol. 422, 89–111, (2012). ISBN 978-3-642-30153-7 DOI 10.1007/978-3-642-30154-4_5

[15] Chandwani, V., Agrawal, V., Nagar, R.: Modeling slump of ready mix concrete using genetic algorithms assisted training of Artificial Neural Networks. Expert Systems with Applications, vol. 42, no. 2, 885–893, (2015). ISSN 0957-4174 DOI 10.1016/j.eswa.2014.08.048

[16] Chen, Y., Yang, B., Dong, J., Abraham. A.: Time-series forecasting using flexible neural tree model. Information Sciences, vol. 174, no. 3–4, 219–235, (2005). ISSN 0020-0255 DOI 10.1016/j.ins.2004.10.005

[17] Chen, J., Do, Q., Hsieh, H.: Training Artificial Neural Networks by a Hybrid PSO-CS Algorithm. Algorithms, vol. 8, no. 2, 292–308, (2015). ISSN 1999-4893 DOI 10.3390/a8020292

[18] Cheng, R., He, C., Jin, Y., Yao, X.: Model-based evolutionary algorithms: a short survey. Complex & Intelligent Systems, vol. 4, 283–292, (2018). ISSN 2198-6053 DOI 10.1007/s40747-018-0080-1

[19] Coello, C.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Computer Methods in Applied Mechanics and Engineering, vol. 191, no. 11-12, 1245–1287, (2002). ISSN 0045-7825 DOI 10.1016/S0045-7825(01)00323-1

[20] Coello, C.: An Introduction to Evolutionary Algorithms and Their Applications. Proceedings of International Symposium and School on Advanced Distributed Systems, vol. 3563, 425–442, (2005). ISBN 978-3-540-28063-7 DOI 10.1007/11533962_39

[21] Cole, N., Desell, T., Gonzalez, D., de Vega, F., Magdon-Ismail, M., Newberg, H., Szymanski, B., Varela, C.: Evolutionary Algorithms on Volunteer Computing Platforms: The MilkyWay@Home Project. Parallel and Distributed Computational Intelligence, vol. 269, 63–90, (2010). ISBN 978-3-642-10674-3 DOI 10.1007/978-3-642-10675-0_4

[22] Cortez, P., Rocha, M., Neves, J.: Evolving Time Series Forecasting ARMA Models. Journal of Heuristics, vol. 10, no. 4, 415–429, (2004). ISSN 1381-1231 DOI 10.1023/B:HEUR.0000034714.09838.1e

[23] Coulibaly, P., Anctil, F., Bobee, B.: Daily reservoir inflow forecasting using artificial neural networks with stopped training approach. Journal of Hydrology, vol. 230, no. 3-4, 244–257, (2000). ISSN 0022-1694 DOI 10.1016/S0022-1694(00)00214-6

[24] Cui, Z., Yang, C., Sanyal, S.: Training artificial neural networks using APPM. International Journal of Wireless and Mobile Computing, vol. 5, no. 2, 168–174, (2012). ISSN 1741-1084 DOI 10.1504/IJWMC.2012.046787

[25] da Silva, I., Hernane Spatti, D., Andrade Flauzino, R., Liboni, L., dos Reis Alves, S.: Artificial Neural Network Architectures and Training Processes. Artificial Neural Networks, 21–28, (2017). ISBN 978-3-319-43161-1 DOI 10.1007/978-3-319-43162-8_2

[26] Desell, T., Anderson, D., Magdon-Ismail, M., Newberg, H., Szymanski, B., Varela, C.: An analysis of massively distributed evolutionary algorithms. Proceedings of IEEE Congress on Evolutionary Computation, 1–8, (2010). ISBN 978-1-4244-6909-3 DOI 10.1109/CEC.2010.5586073

[27] Desell, T., Magdon-Ismail, M., Szymanski, B., Varela, C., Newberg, H., Anderson, D.: Validating Evolutionary Algorithms on Volunteer Computing Grids. Proceedings of International Conference on Distributed Applications and Interoperable Systems, vol. 6115, 29–41, (2010). ISBN 978-3-642-13644-3 DOI 10.1007/978-3-642-13645-0_3

[28] Desell, T., Clachar, S., Higgins, J., Wild, B.: Evolving Neural Network Weights for Time-Series Prediction of General Aviation Flight Data. Proceedings of International Conference on Parallel Problem Solving from Nature, vol. 8672, 771–781, (2014). ISBN 978-3-319-10761-5 DOI 10.1007/978-3-319-10762-2_76

[29] Desell, T.: Large scale evolution of convolutional neural networks using volunteer computing. Proceedings of the Genetic and Evolutionary Computation

Conference Companion, 127–128, (2017). ISBN 978-1-4503-4939-0 DOI 10.1145/3067695.3076002

[30] Ding, Y., Hu, Z., Sun, H.: An antibody network inspired evolutionary framework for distributed object computing. Information Sciences, vol. 178, no. 24, 4619–4631, (2008). ISSN 0020-0255 DOI 10.1016/j.ins.2008.08.010

[31] Ding, S., Li, H., Su, C., Yu, J., Jin, F.: Evolutionary artificial neural networks: a review. Artificial Intelligence Review, vol. 39, no. 3, 251–260, (2013). ISSN 0269-2821 DOI 10.1007/s10462-011-9270-6

[32] Diosan, L., Oltean, M.: Evolutionary design of Evolutionary Algorithms. Genetic Programming and Evolvable Machines, vol. 10, 263–306, (2009). ISSN 1389-2576 DOI 10.1007/s10710-009-9081-6

[33] Duda, J., Dlubacz, W.: Improving Efficiency of a Web-Based Distributed Evolutionary Computing System. Proceedings of International Conferences Big Data Analytics, Data Mining and Computational Intelligence and Theory and Practice in Modern Computing, 143–150, (2019). ISBN 978-989-8533-92-0 DOI 10.33965/tpmc2019_201907L018

[34] Eiben, A., Michalewicz, Z., Schoenauer, M., Smith, J.: Parameter Control in Evolutionary Algorithms. Parameter Setting in Evolutionary Algorithms, vol. 54, 19–46, (2007). ISBN 978-3-540-69431-1 DOI 10.1007/978-3-540-69432-8_2

[35] Eiben, A., Smit, S.: Parameter tuning for configuring and analyzing evolutionary algorithms. Swarm and Evolutionary Computation, vol. 1, no. 1, 19–31, (2011). ISSN 2210-6502 DOI 10.1016/j.swevo.2011.02.001

[36] El-Abd, M.: Performance assessment of foraging algorithms vs. evolutionary algorithms. Information Sciences, vol. 182, no. 1, 243–263, (2012). ISSN 0020-0255 DOI 10.1016/j.ins.2011.09.005

[37] Epitropakis, M., Plagianakos, V., Vrahatis, M.: Hardware-friendly Higher-Order Neural Network Training using Distributed Evolutionary Algorithms.

Applied Soft Computing, vol. 10, no. 2, 398–408, (2010). ISSN 1568-4946 DOI 10.1016/j.asoc.2009.08.010

[38] Ertugrul, O.: A novel type of activation function in artificial neural networks: Trained activation function. Neural Networks, vol. 99, 148–157, (2018). ISSN 0893-6080 DOI 10.1016/j.neunet.2018.01.007

[39] Foo, Y., Goh, C., Lim, H., Zhan, Z., Li, Y.: Evolutionary Neural Network Based Energy Consumption Forecast for Cloud Computing. Proceedings of International Conference on Cloud Computing Research and Innovation, 53–64, (2015). ISBN 978-1-5090-0144-6 DOI 10.1109/ICCCRI.2015.17

[40] Francois, O., Lavergne, C.: Design of evolutionary algorithms-A statistical perspective. IEEE Transactions on Evolutionary Computation, vol. 5, no. 2, 129–148, (2001). ISSN 1089-778X DOI 10.1109/4235.918434

[41] Gong, Y., Chen, W., Zhan, Z., Zhang, J., Li, Y., Zhang, Q., Li, J.: Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. Applied Soft Computing, vol. 34, 286–300, (2015). ISSN 1568-4946 DOI 10.1016/j.asoc.2015.04.061

[42] Gooijer, J., Hyndman, R.: 25 years of time series forecasting. International Journal of Forecasting, vol. 22, no. 3, 443–473, (2006). ISSN 0169-2070 DOI 10.1016/j.ijforecast.2006.01.001

[43] Grosan, C., Abraham, A.: Hybrid Evolutionary Algorithms: Methodologies, Architectures, and Reviews. Hybrid Evolutionary Algorithms, vol 75, 1–17, (2007). ISBN 978-3-540-73296-9 DOI 10.1007/978-3-540-73297-6_1

[44] Guo, Y., Li, J., Zhan, Z.: Efficient Hyperparameter Optimization for Convolution Neural Networks in Deep Learning: A Distributed Particle Swarm Optimization Approach. Cybernetics and Systems, vol. 52, no. 1, 36–57, (2021). ISSN 0196-9722 DOI 10.1080/01969722.2020.1827797

[45] Hamzacebi, C.: Improving artificial neural networks' performance in seasonal time series forecasting. Information Sciences, vol. 178, no. 23, 4550–4559, (2008). ISSN 0020-0255 DOI 10.1016/j.ins.2008.07.024

[46] He, J., Yu, X.: Conditions for the convergence of evolutionary algorithms. Journal of Systems Architecture, vol. 47, no. 7, 601–612, (2001). ISSN 1383-7621 DOI 10.1016/S1383-7621(01)00018-2

[47] He, J., Yao, X.: Towards an analytic framework for analysing the computation time of evolutionary algorithms. Artificial Intelligence, vol. 145, no. 1-2, 59–97, (2003). ISSN 0004-3702 DOI 10.1016/S0004-3702(02)00381-8

[48] Jain, A., Mao, J., Mohiuddin, K.: Artificial neural networks: a tutorial. Computer, vol. 29, no. 3, 31–44, (1996). ISSN 0018-9162 DOI 10.1109/2.485891

[49] Jain, A., Kumar, A.: Hybrid neural network models for hydrologic time series forecasting. Applied Soft Computing, vol. 7, no. 2, 585–592, (2007). ISSN 1568-4946 DOI 10.1016/j.asoc.2006.03.002

[50] Joo, T., Kim, S.: Time series forecasting based on wavelet filtering. Expert Systems with Applications, vol. 42, no. 8, 3868–3874, (2015). ISSN 0957-4174 DOI 10.1016/j.eswa.2015.01.026

[51] Kachitvichyanukul, V.: Comparison of Three Evolutionary Algorithms: GA, PSO, and DE. Industrial Engineering and Management Systems, vol. 11, no. 3, 215–223, (2012). ISSN 1598-7248 DOI 10.7232/iems.2012.11.3.215

[52] Karaboga, D., Akay, B., Ozturk, C.: Artificial Bee Colony (ABC) Optimization Algorithm for Training Feed-Forward Neural Networks. Proceedings of International Conference on Modeling Decisions for Artificial Intelligence, vol. 4617, 318–329, (2007). ISBN 978-3-540-73728-5 DOI 10.1007/978-3-540-73729-2_30

[53] Karafotias, G., Hoogendoorn, M., Eiben, A.: Parameter Control in Evolutionary Algorithms: Trends and Challenges. IEEE Transactions on Evolutionary

Computation, vol. 19, no. 2, 167–187, (2015). ISSN 1089-778X DOI 10.1109/TEVC.2014.2308294

[54] Karnin, E.: A simple procedure for pruning back-propagation trained neural networks. IEEE Transactions on Neural Networks, vol. 1, no. 2, 239–242, (1990). ISSN 1045-9227 DOI 10.1109/72.80236

[55] Kattan, A., Abdullah, R., Salam, R.: Harmony Search Based Supervised Training of Artificial Neural Networks. Proceedings of International Conference on Intelligent Systems, Modelling and Simulation, 105–110, (2010). ISBN 978-1-4244-5984-1 DOI 10.1109/ISMS.2010.31

[56] Kattan, A., Abdullah, R., Salam, R.: Training Feed-Forward Neural Networks Using a Parallel Genetic Algorithm with the Best Must Survive Strategy. Proceedings of International Conference on Intelligent Systems, Modelling and Simulation, 96–99, (2010). ISBN 978-1-4244-5984-1 DOI 10.1109/ISMS.2010.29

[57] Kazimipour, B., Li, X., Qin, A.: A review of population initialization techniques for evolutionary algorithms. Proceedings of IEEE Congress on Evolutionary Computation, 2585–2592, (2014). ISBN 978-1-4799-1488-3 DOI 10.1109/CEC.2014.6900618

[58] Keremedchiev, D., Tomov, P., Barova, M.: Slot machine base game evolutionary RTP optimization. Proceedings of 6th International Conference on Numerical Analysis and Its Applications, vol. 10187, 406–413, (2016). ISBN 978-331957098-3 DOI 10.1007/978-3-319-57099-0_45

[59] Kern, S., Muller, S., Hansen, N., Buche, D., Ocenasek, J., Koumoutsakos, P.: Learning probability distributions in continuous evolutionary algorithms - a comparative review. Natural Computing, vol. 3, 77–112, (2004). ISSN 1572-9796 DOI 10.1023/B:NACO.0000023416.59689.4e

[60] Khashei, M., Bijari, M.: An artificial neural network (p,d,q) model for timeseries forecasting. Expert Systems with Applications, vol. 37, no. 1, 479–489, (2010). ISSN 0957-4174 DOI 10.1016/j.eswa.2009.05.044

[61] Khashei, M., Bijari, M.: A novel hybridization of artificial neural networks and ARIMA models for time series forecasting. Applied Soft Computing, vol. 11, no. 2, 2664–2675, (2011). ISSN 1568-4946 DOI 10.1016/j.asoc.2010.10.015

[62] Khashei, M., Bijari. M.: A new class of hybrid models for time series forecasting. Expert Systems with Applications, vol. 39, no. 4, 4344-4357, (2012). ISSN 0957-4174 DOI 10.1016/j.eswa.2011.09.157

[63] Kingston, G., Lambert, M., Maier, H.: Bayesian training of artificial neural networks used for water resources modeling. Water Resources Research, vol. 41, no. 12, W12409, (2005). ISSN 0043-1397 DOI 10.1029/2005WR004152

[64] Koleva, D., Barova, M., Tomov, P.: 2D optimal packing with population based algorithms. Proceedings of 11th International Conference on Large-Scale Scientific Computations, vol. 10665, 366–373, (2018). ISBN 978-3-319-73440-8 DOI 10.1007/978-3-319-73441-5_39

[65] Kollias, S., Anastassiou, D.: An adaptive least squares algorithm for the efficient training of artificial neural networks. IEEE Transactions on Circuits and Systems, vol. 36, no. 8, 1092–1101, (1989). ISSN 0098-4094 DOI 10.1109/31.192419

[66] Koziel, S., Michalewicz, Z.: Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization. Evolutionary Computation, vol. 7, no. 1, 19–44, (1999). ISSN 1063-6560 DOI 10.1162/evco.1999.7.1.19.

[67] Kubat, M.: Artificial Neural Networks. An Introduction to Machine Learning, 91–111, (2015). ISBN 978-3-319-20009-5 DOI 10.1007/978-3-319-20010-1_5

[68] Kumar, J., Singh, A.: Cloud Resource Demand Prediction using Differential Evolution based Learning. Proceedings of 7th International Conference on

Smart Computing & Communications, 1–5, (2019). ISBN 978-1-7281-1558-0 DOI 10.1109/ICSCC.2019.8843680

[69] Kyoung-jae, K.: Financial time series forecasting using support vector machines. Neurocomputing, vol. 55, no. 1, 307–319, (2003). ISSN 0925-2312 DOI 10.1016/S0925-2312(03)00372-2

[70] Lachtermacher, G., Fuller, J: Back propagation in time-series forecasting. Journal of Forecasting, vol. 14, no. 4, 381–393, (1995). ISSN 1099-131X DOI 10.1002/for.3980140405

[71] Lagaros, N., Papadrakakis, M., Kokossalakis, G.: Structural optimization using evolutionary algorithms. Computers & Structures, vol. 80, no. 7-8, 571–589, (2002). ISSN 0045-7949 DOI S0045-7949(02)00027-5

[72] Liu, X., Zhan, Z., Zhang, J.: Resource-Aware Distributed Differential Evolution for Training Expensive Neural-Network-Based Controller in Power Electronic Circuit. IEEE Transactions on Neural Networks and Learning Systems, 1–11, (2021). ISSN 2162-237X DOI 10.1109/TNNLS.2021.3075205

[73] Lu, C., Lee, T., Chiu. C.: Financial time series forecasting using independent component analysis and support vector regression. Decision Support Systems, vol. 47, no. 2, 115–125, (2009). ISSN 0167-9236 DOI 10.1016/j.dss.2009.02.001

[74] Mendivil, S., Castillo, O., Melin, P.: Optimization of Artificial Neural Network Architectures for Time Series Prediction Using Parallel Genetic Algorithms. Soft Computing for Hybrid Intelligent Systems, vol. 154, 387–399, (2008). ISBN 978-3-540-70811-7 DOI 10.1007/978-3-540-70812-4_23

[75] Merelo-Guervos, J.: Low or no cost distributed evolutionary computation. Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation, 703–706, (2015). ISBN 978-1-4503-3488-4 DOI 10.1145/2739482.2756560

[76] Mocanu, D., Mocanu, E., Stone, P., Nguyen, P., Gibescu, M., Liotta, A.: Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. Nature Communications, vol. 9, 2383, (2018). ISSN 2041-1723 DOI 10.1038/s41467-018-04316-3

[77] Naudts, B., Kallel, L.: A comparison of predictive measures of problem difficulty in evolutionary algorithms. IEEE Transactions on Evolutionary Computation, vol. 4, no. 1, 1–15, (2000). ISSN 1089-778X DOI 10.1109/4235.843491

[78] Nawi, N., Atomi, W., Rehman, M.: The Effect of Data Pre-processing on Optimized Training of Artificial Neural Networks. Procedia Technology, vol. 11, 32–39, (2013). ISSN 2212-0173 DOI 10.1016/j.protcy.2013.12.159

[79] Nelson, M., Hill, T., Remus, W., O'Connor, M.: Time series forecasting using neural networks: should the data be deseasonalized first?. Journal of Forecasting, vol. 18, no. 5, 359–367, (1999). ISSN 1099-131X DOI 10.1002/(SICI)1099-131X(199909)18:5<359::AID-FOR746>3.0.CO;2-P

[80] Oliveira, M., Torgo, L.: Ensembles for Time Series Forecasting. Proceedings of the Sixth Asian Conference on Machine Learning, vol. 39, 360–370, (2015). ISSN 2640-3498

[81] Pandey, S., Tapaswi, S., Srivastava, L.: Integrated evolutionary neural network approach with distributed computing for congestion management. Applied Soft Computing, vol. 10, no. 1, 251–260, (2010). ISSN 1568-4946 DOI 10.1016/j.asoc.2009.07.008

[82] Piotrowski, A., Osuch, M., Napiorkowski, M., Rowinski, P., Napiorkowski, J.: Comparing large number of metaheuristics for artificial neural networks training to predict water temperature in a natural river. Computers & Geosciences, vol. 64, 136–151, (2014). ISSN 0098-3004 DOI 10.1016/j.cageo.2013.12.013

[83] Plagianakos, V., Tasoulis, D., Vrahatis, M.: A Review of Major Application Areas of Differential Evolution. Advances in Differential Evolution, vol. 143, 197–238, (2008). ISBN 978-3-540-68827-3 DOI 10.1007/978-3-540-68830-3_8

[84] Pomerleau, D: Efficient Training of Artificial Neural Networks for Autonomous Navigation. Neural Computation, vol. 3, no. 1, 88–97, (1991). ISSN 0899-7667 DOI 10.1162/neco.1991.3.1.88

[85] Pradhan, B., Naghibi, S., Motevalli, A., Mansor, S: Assessment of the effects of training data selection on the landslide susceptibility mapping: a comparison between support vector machine (SVM), logistic regression (LR) and artificial neural networks (ANN). Geomatics, Natural Hazards and Risk, vol. 9, no. 1, 49–69, (2018). ISSN 1947-5705 DOI 10.1080/19475705.2017.1407368

[86] Qi, M., Zhang, G.: An investigation of model selection criteria for neural network time series forecasting. European Journal of Operational Research, vol. 132, no. 3, 666–680, (2001). ISSN 0377-2217 DOI 10.1016/S0377-2217(00)00171-5

[87] Qu, X., Wang, J., Xiao, J.: ParallelNAS: A Parallel and Distributed System for Neural Architecture Search. Proceedings of IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems, 247–254, (2020). ISBN 978-1-7281-7650-5 DOI 10.1109/HPCC-SmartCity-DSS50907.2020.00031

[88] Rivas, V., Parras-Gutierrez, E., Merelo, J., Arenas, M., Garcia-Fernandez, P.: Time series forecasting using evolutionary neural nets implemented in a volunteer computing system. Intelligent Systems in Accounting, Finance and Management, vol. 24, no. 2-3, 87–95, (2017). ISSN 1055-615X DOI 10.1002/isaf.1409

[89] Roy, A., Dutta, D., Choudhury, K.: Training Artificial Neural Network using Particle Swarm Optimization Algorithm. International Journal of Advanced Research in

Computer Science and Software Engineering, vol. 3, no. 3, 430–434, (2013). ISSN 2277-128X

[90] Ryerkerk, M., Averill, R., Deb, K., Goodman, E.: A survey of evolutionary algorithms using metameric representations. Genetic Programming and Evolvable Machines, vol. 20, 441–478, (2019). ISSN 1389-2576 DOI 10.1007/s10710-019-09356-2

[91] Salami, M., Hendtlass, T.: A fast evaluation strategy for evolutionary algorithms. Applied Soft Computing, vol. 2, no. 3, 156–173, (2003). ISSN 1568-4946 DOI 10.1016/S1568-4946(02)00067-4

[92] Sequin, C., Clay, R.: Fault tolerance in artificial neural networks. Proceedings of International Joint Conference on Neural Networks, vol. 1, 703–708, (1990). DOI 10.1109/IJCNN.1990.137651

[93] Sharma, S.: Applications of Genetic Algorithm in Software Engineering, Distributed Computing and Machine Learning. International Journal of Computer Applications & Information Technology, vol. 9, no. 2, 208–212, (2016). ISSN 2278-7720

[94] Shen, Z., Zhang, Y., Lu, J., Xu, J., Xiao, G.: A novel time series forecasting model with deep learning. Neurocomputing, vol. 396, 302–313, (2020). ISSN 0925-2312 DOI 10.1016/j.neucom.2018.12.084

[95] Sietsma, J., Dow, R.: Creating artificial neural networks that generalize. Neural Networks, vol. 4, no. 1, 67–79, (1991). ISSN 0893-6080 DOI 10.1016/0893-6080(91)90033-2

[96] Singh, S.: Pattern Modelling in Time-series Forecasting. Cybernetics and Systems, vol. 31, no. 1, 49–65, (2000). ISSN 0196-9722 DOI 10.1080/019697200124919

[97] Slowik, A., Bialko, M.: Training of artificial neural networks using differential evolution algorithm. Proceedings of Conference on Human System Interactions, 60–65, (2008). ISBN 978-1-4244-1542-7 DOI 10.1109/HSI.2008.4581409

[98] Slowik, A., Kwasnicka, H.: Evolutionary algorithms and their applications to engineering problems. Neural Computing and Applications vol. 32, 12363–12379, (2020). ISSN 0941-0643 DOI 10.1007/s00521-020-04832-8

[99] Tan, K., Yu, Q., Lee, T.: A distributed evolutionary classifier for knowledge discovery in data mining. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 35, no. 2, 131–142, (2005). ISSN 1094-6977 DOI 10.1109/TSMCC.2004.841911

[100] Tang, Z., de Almeida, C., Fishwick, P.: Time series forecasting using neural networks vs Box-Jenkins methodology. SIMULATION, vol. 57, no. 5, 303-310, (1991). ISSN 0037-5497 DOI 10.1177/003754979105700508

[101] Tay, F., Cao, L.: Application of support vector machines in financial time series forecasting. Omega, vol. 29, no. 4, 309–317, (2001). ISSN 0305-0483 DOI 10.1016/S0305-0483(01)00026-3

[102] Tealab, A.: Time series forecasting using artificial neural networks methodologies: A systematic review. Future Computing and Informatics Journal, vol. 3, no. 2, 334–340, (2018). ISSN 2314-7288 DOI 10.1016/j.fcij.2018.10.003

[103] Tiwari, A., Goteng, G., Roy, R.: Evolutionary Computing within Grid Environment. Advances in Evolutionary Computing for System Design, vol. 66, 229–248, (2007). ISBN 978-3-540-72376-9 DOI 10.1007/978-3-540-72377-6_10

[104] Tomov, P., Zankinski, I., Barova, M.: Mobile Alternative of the Moneybee Project For Financial Forecasting. Proceedings of the Annual University Scientific Conference of the National Military University Vasil Levski, 1085–1089, (2018). ISSN 2367-7481

[105] Tomov, P., Zankinski, I., Barova, M.: Artificial Neural Networks Time Series Forecasting with Android Live Wallpaper Technology. Proceedings of the International Conference Numerical Methods for Scientific Computations and Advanced Applications, 76–79, (2018). ISBN 978-954-91700-7-8

[106] Tomov, P., Zankinski, I., Balabanov, T.: Training of Artificial Neural Networks for Financial Time Series Forecasting in Android Service and Widgets. Problems of Engineering Cybernetics and Robotics, vol. 71, 50–56, (2019). ISSN 1314-409X

[107] Tomov, P., Zankinski, I., Balabanov, T.: Genetic algorithm selection operator based on recursion and Brute-Force. Proceedings of 14th Annual Meeting of the Bulgarian Section of SIAM, 93–93, (2019). ISSN 1313-3357

[108] Tomov, P., Zankinski, I., Balabanov, T.: Server Side Vote Clustering in Human-Computer Distributed Computing. Information Technologies Control, vol. 2, no. 3, 15–19, (2019). ISSN 2367-5357 DOI 10.7546/itc-2019-0008

[109] Tomov, P., Zankinski, I., Danev, V.: Local Search, Brute-Force and Recursion for Selection Operator. Problems of Engineering Cybernetics and Robotics, vol. 74, 24–32, (2020). ISSN 2738-7364 DOI 10.7546/PECR.74.20.03

[110] Tomov, P.: Multilayer Perceptron Fast Prototyping with Differential Evolution and Particle Swarm Optimization in LibreOffice Calc. Problems of Engineering Cybernetics and Robotics, vol. 75, 5–14, (2021). ISSN 2738-7356 DOI 10.7546/PECR.75.21.02

[111] Tomov, P.: Encog Gradient Training Algorithms Evaluation. Problems of Engineering Cybernetics and Robotics, vol. 77, 11–19, (2021). ISSN 2738-7356 DOI 10.7546/PECR.77.21.02

[112] Topping, B., Sziveri, J., Bahreinejad, A., Leite, J., Cheng, B.: Parallel processing, neural networks and genetic algorithms. Advances in Engineering Software, vol. 29, no. 10, 763–786, (1998). ISSN 0965-9978 DOI 10.1016/S0965-9978(97)00062-8

[113] Ursem, R.: Diversity-Guided Evolutionary Algorithms. Proceedings of International Conference on Parallel Problem Solving from Nature, vol. 2439, 462–471, (2002). ISBN 978-3-540-44139-7 DOI 10.1007/3-540-45712-7_45

[114] Varacha, P., Zelinka, I.: Analytic Programming Powered by Distributed Self-Organizing Migrating Algorithm Application. Proceedings of 7th Computer

Information Systems and Industrial Management Applications, 99–100, (2008). ISBN 978-0-7695-3184-7 DOI 10.1109/CISIM.2008.50

[115] Velichkova, V., Tomov, P., Balabanov, T.: Incremental Sinusoidal Approximation of Time Series with LibreOffice Calc Solver. Problems of Engineering Cybernetics and Robotics, vol. 75, 43–50, (2021). ISSN 2738-7364 DOI 10.7546/PECR.75.21.05

[116] Viharos, Z., Monostori, L., Vincze, T.: Training and Application of Artificial Neural Networks with Incomplete Data. Proceedings of International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, vol 2358, 649–659, (2002). ISBN 978-3-540-43781-9 DOI 10.1007/3-540-48035-8_63

[117] Vikhar, P.: Evolutionary algorithms: A critical review and its future prospects. Proceedings of International Conference on Global Trends in Signal Processing, Information Computing and Communication, 261–265, (2016). ISBN 978-1-5090-0468-3 DOI 10.1109/ICGTSPICC.2016.7955308

[118] Wagner, N., Michalewicz, Z., Khouja, M., McGregor, R.: Time Series Forecasting for Dynamic Environments: The DyFor Genetic Program Model. IEEE Transactions on Evolutionary Computation, vol. 11, no. 4, 433–452, (2007). ISSN 1089-778X DOI 10.1109/TEVC.2006.882430

[119] Wegener, I.: Theoretical Aspects of Evolutionary Algorithms. Proceedings of International Colloquium on Automata, Languages, and Programming, vol 2076, 64–78, (2001). ISBN 978-3-540-42287-7 DOI 10.1007/3-540-48224-5_6

[120] Wang, B., Huang, H., Wang, X.: A novel text mining approach to financial time series forecasting, Neurocomputing, vol. 83, 136–145, (2012). ISSN 0925-2312 DOI 10.1016/j.neucom.2011.12.013

[121] Wang, J., Gong, B., Liu, H., Li, S., Yi, J.: Heterogeneous computing and grid scheduling with parallel biologically inspired hybrid heuristics. Transactions of the Institute of Measurement and Control, vol. 36, no. 6, 805–814, (2014). ISSN 0142-3312 DOI 10.1177/0142331214522287

[122] Whitley, D., Rana, S., Dzubera, J., Mathias, K.: Evaluating evolutionary algorithms. Artificial Intelligence, vol. 85, no. 1-2, 245–276, (1996). ISSN 0004-3702 DOI 10.1016/0004-3702(95)00124-7

[123] Wilamowski, B., Iplikci, S., Kaynak, O., Efe, M.: An algorithm for fast convergence in training neural networks. Proceedings of International Joint Conference on Neural Networks, vol.3, 1778–1782, (2001). ISBN 0-7803-7044-9 DOI 10.1109/IJCNN.2001.938431

[124] Yan, W.: Toward Automatic Time-Series Forecasting Using Neural Networks. IEEE Transactions on Neural Networks and Learning Systems, vol. 23, no. 7, 1028–1039, (2012). ISSN 2162-237X DOI 10.1109/TNNLS.2012.2198074

[125] Yao, X., Liu, Y.: A new evolutionary system for evolving artificial neural networks. IEEE Transactions on Neural Networks, vol. 8, no. 3, 694–713, (1997). ISSN 1045-9227 DOI 10.1109/72.572107

[126] Yao, X., Liu, Y., Liang, K., Lin, G.: Fast Evolutionary Algorithms. Advances in Evolutionary Computing, 45–94, (2003). ISBN 978-3-642-62386-8 DOI 10.1007/978-3-642-18965-4_2

[127] Zankinski, I., Barova, M., Tomov, P.: Hybrid Approach Based on Combination of Backpropagation and Evolutionary Algorithms for Artificial Neural Networks Training by Using Mobile Devices in Distributed Computing Environment. Proceedings of 11th International Conference on Large-Scale Scientific Computations, 425–434, (2017). ISBN 978-3-319-73440-8 DOI 10.1007/978-3-319-73441-5

[128] Zankinski, I., Tomov, P., Balabanov, T.: Alternative Activation Function Derivative in Artificial Neural Networks. Proceedings of XXV Symposium with International Participation - Control of Energy, Industrial and Ecological Systems, 79–81 (2017). ISSN 1313-2237

[129] Zhang, G: An investigation of neural networks for linear time-series forecasting. Computers and Operations Research, vol. 28, no. 12, 1183-1202, (2001). ISSN 0305-0548 DOI 10.1016/S0305-0548(00)00033-2

[130] Zhang, G.: Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing, vol. 50, 159–175, (2003). ISSN 0925-2312 DOI 10.1016/S0925-2312(01)00702-0

[131] Zhang, G., Qi, M.: Neural network forecasting for seasonal and trend time series. European Journal of Operational Research, vol. 160, no. 2, 501–514, (2005). ISSN 0377-2217 DOI 10.1016/j.ejor.2003.08.037

[132] Zhang, G., Kline, D.: Quarterly Time-Series Forecasting With Neural Networks. IEEE Transactions on Neural Networks, vol. 18, no. 6, 1800–1814, (2007). ISSN 1045-9227 DOI 10.1109/TNN.2007.896859

[133] Zhu, J., Liu, C., Gong, J., Wang, D., Song, T.: A Distributed Computing Service for Neural Networks and Its Application to Flood Peak Forecasting. Proceedings of International Conference on Neural Information Processing, vol. 4233, 890–896, (2006). ISBN 978-3-540-46481-5 DOI 10.1007/11893257_98

[134] Zur, R., Jiang, Y., Pesce, L., Drukker, K.: Noise injection for training artificial neural networks: A comparison with weight decay and early stopping. Medical Physics, vol. 36, no. 10, 4810–4818, (2009). ISSN 0094-2405 DOI 10.1118/1.3213517

Приложение А - програмен код

Представеният програмен код представлява самостоятелно приложение за мобилни устройства, работещи с операционната система Android OS. Мобилното приложение комуникира с отдалечен сървър, разработката на който не е част от настоящия дисертационен труд. Програмния код включва графичен потребителски нитерфейс, работа във фонов режим, хибриден алгоритъм за обучение на трислоен перцептрон. Програмният код работи на мобилни устройства с операционна система Android OS, поне версия 11. Приложението зарежда времеви редове от отдалечен сървър. Времевите редове са предимно финансови, но на сървъра може да се зареждат и други видове времеви редове. Времевите редове се декомпозират на тренировъчни примери и се обучава изкуствената невронна мрежа.

ForecastDatabaseHelper.java

```
package eu.veldsoft.vitosha.trade;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.provider.BaseColumns;

/**
 * Database helper class.
 */
class ForecastDatabaseHelper extends SQLiteOpenHelper {
    /**
     * Database integer version.
     */
    public static final int DATABASE_VERSION = 1;
    /**
     * Database file name.
     */
    public static final String DATABASE_NAME = "Forecast.db";
    /**
     * Drop rates table SQL pattern.
     */
    static final String SQL_DELETE_RATES = "DROP TABLE IF EXISTS "
            + RatesColumns.TABLE_NAME;
    /**
     * Drop rates table SQL pattern.
     */
    static final String SQL_DELETE_ANNS = "DROP TABLE IF EXISTS "
            + ANNsColumns.TABLE_NAME;
    /**
     * Create rates table SQL patter.
     */
    private static final String SQL_CREATE_RATES = "CREATE TABLE "
            + RatesColumns.TABLE_NAME + " (" + RatesColumns._ID
            + " INTEGER NOT NULL," + RatesColumns.COLUMN_NAME_SYMBOL
            + " TEXT, " + RatesColumns.COLUMN_NAME_PERIOD
            + " INTEGER, " + RatesColumns.COLUMN_NAME_TIME + " TEXT, "
            + RatesColumns.COLUMN_NAME_OPEN + " TEXT, "
            + RatesColumns.COLUMN_NAME_LOW + " TEXT, "
            + RatesColumns.COLUMN_NAME_HIGH + " TEXT, "
            + RatesColumns.COLUMN_NAME_CLOSE + " TEXT, "
```

```
            + RatesColumns.COLUMN_NAME_VOLUME + " TEXT PRIMARY KEY ("
            + RatesColumns.COLUMN_NAME_SYMBOL + ", "
            + RatesColumns.COLUMN_NAME_PERIOD + "))";

    /**
     * Create ANNs table SQL patter.
     */
    private static final String SQL_CREATE_ANNS = "CREATE TABLE "
            + ANNsColumns.TABLE_NAME + " (" + ANNsColumns._ID
            + " INTEGER PRIMARY KEY," + ANNsColumns.COLUMN_NAME_SYMBOL
            + " TEXT, " + ANNsColumns.COLUMN_NAME_PERIOD
            + " INTEGER, " + ANNsColumns.COLUMN_NAME_NEURONS + " TEXT, "
            + ANNsColumns.COLUMN_NAME_ACTIVITIES
            + " TEXT, " + ANNsColumns.COLUMN_NAME_WEIGHTS
            + " TEXT, FOREIGN KEY (" + ANNsColumns.COLUMN_NAME_SYMBOL
            + ") REFERENCES " + RatesColumns.TABLE_NAME
            + "(" + RatesColumns.COLUMN_NAME_SYMBOL + "), FOREIGN KEY ("
            + ANNsColumns.COLUMN_NAME_PERIOD + ") REFERENCES "
            + RatesColumns.TABLE_NAME + "("
            + RatesColumns.COLUMN_NAME_PERIOD + "))";

    /**
     * Constructor.
     *
     * @param context Context of database helper usage.
     */
    public ForecastDatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    /**
     * {@inheritDoc}
     */
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(SQL_CREATE_RATES);
        db.execSQL(SQL_CREATE_ANNS);
    }

    /**
     * {@inheritDoc}
     */
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL(SQL_DELETE_ANNS);
        db.execSQL(SQL_DELETE_RATES);
        onCreate(db);
    }

    /**
     * Rates table columns description class.
     */
    public static abstract class RatesColumns implements BaseColumns {
        public static final String TABLE_NAME = "rates";
        public static final String COLUMN_NAME_SYMBOL = "symbol";
        public static final String COLUMN_NAME_PERIOD = "period";
        public static final String COLUMN_NAME_TIME = "time";
        public static final String COLUMN_NAME_OPEN = "open";
        public static final String COLUMN_NAME_LOW = "low";
        public static final String COLUMN_NAME_HIGH = "high";
        public static final String COLUMN_NAME_CLOSE = "close";
        public static final String COLUMN_NAME_VOLUME = "volume";
    }

    /**
     * ANN table columns description class.
     */
```

```
    public static abstract class ANNsColumns implements BaseColumns {
        public static final String TABLE_NAME = "anns";
        public static final String COLUMN_NAME_SYMBOL = "symbol";
        public static final String COLUMN_NAME_PERIOD = "period";
        public static final String COLUMN_NAME_NEURONS = "neurons";
        public static final String COLUMN_NAME_ACTIVITIES = "activities";
        public static final String COLUMN_NAME_WEIGHTS = "weights";
    }
}
```

ProgressReportingWallpaperConfigureActivity.java

```
package eu.veldsoft.vitosha.trade;

import android.app.WallpaperManager;
import android.content.ComponentName;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.preference.PreferenceActivity;
import android.preference.PreferenceManager;

/**
 * Options screen.
 */
public class ProgressReportingWallpaperConfigureActivity extends PreferenceActivity {

    /**
     * {@inheritDoc}
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.layout.wallpaper_configure);
    }

    /**
     * {@inheritDoc}
     */
    @Override
    protected void onPause() {
        super.onPause();
        SharedPreferences preferences = PreferenceManager
                .getDefaultSharedPreferences(ProgressReportingWallpaperConfigureActivity.this);

        /*
         * Remove our wallpaper.
         */
        if (preferences.getBoolean("set_wallpaper", false) == false) {
            stopService(new Intent(ProgressReportingWallpaperConfigureActivity.this,
                    ProgressReportingWallpaperService.class));
            startActivity(new Intent(WallpaperManager.ACTION_LIVE_WALLPAPER_CHOOSER));
            ProgressReportingWallpaperConfigureActivity.this.finish();

            return;
        }

        /*
         * Run wallpaper service.
         */
        Intent intent = new Intent(WallpaperManager.ACTION_CHANGE_LIVE_WALLPAPER);
        intent.putExtra(WallpaperManager.EXTRA_LIVE_WALLPAPER_COMPONENT,
                new ComponentName(ProgressReportingWallpaperConfigureActivity.this,
                        ProgressReportingWallpaperService.class));
        startActivity(intent);
        ProgressReportingWallpaperConfigureActivity.this.finish();
    }
```

```
}
```

ProgressReportingWallpaperService.java

```
package eu.veldsoft.vitosha.trade;

import android.content.SharedPreferences;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Rect;
import android.os.Handler;
import android.preference.PreferenceManager;
import android.service.wallpaper.WallpaperService;
import android.view.SurfaceHolder;

import java.util.Calendar;
import java.util.Random;

import eu.veldsoft.vitosha.trade.communication.HttpHelper;
import eu.veldsoft.vitosha.trade.communication.TimePeriod;
import eu.veldsoft.vitosha.trade.dummy.InputData;
import eu.veldsoft.vitosha.trade.engine.Predictor;

/**
 * Background calculation unit.
 */
public class ProgressReportingWallpaperService extends WallpaperService {

    /**
     * Pseudo-random number generator.
     */
    private static final Random PRNG = new Random();

    /**
     * Space between visual spots in pixels.
     */
    private static final int GAP_BETWEEN_PANELS = 10;

    /**
     * Default training time delay.
     */
    private static final long DEFAULT_DELAY = 86400000L;

    //TODO Images should be loaded from a remote image server.

    /**
     * Identifiers for the background resources images to be used as background.
     */
    private static final int[] IMAGES_IDS = {
            R.drawable.vitosha_mountain_dimitar_petarchev_001,
            R.drawable.vitosha_mountain_dimitar_petarchev_002,
            R.drawable.vitosha_mountain_dimitar_petarchev_003,
            R.drawable.vitosha_mountain_dimitar_petarchev_004,
    };

    // TODO Put all colors in the settings dialog.

    /**
     * Panel background color in order to be a part transparent from the real background.
     */
    private static final int PANEL_BACKGROUND_COLOR =
            Color.argb(63, 0, 0, 0);

    /**
```

```
     * Text color to be used in panels.
     */
    private static final int PANEL_TEXT_COLOR =
            Color.argb(95, 255, 255, 255);

    /**
     * Time delay between neural network trainings.
     */
    private static long delay = 0;

    /**
     * Visible surface width.
     */
    private static int screenWidth = 0;

    /**
     * Visible surface height.
     */
    private static int screenHeight = 0;

    /**
     * Wallpaper visibility flag.
     */
    private static boolean visible = false;

    /**
     * List of information panels rectangles information.
     */
    private static Rect[] panels = {new Rect(), new Rect(), new Rect()};

    /**
     * Forecasting object.
     */
    private static Predictor predictor = new Predictor();

    /**
     * Initialize common class members.
     */
    private void initialize() {
        /*
         * Load ANN structure and time series data from the remote server.
         */
        HttpHelper helper = new HttpHelper(PreferenceManager
                .getDefaultSharedPreferences(
                        ProgressReportingWallpaperService.this).
                        getString("server_url", "localhost"));

        if (helper.load() == false) {
            // TODO Use local data if the remote server is not available.
        }

        predictor.initialize();
    }

    /**
     * {@inheritDoc}
     */
    @Override
    public void onCreate() {
        super.onCreate();

        initialize();
    }

    /**
     * {@inheritDoc}
     */
```

```
@Override
public Engine onCreateEngine() {
    return new WallpaperEngine();
}

/**
 * Wallpaper engine class.
 */
private class WallpaperEngine extends Engine {

    /**
     * Thread handler.
     */
    private final Handler handler = new Handler();

    /**
     * Paint object.
     */
    private final Paint paint = new Paint();

    /**
     * Neural network training cycle thread.
     */
    private final Runnable trainer = new Runnable() {
        @Override
        public void run() {
            predictor.predict();
            draw();
            predictor.train();
        }
    };

    /**
     * Constructor without parameters.
     */
    public WallpaperEngine() {
        super();
        handler.post(trainer);
    }

    /**
     * Common drawing procedure.
     */
    private void draw() {
        SurfaceHolder holder = getSurfaceHolder();
        Canvas canvas = null;

        try {
            canvas = holder.lockCanvas();

            if (canvas != null) {
                drawBackground(canvas);
                drawPanels(canvas);
                drawCurrencyPairInfo(canvas);
                drawForecast(canvas);
                drawAnn(canvas);
            }
        } finally {
            if (canvas != null) {
                holder.unlockCanvasAndPost(canvas);
            }
        }

        handler.removeCallbacks(trainer);
        if (visible == true) {
            handler.postDelayed(trainer, delay);
        }
```

```
}

/**
 * Background drawing procedure.
 *
 * @param canvas Canvas object for background.
 */
private void drawBackground(Canvas canvas) {
    // TODO Images should be loaded from an image server.
    /*
     * Change picture according the day in the year.
     */
    Bitmap image = BitmapFactory.decodeResource(
            ProgressReportingWallpaperService.this.getResources(),
            IMAGES_IDS[Calendar.getInstance().
                    get(Calendar.DAY_OF_YEAR) % IMAGES_IDS.length]);

    /*
     * Select random top-left corner for image clip.
     */
    int left = PRNG.nextInt(image.getWidth() - screenWidth);
    int top = PRNG.nextInt(image.getHeight() - screenHeight);

    /*
     * Clip part of the image.
     */
    canvas.drawBitmap(image, new Rect(left, top,
                    left + screenWidth - 1,
                    top + screenHeight - 1),
            new Rect(0, 0, screenWidth - 1,
                    screenHeight - 1), null);
}

/**
 * Panels drawing procedure.
 *
 * @param canvas Canvas object for panels drawing.
 */
private void drawPanels(Canvas canvas) {
    /*
     * Panels.
     */
    paint.setColor(PANEL_BACKGROUND_COLOR);
    for (Rect rectangle : panels) {
        canvas.drawRect(rectangle, paint);
    }
}

/**
 * Currency pair info drawing procedure.
 *
 * @param canvas Canvas object for currency pair info drawing.
 */
private void drawCurrencyPairInfo(Canvas canvas) {
    /*
     * Time series info.
     */
    int textSize = (panels[0].bottom - panels[0].top) / 5;
    paint.setTextSize(textSize);
    paint.setColor(PANEL_TEXT_COLOR);
    canvas.drawText("" + InputData.SYMBOL,
            GAP_BETWEEN_PANELS + panels[0].left,
            GAP_BETWEEN_PANELS + panels[0].top + textSize, paint);
    canvas.drawText("" + TimePeriod.value(InputData.PERIOD),
            GAP_BETWEEN_PANELS + panels[0].left,
            GAP_BETWEEN_PANELS + panels[0].top + 2 * textSize, paint);
}
```

```
/**
 * Forecast drawing procedure.
 *
 * @param canvas Canvas object for forecast drawing.
 */
private void drawForecast(Canvas canvas) {
    int width = panels[1].right - panels[1].left;
    int height = panels[1].bottom - panels[1].top;
    int stride = width;

    int[] pixels = new int[width * height];
    predictor.drawForecast(pixels, width, height);
    Bitmap bitmap = Bitmap.createBitmap(pixels, 0, stride, width, height, Bitmap.Config.ARGB_
    canvas.drawBitmap(bitmap, new Rect(0,0,width,height), panels[1], paint);
}

/**
 * Neural network drawing procedure.
 *
 * @param canvas Canvas object for neural network drawing.
 */
private void drawAnn(Canvas canvas) {
    int width = panels[2].right - panels[2].left;
    int height = panels[2].bottom - panels[2].top;
    int stride = width;

    int[] pixels = new int[width * height];
    predictor.drawAnn(pixels, width, height);
    Bitmap bitmap = Bitmap.createBitmap(pixels, 0, stride, width, height, Bitmap.Config.ARGB_
    canvas.drawBitmap(bitmap, new Rect(0,0,width,height), panels[2], paint);
}

/**
 * {@inheritDoc}
 */
@Override
public void onVisibilityChanged(boolean visible) {
    ProgressReportingWallpaperService.visible = visible;

    /*
     * Do calculations only if the wallpaper is visible.
     */
    if (visible == true) {
        handler.post(trainer);
    } else {
        handler.removeCallbacks(trainer);
    }
}

/**
 * {@inheritDoc}
 */
@Override
public void onSurfaceDestroyed(SurfaceHolder holder) {
    super.onSurfaceDestroyed(holder);
    ProgressReportingWallpaperService.visible = false;
    handler.removeCallbacks(trainer);
}

/**
 * {@inheritDoc}
 */
@Override
public void onSurfaceChanged(SurfaceHolder holder,
                             int format, int width, int height) {
    super.onSurfaceChanged(holder, format, width, height);
```

```
screenWidth = width;
screenHeight = height;

SharedPreferences preferences = PreferenceManager
        .getDefaultSharedPreferences(
                ProgressReportingWallpaperService.this);

int panelsSideSize = Integer.parseInt(
        preferences.getString("sizing", "100"));

switch (preferences.getString("positioning", "0 0")) {
    case "lt":
        panels[0].left = GAP_BETWEEN_PANELS;
        panels[0].top = GAP_BETWEEN_PANELS;
        panels[0].right = GAP_BETWEEN_PANELS + panelsSideSize;
        panels[0].bottom = panelsSideSize + GAP_BETWEEN_PANELS;

        panels[1].left = GAP_BETWEEN_PANELS;
        panels[1].top = 2 * GAP_BETWEEN_PANELS + panelsSideSize;
        panels[1].right = GAP_BETWEEN_PANELS + panelsSideSize;
        panels[1].bottom = 2 * GAP_BETWEEN_PANELS + 2 * panelsSideSize;

        panels[2].left = GAP_BETWEEN_PANELS;
        panels[2].top = 3 * GAP_BETWEEN_PANELS + 2 * panelsSideSize;
        panels[2].right = GAP_BETWEEN_PANELS + panelsSideSize;
        panels[2].bottom = 3 * GAP_BETWEEN_PANELS + 3 * panelsSideSize;
        break;
    case "ct":
        panels[0].left = width / 2 - panelsSideSize / 2;
        panels[0].top = GAP_BETWEEN_PANELS;
        panels[0].right = width / 2 + panelsSideSize / 2;
        panels[0].bottom = panelsSideSize + GAP_BETWEEN_PANELS;

        panels[1].left = width / 2 - panelsSideSize / 2;
        panels[1].top = 2 * GAP_BETWEEN_PANELS + panelsSideSize;
        panels[1].right = width / 2 + panelsSideSize / 2;
        panels[1].bottom = 2 * GAP_BETWEEN_PANELS + 2 * panelsSideSize;

        panels[2].left = width / 2 - panelsSideSize / 2;
        panels[2].top = 3 * GAP_BETWEEN_PANELS + 2 * panelsSideSize;
        panels[2].right = width / 2 + panelsSideSize / 2;
        panels[2].bottom = 3 * GAP_BETWEEN_PANELS + 3 * panelsSideSize;
        break;
    case "rt":
        panels[0].left = width - panelsSideSize - GAP_BETWEEN_PANELS;
        panels[0].top = GAP_BETWEEN_PANELS;
        panels[0].right = width - GAP_BETWEEN_PANELS;
        panels[0].bottom = panelsSideSize + GAP_BETWEEN_PANELS;

        panels[1].left = width - panelsSideSize - GAP_BETWEEN_PANELS;
        panels[1].top = 2 * GAP_BETWEEN_PANELS + panelsSideSize;
        panels[1].right = width - GAP_BETWEEN_PANELS;
        panels[1].bottom = 2 * GAP_BETWEEN_PANELS + 2 * panelsSideSize;

        panels[2].left = width - panelsSideSize - GAP_BETWEEN_PANELS;
        panels[2].top = 3 * GAP_BETWEEN_PANELS + 2 * panelsSideSize;
        panels[2].right = width - GAP_BETWEEN_PANELS;
        panels[2].bottom = 3 * GAP_BETWEEN_PANELS + 3 * panelsSideSize;
        break;
    case "lc":
        panels[0].left = GAP_BETWEEN_PANELS;
        panels[0].top = height / 2 - panelsSideSize / 2 -
                GAP_BETWEEN_PANELS - panelsSideSize;
        panels[0].right = GAP_BETWEEN_PANELS + panelsSideSize;
        panels[0].bottom = height / 2 - panelsSideSize / 2 -
                GAP_BETWEEN_PANELS;
```

```
            panels[1].left = GAP_BETWEEN_PANELS;
            panels[1].top = height / 2 - panelsSideSize / 2;
            panels[1].right = GAP_BETWEEN_PANELS + panelsSideSize;
            panels[1].bottom = height / 2 + panelsSideSize / 2;

            panels[2].left = GAP_BETWEEN_PANELS;
            panels[2].top = height / 2 + panelsSideSize / 2 +
                    GAP_BETWEEN_PANELS;
            panels[2].right = GAP_BETWEEN_PANELS + panelsSideSize;
            panels[2].bottom = height / 2 + panelsSideSize / 2 +
                    GAP_BETWEEN_PANELS + panelsSideSize;
            break;
        case "cc":
            panels[0].left = width / 2 - panelsSideSize / 2;
            panels[0].top = height / 2 - panelsSideSize / 2 -
                    GAP_BETWEEN_PANELS - panelsSideSize;
            panels[0].right = width / 2 + panelsSideSize / 2;
            panels[0].bottom = height / 2 - panelsSideSize / 2 -
                    GAP_BETWEEN_PANELS;

            panels[1].left = width / 2 - panelsSideSize / 2;
            panels[1].top = height / 2 - panelsSideSize / 2;
            panels[1].right = width / 2 + panelsSideSize / 2;
            panels[1].bottom = height / 2 + panelsSideSize / 2;

            panels[2].left = width / 2 - panelsSideSize / 2;
            panels[2].top = height / 2 + panelsSideSize / 2 +
                    GAP_BETWEEN_PANELS;
            panels[2].right = width / 2 + panelsSideSize / 2;
            panels[2].bottom = height / 2 + panelsSideSize / 2 +
                    GAP_BETWEEN_PANELS + panelsSideSize;
            break;
        case "rc":
            panels[0].left = width - panelsSideSize - GAP_BETWEEN_PANELS;
            panels[0].top = height / 2 - panelsSideSize / 2 -
                    GAP_BETWEEN_PANELS - panelsSideSize;
            panels[0].right = width - GAP_BETWEEN_PANELS;
            panels[0].bottom = height / 2 - panelsSideSize / 2 -
                    GAP_BETWEEN_PANELS;

            panels[1].left = width - panelsSideSize - GAP_BETWEEN_PANELS;
            panels[1].top = height / 2 - panelsSideSize / 2;
            panels[1].right = width - GAP_BETWEEN_PANELS;
            panels[1].bottom = height / 2 + panelsSideSize / 2;

            panels[2].left = width - panelsSideSize - GAP_BETWEEN_PANELS;
            panels[2].top = height / 2 + panelsSideSize / 2 +
                    GAP_BETWEEN_PANELS;
            panels[2].right = width - GAP_BETWEEN_PANELS;
            panels[2].bottom = height / 2 + panelsSideSize / 2 +
                    GAP_BETWEEN_PANELS + panelsSideSize;
            break;
        case "lb":
            panels[0].left = GAP_BETWEEN_PANELS;
            panels[0].top = height - 3 * GAP_BETWEEN_PANELS - 3 *
                    panelsSideSize;
            panels[0].right = GAP_BETWEEN_PANELS + panelsSideSize;
            panels[0].bottom = height - 3 * GAP_BETWEEN_PANELS - 2 *
                    panelsSideSize;

            panels[1].left = GAP_BETWEEN_PANELS;
            panels[1].top = height - 2 * GAP_BETWEEN_PANELS - 2 *
                    panelsSideSize;
            panels[1].right = GAP_BETWEEN_PANELS + panelsSideSize;
            panels[1].bottom = height - 2 * GAP_BETWEEN_PANELS -
                    panelsSideSize;
```

```
                    panels[2].left = GAP_BETWEEN_PANELS;
                    panels[2].top = height - GAP_BETWEEN_PANELS - panelsSideSize;
                    panels[2].right = GAP_BETWEEN_PANELS + panelsSideSize;
                    panels[2].bottom = height - GAP_BETWEEN_PANELS;
                    break;
                case "cb":
                    panels[0].left = width / 2 - panelsSideSize / 2;
                    panels[0].top = height - 3 * GAP_BETWEEN_PANELS - 3 *
                            panelsSideSize;
                    panels[0].right = width / 2 + panelsSideSize / 2;
                    panels[0].bottom = height - 3 * GAP_BETWEEN_PANELS - 2 *
                            panelsSideSize;

                    panels[1].left = width / 2 - panelsSideSize / 2;
                    panels[1].top = height - 2 * GAP_BETWEEN_PANELS - 2 *
                            panelsSideSize;
                    panels[1].right = width / 2 + panelsSideSize / 2;
                    panels[1].bottom = height - 2 * GAP_BETWEEN_PANELS -
                            panelsSideSize;

                    panels[2].left = width / 2 - panelsSideSize / 2;
                    panels[2].top = height - GAP_BETWEEN_PANELS - panelsSideSize;
                    panels[2].right = width / 2 + panelsSideSize / 2;
                    panels[2].bottom = height - GAP_BETWEEN_PANELS;
                    break;
                case "rb":
                    panels[0].left = width - panelsSideSize - GAP_BETWEEN_PANELS;
                    panels[0].top = height - 3 * GAP_BETWEEN_PANELS - 3 *
                            panelsSideSize;
                    panels[0].right = width - GAP_BETWEEN_PANELS;
                    panels[0].bottom = height - 3 * GAP_BETWEEN_PANELS - 2 *
                            panelsSideSize;

                    panels[1].left = width - panelsSideSize - GAP_BETWEEN_PANELS;
                    panels[1].top = height - 2 * GAP_BETWEEN_PANELS - 2 *
                            panelsSideSize;
                    panels[1].right = width - GAP_BETWEEN_PANELS;
                    panels[1].bottom = height - 2 * GAP_BETWEEN_PANELS -
                            panelsSideSize;

                    panels[2].left = width - panelsSideSize - GAP_BETWEEN_PANELS;
                    panels[2].top = height - GAP_BETWEEN_PANELS - panelsSideSize;
                    panels[2].right = width - GAP_BETWEEN_PANELS;
                    panels[2].bottom = height - GAP_BETWEEN_PANELS;
                    break;
                default:
                    break;
            }

            delay = Long.parseLong(preferences.getString("loading",
                    "" + DEFAULT_DELAY));
        }
    }
}
```

VotingWidget.java

```
package eu.veldsoft.vitosha.trade;

import android.appwidget.AppWidgetManager;
import android.appwidget.AppWidgetProvider;
import android.content.Context;
import android.widget.RemoteViews;

import eu.veldsoft.vitosha.trade.dummy.InputData;
```

```
/**
 * User voting widget.
 */
public class VotingWidget extends AppWidgetProvider {

    /**
     *
     * @param context
     * @param appWidgetManager
     * @param appWidgetId
     */
    static void updateAppWidget(Context context, AppWidgetManager appWidgetManager,
                                int appWidgetId) {

        CharSequence widgetText = VotingWidgetConfigureActivity.loadTitlePref(context, appWidgetId);

        RemoteViews views = new RemoteViews(context.getPackageName(), R.layout.voting_widget);
        views.setTextViewText(R.id.appwidget_text, widgetText);

        //TODO Information should be taken from other source.
        views.setTextViewText(R.id.symbol_ticker, InputData.SYMBOL);
        views.setTextViewText(R.id.current_value, ""+InputData.OPEN[InputData.OPEN.length-1]);

        appWidgetManager.updateAppWidget(appWidgetId, views);
    }

    /**
     * {@inheritDoc}
     */
    @Override
    public void onUpdate(Context context, AppWidgetManager appWidgetManager, int[] appWidgetIds) {
        for (int appWidgetId : appWidgetIds) {
            updateAppWidget(context, appWidgetManager, appWidgetId);
        }
    }

    /**
     * {@inheritDoc}
     */
    @Override
    public void onDeleted(Context context, int[] appWidgetIds) {
        for (int appWidgetId : appWidgetIds) {
            VotingWidgetConfigureActivity.deleteTitlePref(context, appWidgetId);
        }
    }

    /**
     * {@inheritDoc}
     */
    @Override
    public void onEnabled(Context context) {
    }

    /**
     * {@inheritDoc}
     */
    @Override
    public void onDisabled(Context context) {
    }
}
```

VotingWidgetConfigureActivity.java

```
package eu.veldsoft.vitosha.trade;

import android.app.Activity;
import android.appwidget.AppWidgetManager;
```

```
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;

/**
 * Voting widget preference activity.
 */
public class VotingWidgetConfigureActivity extends Activity {

    /** */
    private static final String PREFS_NAME = "eu.veldsoft.vitosha.trade.VotingWidget";

    /** */
    private static final String PREF_PREFIX_KEY = "appwidget_";

    /** */
    int mAppWidgetId = AppWidgetManager.INVALID_APPWIDGET_ID;

    /** */
    EditText mAppWidgetText;

    /** */
    View.OnClickListener mOnClickListener = new View.OnClickListener() {
        public void onClick(View v) {
            final Context context = VotingWidgetConfigureActivity.this;

            String widgetText = mAppWidgetText.getText().toString();
            saveTitlePref(context, mAppWidgetId, widgetText);

            AppWidgetManager appWidgetManager = AppWidgetManager.getInstance(context);
            VotingWidget.updateAppWidget(context, appWidgetManager, mAppWidgetId);

            Intent resultValue = new Intent();
            resultValue.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, mAppWidgetId);
            setResult(RESULT_OK, resultValue);
            finish();
        }
    };

    /** */
    public VotingWidgetConfigureActivity() {
        super();
    }

    /**
     *
     * @param context
     * @param appWidgetId
     * @param text
     */
    static void saveTitlePref(Context context, int appWidgetId, String text) {
        SharedPreferences.Editor prefs = context.getSharedPreferences(PREFS_NAME, 0).edit();
        prefs.putString(PREF_PREFIX_KEY + appWidgetId, text);
        prefs.apply();
    }

    /**
     *
     * @param context
     * @param appWidgetId
     * @return
     */
    static String loadTitlePref(Context context, int appWidgetId) {
        SharedPreferences prefs = context.getSharedPreferences(PREFS_NAME, 0);
```

```
        String titleValue = prefs.getString(PREF_PREFIX_KEY + appWidgetId, null);
        if (titleValue != null) {
            return titleValue;
        } else {
            return "";
        }
    }

    /**
     *
     * @param context
     * @param appWidgetId
     */
    static void deleteTitlePref(Context context, int appWidgetId) {
        SharedPreferences.Editor prefs = context.getSharedPreferences(PREFS_NAME, 0).edit();
        prefs.remove(PREF_PREFIX_KEY + appWidgetId);
        prefs.apply();
    }

    /**
     * {@inheritDoc}
     */
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);

        setResult(RESULT_CANCELED);

        setContentView(R.layout.voting_widget_configure);
        mAppWidgetText = (EditText) findViewById(R.id.appwidget_text);
        findViewById(R.id.add_button).setOnClickListener(mOnClickListener);

        Intent intent = getIntent();
        Bundle extras = intent.getExtras();
        if (extras != null) {
            mAppWidgetId = extras.getInt(
                    AppWidgetManager.EXTRA_APPWIDGET_ID, AppWidgetManager.INVALID_APPWIDGET_ID);
        }

        if (mAppWidgetId == AppWidgetManager.INVALID_APPWIDGET_ID) {
            finish();
            return;
        }

        mAppWidgetText.setText(loadTitlePref(VotingWidgetConfigureActivity.this, mAppWidgetId));
    }
}
```

ConsolePredictor.java

```
package eu.veldsoft.vitosha.trade;

import eu.veldsoft.vitosha.trade.dummy.InputData;
import eu.veldsoft.vitosha.trade.engine.Predictor;

/**
 * Single entry point class for command line application interface.
 */
public class ConsolePredictor {
    public static void main(String[] args) {
        Predictor predictor = new Predictor();
        predictor.initialize();
        predictor.train();
        predictor.predict();
    }
}
```

HttpHelper.java

```
package eu.veldsoft.vitosha.trade.communication;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.util.List;

import cz.msebera.android.httpclient.HttpResponse;
import cz.msebera.android.httpclient.NameValuePair;
import cz.msebera.android.httpclient.client.ClientProtocolException;
import cz.msebera.android.httpclient.client.HttpClient;
import cz.msebera.android.httpclient.client.entity.UrlEncodedFormEntity;
import cz.msebera.android.httpclient.client.methods.HttpPost;
import cz.msebera.android.httpclient.impl.client.DefaultHttpClient;
import cz.msebera.android.httpclient.message.BasicNameValuePair;
import cz.msebera.android.httpclient.util.EntityUtils;
import eu.veldsoft.vitosha.trade.dummy.InputData;

/**
 * It is used for HTTP communication with the remote server.
 */
public class HttpHelper {
    /**
     * Number of ANN found as response of the request.
     */
    private static final String JSON_SIZE_KEY = "size";
    /**
     * Time series symbol ticker.
     */
    private static final String JSON_SYMBOL_KEY = "symbol";
    /**
     * Time series period as integer number of minutes.
     */
    private static final String JSON_PERIOD_KEY = "period";
    /**
     * Fitness value of the ANN.
     */
    private static final String JSON_FITNESS_KEY = "fitness";
    /**
     * Array with neurons flags.
     */
    private static final String JSON_FLAGS_KEY = "flags";
    /**
     * Array with ANN weights.
     */
    private static final String JSON_WEIGHTS_KEY = "weights";
    /**
     * Array with ANN connections activities.
     */
    private static final String JSON_ACTIVITIES_KEY = "activities";
    /**
     * Number of training examples.
     */
    private static final String JSON_NUMBER_OF_EXAMPLES_KEY = "numberOfExamples";
    /**
     * Time array.
     */
    private static final String JSON_TIME_KEY = "time";
    /**
     * Open array.
     */
    private static final String JSON_OPEN_KEY = "open";
    /**
```

```
     * Low array.
     */
    private static final String JSON_LOW_KEY = "low";
    /**
     * High array.
     */
    private static final String JSON_HIGH_KEY = "high";
    /**
     * Close array.
     */
    private static final String JSON_CLOSE_KEY = "close";
    /**
     * Volume array.
     */
    private static final String JSON_VOLUME_KEY = "volume";
    /**
     * Load random ANN remote server script name.
     */
    private final String LOAD_RANDOM_ANN_SCRIPT = "logic/json_load_random_ann.php";
    /**
     * Loading training set for particular ticker and time period.
     */
    private final String LOAD_TRAINING_SET_SCRIPT = "logic/json_load_training_set.php";
    /**
     * Report retrained ANN remote server script name.
     */
    private final String SAVE_RETRAINED_ANN_SCRIPT = "logic/save_retrained_ann.php";
    /**
     * Number of neurons for the ANN.
     */
    private final String JSON_NUMBER_OF_NEURONS_KEY = "numberOfNeurons";
    /**
     * Remote server URl address.
     */
    private final String url;

    /**
     * Constructor with all parameters needed.
     *
     * @param url Remote server URL address.
     */
    public HttpHelper(String url) {
        this.url = url;
    }

    /*
     * Load remote data into input data structure.
     *
     * @return True if the loading was successful, false otherwise.
     */
    public boolean load() {
        String symbol = InputData.SYMBOL;
        int period = InputData.PERIOD;
        int[] flags = InputData.NEURONS;
        double[][] weights = InputData.WEIGHTS;
        double[][] activities = InputData.ACTIVITIES;
        long[] time = InputData.TIME;
        double[] open = InputData.OPEN;
        double[] low = InputData.LOW;
        double[] high = InputData.HIGH;
        double[] close = InputData.HIGH;
        double[] volume = InputData.VOLUME;

        HttpClient client = new DefaultHttpClient();
        client.getParams().setParameter("http.protocol.content-charset", "UTF-8");

        /*
```

```
         * Load randomly selected ANN.
         */
        HttpPost post = new HttpPost("http://" + url.trim() + "/" + LOAD_RANDOM_ANN_SCRIPT);

        try {
            HttpResponse response = client.execute(post);

            JSONObject result = new JSONObject(EntityUtils.toString(response.getEntity(), "UTF-8"));

            int size = result.getInt(JSON_SIZE_KEY);

            /*
             * If there is no ANN on the server side nothing can be loaded.
             */
            if (size <= 0) {
                return false;
            }

            /*
             * Extract JSON from HTTP response.
             */
            symbol = result.getString(JSON_SYMBOL_KEY);

            period = result.getInt(JSON_PERIOD_KEY);

            double fitness = result.getDouble(JSON_FITNESS_KEY);

            int numberOfNeurons = result.getInt(JSON_NUMBER_OF_NEURONS_KEY);

            flags = new int[numberOfNeurons];
            JSONArray array1 = result.getJSONArray(JSON_FLAGS_KEY);
            for (int i = 0; i < array1.length(); i++) {
                flags[i] = array1.getInt(i);
            }

            //TODO Matrix transpose is possible.
            weights = new double[numberOfNeurons][numberOfNeurons];
            JSONArray array2 = result.getJSONArray(JSON_WEIGHTS_KEY);
            for (int j = 0; j < array2.length(); j++) {
                JSONArray array3 = array2.getJSONArray(j);
                for (int i = 0; i < array3.length(); i++) {
                    weights[i][j] = array3.getDouble(i);
                }
            }

            //TODO Matrix transpose is possible.
            activities = new double[numberOfNeurons][numberOfNeurons];
            JSONArray array4 = result.getJSONArray(JSON_ACTIVITIES_KEY);
            for (int j = 0; j < array4.length(); j++) {
                JSONArray array5 = array4.getJSONArray(j);
                for (int i = 0; i < array5.length(); i++) {
                    activities[i][j] = array5.getDouble(i);
                }
            }
        } catch (ClientProtocolException exception) {
            return false;
        } catch (IOException exception) {
            return false;
        } catch (JSONException exception) {
            return false;
        } catch (Exception exception) {
            return false;
        }

        /*
         * Load training set for the selected ANN.
         */
```

```
post = new HttpPost("http://" + url.trim() + "/" + LOAD_TRAINING_SET_SCRIPT);
List<NameValuePair> pairs = new ArrayList<NameValuePair>();
pairs.add(new BasicNameValuePair("symbol", symbol));
pairs.add(new BasicNameValuePair("period", "" + period));
try {
    post.setEntity(new UrlEncodedFormEntity(pairs));
} catch (UnsupportedEncodingException e) {
    return false;
}

try {
    HttpResponse response = client.execute(post);

    JSONObject result = new JSONObject(EntityUtils.toString(response.getEntity(), "UTF-8"));

    int size = result.getInt(JSON_NUMBER_OF_EXAMPLES_KEY);

    /*
     * If there is no ANN training set on the server side nothing can be loaded.
     */
    if (size <= 0) {
        return false;
    }

    time = new long[size];
    JSONArray array1 = result.getJSONArray(JSON_TIME_KEY);
    for (int i = 0; i < array1.length(); i++) {
        time[i] = array1.getLong(i);
    }

    open = new double[size];
    JSONArray array2 = result.getJSONArray(JSON_OPEN_KEY);
    for (int i = 0; i < array2.length(); i++) {
        open[i] = array2.getDouble(i);
    }

    low = new double[size];
    JSONArray array3 = result.getJSONArray(JSON_LOW_KEY);
    for (int i = 0; i < array3.length(); i++) {
        low[i] = array3.getDouble(i);
    }

    high = new double[size];
    JSONArray array4 = result.getJSONArray(JSON_HIGH_KEY);
    for (int i = 0; i < array4.length(); i++) {
        high[i] = array4.getDouble(i);
    }

    close = new double[size];
    JSONArray array5 = result.getJSONArray(JSON_CLOSE_KEY);
    for (int i = 0; i < array5.length(); i++) {
        close[i] = array5.getDouble(i);
    }

    volume = new double[size];
    JSONArray array6 = result.getJSONArray(JSON_VOLUME_KEY);
    for (int i = 0; i < array6.length(); i++) {
        volume[i] = array6.getDouble(i);
    }
} catch (ClientProtocolException exception) {
    return false;
} catch (IOException exception) {
    return false;
} catch (JSONException exception) {
    return false;
} catch (Exception exception) {
    return false;
```

```
        }

        /*
         * Load data in the global data structure.
         */
        InputData.SYMBOL = symbol;
        InputData.PERIOD = period;
        InputData.NEURONS = flags;
        InputData.WEIGHTS = weights;
        InputData.ACTIVITIES = activities;
        InputData.TIME = time;
        InputData.OPEN = open;
        InputData.LOW = low;
        InputData.HIGH = high;
        InputData.CLOSE = close;
        InputData.VOLUME = volume;
        InputData.RATES = new double[][]{InputData.OPEN, InputData.LOW, InputData.HIGH, InputData.CLO

        return true;
    }

    /**
     * Store calculated ANN weights on the remote web server.
     *
     * @return True if the saving was successful, false otherwise.
     */
    public boolean store() {
        HttpClient client = new DefaultHttpClient();
        client.getParams().setParameter("http.protocol.content-charset", "UTF-8");

        /*
         * Store retrained ANN.
         */
        HttpPost post = new HttpPost("http://" + url.trim() + "/" + SAVE_RETRAINED_ANN_SCRIPT);
        List<NameValuePair> pairs = new ArrayList<NameValuePair>();

        pairs.add(new BasicNameValuePair("symbol", InputData.SYMBOL));
        pairs.add(new BasicNameValuePair("period", "" + InputData.PERIOD));
        pairs.add(new BasicNameValuePair("fitness", "" + InputData.FITNESS));
        pairs.add(new BasicNameValuePair("number_of_neurons", "" + InputData.NEURONS.length));

        String flags = "";
        for (int i = 0; i < InputData.NEURONS.length; i++) {
            flags += InputData.NEURONS[i] + " ";
        }
        pairs.add(new BasicNameValuePair("flags", "" + flags.trim()));

        //TODO Matrix transpose is possible.
        String weights = "";
        for (int j = 0; j < InputData.NEURONS.length; j++) {
            for (int i = 0; i < InputData.NEURONS.length; i++) {
                weights += InputData.WEIGHTS[i][j] + " ";
            }
            weights = weights.trim() + "\r\n";
        }
        pairs.add(new BasicNameValuePair("weights", "" + weights.trim()));

        //TODO Matrix transpose is possible.
        String activites = "";
        for (int j = 0; j < InputData.NEURONS.length; j++) {
            for (int i = 0; i < InputData.NEURONS.length; i++) {
                activites += InputData.ACTIVITIES[i][j] + " ";
            }
            activites = activites.trim() + "\r\n";
        }
        pairs.add(new BasicNameValuePair("activites", "" + activites.trim()));
```

```
        try {
            post.setEntity(new UrlEncodedFormEntity(pairs));
            client.execute(post);
        } catch (ClientProtocolException exception) {
            return false;
        } catch (UnsupportedEncodingException exception) {
            return false;
        } catch (IOException exception) {
            return false;
        }

        return true;
    }
}
```

NeuronType.java

```
package eu.veldsoft.vitosha.trade.communication;

/**
 * Numeric constants for neurons type.
 */
public enum NeuronType {

    /**
     * Regular neuron flag.
     */
    REGULAR(0x00),

    /**
     * Bias neuron flag.
     */
    BIAS(0x01),

    /**
     * Input neuron flag.
     */
    INPUT(0x02),

    /**
     * Input and bias neuron flag.
     */
    INPUT_BIAS(0x03),

    /**
     * Output neuron flag.
     */
    OUTPUT(0x04),

    /**
     * Output and bias neuron flag.
     */
    OUTPUT_BIAS(0x05),

    /**
     * Output and input neuron flag.
     */
    OUTPUT_INPUT(0x06),

    /**
     * Output, input and bias neuron flag.
     */
    OUTPUT_INPUT_BIAS(0x07);

    /*
     * Numeric value representation.
     */
```

```
    private final int value;

    /**
     * Constructor with all parameters.
     *
     * @param value
     */
    private NeuronType(int value) {
        this.value = value;
    }

    /**
     * Value factory function.
     *
     * @param type Numerical type representation.
     * @return Corresponding enumeration or regular if there is no correspondence.
     */
    public static NeuronType valueOf(int type) {
        for (NeuronType item : NeuronType.values()) {
            if (item.value() == type) {
                return item;
            }
        }

        return REGULAR;
    }

    /**
     * Value getter.
     *
     * @return Numeric representation of the type.
     */
    public int value() {
        return value;
    }
}
```

TimePeriod.java

```
package eu.veldsoft.vitosha.trade.communication;

/**
 * Time series fixed time periods.
 */
public enum TimePeriod {

    /**
     * No time period at all.
     */
    NONE(0, ""),

    /**
     * One minute.
     */
    M1(1, "M1"),

    /**
     * Five minutes.
     */
    M5(5, "M5"),

    /**
     * Fifteen minutes.
     */
    M15(15, "M15"),

    /**
```

```
     * Thirty minutes.
     */
    M30(30, "M30"),

    /**
     * One hour.
     */
    H1(60, "H1"),

    /**
     * Four hours.
     */
    H4(240, "H4"),

    /**
     * One day.
     */
    D1(1440, "D1"),

    /**
     * One week.
     */
    W1(10080, "W1"),

    /**
     * One month.
     */
    MN1(43200, "MN1");

    /**
     * Time period as number of minutes.
     */
    private int minutes;

    /**
     * Time period as text description.
     */
    private String name;

    /**
     * Constructor with all parameters.
     *
     * @param minutes Minutes as numbers.
     * @param name    Interval as name.
     */
    private TimePeriod(int minutes, String name) {
        this.minutes = minutes;
        this.name = name;
    }

    /**
     * Factory function for object reference from time interval.
     *
     * @param minutes Time interval in minutes.
     * @return Time period as object.
     * @throws RuntimeException Rise exception if there is no such time interval in minutes.
     */
    public static TimePeriod value(int minutes) throws RuntimeException {
        for (TimePeriod item : TimePeriod.values()) {
            if (item.minutes == minutes) {
                return item;
            }
        }

        // TODO Report exception.

        return NONE;
```

```
    }

    /**
     * Factory function for object reference from form interval name.
     *
     * @param name Time period name.
     * @return Time period as object.
     * @throws RuntimeException Rise exception if there is no such time interval in minutes.
     */
    public static TimePeriod value(String name) throws RuntimeException {
        for (TimePeriod item : TimePeriod.values()) {
            if (name.equals(item.name) == true) {
                return item;
            }
        }

        // TODO Report exception.

        return NONE;
    }

    /**
     * {@inheritDoc}
     */
    @Override
    public String toString() {
        return name;
    }
}
```

ApacheOptimizer.java

```
package eu.veldsoft.vitosha.trade.engine;

import org.apache.commons.math3.genetics.Chromosome;
import org.apache.commons.math3.genetics.ElitisticListPopulation;
import org.apache.commons.math3.genetics.FixedElapsedTime;
import org.apache.commons.math3.genetics.GeneticAlgorithm;
import org.apache.commons.math3.genetics.Population;
import org.apache.commons.math3.genetics.TournamentSelection;
import org.apache.commons.math3.genetics.UniformBinaryMutation;
import org.apache.commons.math3.genetics.UniformCrossover;
import org.apache.commons.math3.genetics.WeightsChromosome;
import org.encog.neural.networks.BasicNetwork;
import org.encog.neural.networks.training.propagation.Propagation;

import java.util.LinkedList;
import java.util.List;

/**
 * Apache Common Math Genetic Algorithms based optimizer.
 */
public class ApacheOptimizer implements Optimizer {
    /**
     *
     */
    private final long optimizationTimeout;

    /**
     *
     */
    private final BasicNetwork network;

    /**
     *
     */
    private final Propagation train;
```

```
/**
 *
 */
private final int populationSize;

/**
 *
 */
private final int tournamentArity;

/**
 *
 */
private final double crossoverRate;

/**
 *
 */
private final double mutationRate;

/**
 *
 */
private final double elitismRate;

/**
 * @param optimizationTimeout
 * @param network
 * @param train
 * @param populationSize
 * @param tournamentArity
 * @param crossoverRate
 * @param mutationRate
 * @param elitismRate
 */
public ApacheOptimizer(long optimizationTimeout, BasicNetwork network, Propagation train, int pop
    this.optimizationTimeout = optimizationTimeout;
    this.network = network;
    this.train = train;
    this.populationSize = populationSize;
    this.tournamentArity = tournamentArity;
    this.crossoverRate = crossoverRate;
    this.mutationRate = mutationRate;
    this.elitismRate = elitismRate;
}

/**
 * {@inheritDoc}
 */
@Override
public List<Double> optimize(List<Double> weights) {

    /*
     * Generate population.
     */
    List<Chromosome> chromosomes = new LinkedList<Chromosome>();
    for (int i = 0; i < populationSize; i++) {
        chromosomes.add(new WeightsChromosome(weights, true, network, train));
    }
    Population initial = new ElitisticListPopulation(chromosomes,
            2 * chromosomes.size(), elitismRate);

    /*
     * Initialize genetic algorithm.
     */
    GeneticAlgorithm algorithm = new GeneticAlgorithm(
```

```
                new UniformCrossover<WeightsChromosome>(0.5),
                crossoverRate, new UniformBinaryMutation(),
                mutationRate, new TournamentSelection(tournamentArity));

        /*
         * Run optimization.
         */
        Population optimized = algorithm.evolve(initial,
                new FixedElapsedTime(optimizationTimeout));

        /*
         * Obtain result.
         */
        weights = ((WeightsChromosome)
                optimized.getFittestChromosome()).
                getRepresentation();

        return weights;
    }
}
```

MoeaOptimizer.java

```
package eu.veldsoft.vitosha.trade.engine;

import org.encog.neural.networks.BasicNetwork;
import org.encog.neural.networks.training.propagation.Propagation;
import org.moeaframework.algorithm.AbstractAlgorithm;
import org.moeaframework.algorithm.single.AggregateObjectiveComparator;
import org.moeaframework.algorithm.single.DifferentialEvolution;
import org.moeaframework.algorithm.single.LinearDominanceComparator;
import org.moeaframework.core.Initialization;
import org.moeaframework.core.NondominatedPopulation;
import org.moeaframework.core.Problem;
import org.moeaframework.core.Solution;
import org.moeaframework.core.operator.InjectedInitialization;
import org.moeaframework.core.operator.real.DifferentialEvolutionSelection;
import org.moeaframework.core.operator.real.DifferentialEvolutionVariation;
import org.moeaframework.core.variable.EncodingUtils;
import org.moeaframework.problem.misc.AnnErrorMinimizationProblem;

import java.util.ArrayList;
import java.util.List;

/**
 * MOEA Framework based optimizer.
 */
public class MoeaOptimizer implements Optimizer {
    /**
     *
     */
    private final long optimizationTimeout;

    /**
     *
     */
    private final BasicNetwork network;

    /**
     *
     */
    private final Propagation train;

    /**
     *
     */
    private final int populationSize;
```

```
    /**
     *
     */
    private final double crossoverRate;

    /**
     *
     */
    private final double scalingFactor;

    /**
     *
     * @param optimizationTimeout
     * @param network
     * @param train
     * @param populationSize
     * @param crossoverRate
     * @param scalingFactor
     */
    public MoeaOptimizer(long optimizationTimeout, BasicNetwork network, Propagation train, int popul
        this.optimizationTimeout = optimizationTimeout;
        this.network = network;
        this.train = train;
        this.populationSize = populationSize;
        this.crossoverRate = crossoverRate;
        this.scalingFactor = scalingFactor;
    }

    /**
     * {@inheritDoc}
     */
    @Override
    public List<Double> optimize(List<Double> weights) {
        Problem problem = new AnnErrorMinimizationProblem(weights, network, train);

        List<Solution> solutions = new ArrayList<Solution>();
        for (int i = 0; i < populationSize; i++) {
            Solution solution = problem.newSolution();
            solutions.add(solution);
        }

        AggregateObjectiveComparator comparator = new LinearDominanceComparator();
        Initialization initialization = new InjectedInitialization(problem, populationSize, solutions
        DifferentialEvolutionSelection selection = new DifferentialEvolutionSelection();
        DifferentialEvolutionVariation variation = new DifferentialEvolutionVariation(crossoverRate,

        AbstractAlgorithm algorithm = new DifferentialEvolution(problem, comparator, initialization,

        long stop = System.currentTimeMillis() + optimizationTimeout;
        while (System.currentTimeMillis() < stop) {
            algorithm.step();
        }

        weights = new ArrayList<Double>();
        NondominatedPopulation population = algorithm.getResult();
        if (population.size() > 0) {
            for (Double value : EncodingUtils.getReal(population.get(0))) {
                weights.add(value);
            }
        }

        return weights;
    }
}
```

Optimizer.java

```
package eu.veldsoft.vitosha.trade.engine;

import java.util.List;

/**
 * Optimizer interface.
 */
public interface Optimizer {
    /**
     * Single cycle of optimization.
     *
     * @param weights Initial weights.
     * @return Weights after single cycle of optimization.
     */
    public List<Double> optimize(List<Double> weights);
}
```

Predictor.java

```
package eu.veldsoft.vitosha.trade.engine;

import org.encog.engine.network.activation.ActivationFunction;
import org.encog.engine.network.activation.ActivationTANH;
import org.encog.ml.data.MLData;
import org.encog.ml.data.MLDataSet;
import org.encog.ml.data.basic.BasicMLData;
import org.encog.ml.data.basic.BasicMLDataSet;
import org.encog.ml.data.market.MarketDataDescription;
import org.encog.ml.data.market.MarketDataType;
import org.encog.ml.data.market.MarketMLDataSet;
import org.encog.ml.data.market.TickerSymbol;
import org.encog.ml.data.market.loader.LoadedMarketData;
import org.encog.ml.data.market.loader.MarketLoader;
import org.encog.neural.networks.BasicNetwork;
import org.encog.neural.networks.layers.BasicLayer;
import org.encog.neural.networks.training.propagation.Propagation;
import org.encog.neural.networks.training.propagation.resilient.ResilientPropagation;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collection;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Random;
import java.util.Set;

import eu.veldsoft.vitosha.trade.communication.NeuronType;
import eu.veldsoft.vitosha.trade.dummy.InputData;

/**
 * Forecasting engine main class.
 */
public class Predictor {

    /**
     * Pseudo-random number generator.
     */
    private static final Random PRNG = new Random();

    // TODO Put all colors in the settings dialog.

    /**
     * Colors used in the charts.
     */
    private static final int CHART_COLORS[] = {
```

```
        (95 << 24 | 0 << 16 | 255 << 8 | 0),
        (95 << 24 | 255 << 16 | 0 << 8 | 0),
};

/**
 * Colors used to visualize neural networks.
 */
private static final int ANN_COLORS[] = {
        (95 << 24 | 0 << 16 | 255 << 8 | 0),
        (95 << 24 | 255 << 16 | 255 << 8 | 255),
        (95 << 24 | 0 << 16 | 0 << 8 | 255),
        (95 << 24 | 255 << 16 | 255 << 8 | 255),
        (95 << 24 | 255 << 16 | 0 << 8 | 0),
};

/**
 * Neural network object.
 */
private static BasicNetwork network = new BasicNetwork();

/**
 * Training examples data set.
 */
private static MLDataSet examples = null;

/**
 * Data of the forecast.
 */
private static MLData forecast = null;

/**
 * Calculated output data.
 */
private static MLData output = null;

/**
 * Training rule object.
 */
private static Propagation train = null;

/**
 * Lowest and highest values of particular activation function. It is used for time series scalin
 *
 * @param activation Activation function object.
 * @return Array with two values - lowest in the first index and highest in the second index.
 */
private static double[] findLowAndHigh(ActivationFunction activation) {
    /*
     * Use range of double values.
     */
    double check[] = {
            Double.MIN_VALUE, -0.000001, -0.00001, -0.0001,
            -0.001, -0.01, -0.1, -1, -10, -100, -1000,
            -10000, -100000, -1000000, 0, 0.000001, 0.00001,
            0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000,
            100000, 1000000, Double.MAX_VALUE};

    /*
     * Map the range of double values to activation function output.
     */
    activation.activationFunction(check, 0, check.length);

    /*
     * Soft the result of the activation fuction output.
     */
    Arrays.sort(check);
```

```
        /*
         * Return minimum and maximum values of the activation function output.
         */
        return new double[]{check[0], check[check.length - 1]};
    }

    /**
     * Initialize common class members.
     */
    public void initialize() {
        Map<NeuronType, Integer> counters = new HashMap<NeuronType, Integer>();
        counters.put(NeuronType.REGULAR, 0);
        counters.put(NeuronType.BIAS, 0);
        counters.put(NeuronType.INPUT, 0);
        counters.put(NeuronType.OUTPUT, 0);

        for (int type : InputData.NEURONS) {
            counters.put(NeuronType.valueOf(type),
                    counters.get(NeuronType.valueOf(type)) + 1);
        }

        int inputSize = counters.get(NeuronType.INPUT);
        int hiddenSize = counters.get(NeuronType.REGULAR);
        int outputSize = counters.get(NeuronType.OUTPUT);

        /*
         * Network construction.
         */
        network.addLayer(new BasicLayer(null,
                true, inputSize));
        network.addLayer(new BasicLayer(new ActivationTANH(),
                true, hiddenSize));
        network.addLayer(new BasicLayer(new ActivationTANH(),
                false, outputSize));
        network.getStructure().finalizeStructure();
        network.reset();

        // TODO Load weights to the network.

        double values[] = InputData.RATES[PRNG.nextInt(InputData.RATES.length)];

        /*
         * Data construction.
         */
        MarketMLDataSet data = new MarketMLDataSet(new MarketLoader() {
            @Override
            public Collection<LoadedMarketData> load(TickerSymbol symbol,
                                                     Set<MarketDataType> types, Date start,
                                                     Date end) {
                Collection<LoadedMarketData> result = new
                        ArrayList<LoadedMarketData>();

                for (int i = 0; i < InputData.TIME.length; i++) {
                    /*
                     * Data outside of the desired time frame are not loaded.
                     */
                    if (InputData.TIME[i] < start.getTime() || end.getTime() < InputData.TIME[i]) {
                        continue;
                    }

                    LoadedMarketData value = new LoadedMarketData(new
                            Date(InputData.TIME[i]), symbol);

                    value.setData(MarketDataType.CLOSE, InputData.CLOSE[i]);
                    value.setData(MarketDataType.HIGH, InputData.HIGH[i]);
                    value.setData(MarketDataType.LOW, InputData.LOW[i]);
                    value.setData(MarketDataType.OPEN, InputData.OPEN[i]);
```

```
                value.setData(MarketDataType.VOLUME, InputData.VOLUME[i]);
                value.setData(MarketDataType.ADJUSTED_CLOSE, InputData.CLOSE[i]);

                result.add(value);
            }

            return result;
        }
    }, inputSize, outputSize);

    MarketDataDescription description = new MarketDataDescription(new
            TickerSymbol(InputData.SYMBOL),
            (new MarketDataType[]{MarketDataType.CLOSE, MarketDataType.HIGH,
                    MarketDataType.LOW,
                    MarketDataType.OPEN})[(int) (Math.random() * 4)],
            true, true);
    data.addDescription(description);
    data.load(new Date(InputData.TIME[0]), new
            Date(InputData.TIME[InputData.TIME.length - 1]));
    data.generate();

    /*
     * Normalize data.
     */
    double min = values[0];
    double max = values[0];
    for (double value : values) {
        if (value < min) {
            min = value;
        }
        if (value > max) {
            max = value;
        }
    }

    /*
     * At the first index is the low value. At the second index is the high
     * value.
     *
     * There is a problem with this approach, because some activation
     * functions are zero if the argument is infinity.
     *
     * The fist layer has no activation function.
     */
    double range[] = findLowAndHigh(network.getActivation(2));

    /*
     * Prepare training set.
     */
    double input[][] = new double[values.length -
            (inputSize + outputSize)][inputSize];
    double target[][] = new double[values.length -
            (inputSize + outputSize)][outputSize];
    for (int i = 0; i < values.length - (inputSize + outputSize); i++) {
        for (int j = 0; j < inputSize; j++) {
            input[i][j] = range[0] + (range[1] - range[0]) *
                    (values[i + j] - min) / (max - min);
        }
        for (int j = 0; j < outputSize; j++) {
            target[i][j] = range[0] + (range[1] - range[0]) *
                    (values[i + inputSize + j] - min) / (max - min);
        }
    }
    examples = new BasicMLDataSet(input, target);

    /*
     * Prepare forecast set.
```

```
         */
        input = new double[1][inputSize];
        for (int j = 0; j < inputSize; j++) {
            input[0][j] = range[0] + (range[1] - range[0]) *
                    (values[values.length - inputSize + j] - min) / (max - min);
        }
        forecast = new BasicMLData(input[0]);
    }

    /**
     * Single neural network training cycle.
     */
    public void train() {
        if (network == null) {
            return;
        }
        if (examples == null) {
            return;
        }

        train = new ResilientPropagation(network, examples);

        /*
         * Switch between backpropagation and genetic algorithm.
         */
        if (PRNG.nextBoolean() == true) {
            train.iteration();
            train.finishTraining();
        } else {
            int populationSize = 4 + PRNG.nextInt(33);
            double elitismRate = PRNG.nextInt(100) / 1000D;
            double crossoverRate = PRNG.nextInt(900) / 1000D;
            double mutationRate = PRNG.nextInt(10) / 1000D;
            int tournamentArity = PRNG.nextBoolean() ? 1 : 2;
            double scalingFactor = PRNG.nextInt(500) / 1000D;
            long optimizationTimeout = 1000;

            /*
             * Obtain ANN weights.
             */
            List<Double> weights = new ArrayList<Double>();
            for (int layer = 0; layer < network.getLayerCount() - 1; layer++) {
                int bias = network.isLayerBiased(layer) ? 1 : 0;
                for (int from = 0; from < network.getLayerNeuronCount(layer) + bias; from++) {
                    for (int to = 0; to < network.getLayerNeuronCount(layer + 1); to++) {
                        weights.add(network.getWeight(layer, from, to));
                    }
                }
            }

            /* Do evolutionary optimization. */
            Optimizer optimizer = new MoeaOptimizer(optimizationTimeout, network, train, populationSi
            weights = optimizer.optimize(weights);

            /*
             * Replace ANN weights.
             */
            for (int layer = 0, index = 0; layer < network.getLayerCount() - 1; layer++) {
                int bias = network.isLayerBiased(layer) ? 1 : 0;
                for (int from = 0; from < network.getLayerNeuronCount(layer) + bias; from++) {
                    for (int to = 0; to < network.getLayerNeuronCount(layer + 1); to++, index++) {
                        network.setWeight(layer, from, to, weights.get(index));
                    }
                }
            }
        }
    }
```

```
/**
 * Neural network prediction getter.
 */
public void predict() {
    if (network == null) {
        return;
    }
    if (forecast == null) {
        return;
    }

    output = network.compute(forecast);
}

/**
 * Draw forecast.
 *
 * @param pixels Array with ARGB pixels.
 * @param width  Drawing area width.
 * @param height Drawing area height.
 */
public void drawForecast(int[] pixels, int width, int height) {
    if (network == null) {
        return;
    }
    if (forecast == null) {
        return;
    }
    if (output == null) {
        return;
    }

    /*
     * Output layer activation function is used because input layer
     * has no activation function.
     */
    double range[] = findLowAndHigh(network.getActivation(2));

    /*
     * Total number of values to be visualized.
     */
    int numberOfValues = network.getLayerNeuronCount(0) +
            network.getLayerNeuronCount(2);

    int x = 0;
    int y = 0;

    /*
     * Visualize past data.
     */
    for (int i = 0; forecast.getData() != null &&
            i < forecast.getData().length; i++) {
        int offset = (int) (height * (forecast.getData()[i] - range[0]) /
                (range[1] - range[0]));
        for (int dx = 0; dx < width / numberOfValues; dx++) {
            for (y = height - offset; y < height; y++) {
                pixels[x + y * width] = CHART_COLORS[0];
            }
            x++;
        }
    }

    /*
     * Visualize future data.
     */
    for (int i = 0; output.getData() != null &&
```

```
            i < output.getData().length; i++) {
        int offset = (int) (height * (output.getData()[i] - range[0]) /
                (range[1] - range[0]));
        for (int dx = 0; dx < width / numberOfValues; dx++) {
            for (y = height - offset; y < height; y++) {
                pixels[x + y * width] = CHART_COLORS[1];
            }
            x++;
        }
    }
}

/**
 * Draw ANN topology.
 *
 * @param pixels Array with ARGB pixels.
 * @param width  Drawing area width.
 * @param height Drawing area height.
 */
public void drawAnn(int[] pixels, int width, int height) {
    if (network == null) {
        return;
    }
    if (forecast == null) {
        return;
    }
    if (output == null) {
        return;
    }

    /*
     * Artificial neural network.
     */
    double topology[][] = {
            forecast.getData(),
            new double[network.getLayerNeuronCount(0) *
                    network.getLayerNeuronCount(1)],
            new double[network.getLayerNeuronCount(1)],
            new double[network.getLayerNeuronCount(1) *
                    network.getLayerNeuronCount(2)],
            output.getData()
    };

    /*
     * At the first index is the low value. At the second index is
     * the high value.
     *
     * There is a problem with this approach, because some activation
     * functions are zero if the argument is infinity.
     *
     * The fist layer has no activation function.
     */
    double range[] = findLowAndHigh(network.getActivation(2));

    /*
     * Scale input layer data.
     */
    for (int i = 0; i < topology[0].length; i++) {
        topology[0][i] = (topology[0][i] - range[0]) /
                (range[1] - range[0]);
    }

    /*
     * Scale output layer data.
     */
    for (int i = 0; i < topology[4].length; i++) {
        topology[4][i] = (topology[4][i] - range[0]) /
```

```
            (range[1] - range[0]);
}

for (int i = 0, from = 0, to = 0, bias = network.isLayerBiased(0) ? 1 : 0; i < topology[1].le
    if (to >= network.getLayerNeuronCount(1)) {
        to = 0;
        from++;
    }
    if ((from + bias) >= network.getLayerNeuronCount(0)) {
        from = 0;
    }
    topology[1][i] = network.getWeight(0, from, to);
    to++;
}

for (int i = 0, from = 0, to = 0, bias = network.isLayerBiased(1) ? 1 : 0; i < topology[3].le
    if (to >= network.getLayerNeuronCount(2)) {
        to = 0;
        from++;
    }
    if ((from + bias) >= network.getLayerNeuronCount(1)) {
        from = 0;
    }
    topology[3][i] = network.getWeight(1, from, to);
    to++;
}

/*
 * Hidden layer values. Activation function of the second layer
 * is used for scaling.
 */
range = findLowAndHigh(network.getActivation(1));
for (int i = 0; i < topology[2].length; i++) {
    topology[2][i] = (network.getLayerOutput(1, i) - range[0]) /
            (range[1] - range[0]);
}

/*
 * Weights normalization.
 */
double min = Double.MAX_VALUE;
double max = Double.MIN_VALUE;
for (double value : topology[1]) {
    if (value < min) {
        min = value;
    }
    if (value > max) {
        max = value;
    }
}
for (double value : topology[3]) {
    if (value < min) {
        min = value;
    }
    if (value > max) {
        max = value;
    }
}
for (int i = 0; i < topology[1].length; i++) {
    topology[1][i] = (topology[1][i] - min) / (max - min);
}
for (int i = 0; i < topology[3].length; i++) {
    topology[3][i] = (topology[3][i] - min) / (max - min);
}

for (int x = 0, k = 0; k < ANN_COLORS.length;
     x += width / ANN_COLORS.length, k++) {
```

```
            for (int dx = 0; dx < width / ANN_COLORS.length; dx++) {
                for (int y = 0, l = 0; y < height &&
                        l < topology[k].length; y += height / topology[k].length, l++) {
                    for (int dy = 0; dy < height / topology[k].length; dy++) {
                        int alpha = (int) (((ANN_COLORS[k] >> 24) & 0xFF) * topology[k][l]);
                        int red = (int) (((ANN_COLORS[k] >> 16) & 0xFF) * topology[k][l]);
                        int green = (int) (((ANN_COLORS[k] >> 8) & 0xFF) * topology[k][l]);
                        int blue = (int) ((ANN_COLORS[k] & 0xFF) * topology[k][l]);

                        int color = alpha << 24 | red << 16 | green << 8 | blue;

                        pixels[(x + dx) + (y + dy) * width] = color;
                    }
                }
            }
        }
    }
}
```

UniformBinaryMutation.java

```
package org.apache.commons.math3.genetics;

import org.apache.commons.math3.exception.MathIllegalArgumentException;
import org.apache.commons.math3.exception.util.LocalizedFormats;
import org.apache.commons.math3.random.RandomGenerator;

import java.util.ArrayList;
import java.util.List;

/**
 * Uniform random bits mutation applied over vector of double values.
 */
public class UniformBinaryMutation implements MutationPolicy {

    /**
     * Pseudo random number generator.
     */
    private static final RandomGenerator PRNG =
            GeneticAlgorithm.getRandomGenerator();

    /**
     * {@inheritDoc}
     */
    @Override
    public Chromosome mutate(Chromosome original)
            throws MathIllegalArgumentException {
        if (original instanceof WeightsChromosome == false) {
            throw new MathIllegalArgumentException(
                    LocalizedFormats.INVALID_BINARY_CHROMOSOME);
        }

        /*
         * Deep copy of list values.
         */
        List<Double> representation = new ArrayList<Double>(
                ((WeightsChromosome) original).getRepresentation());

        /*
         * Mutate a single bit in each chromosome value.
         */
        for (int i = 0; i < representation.size(); i++) {
            double value = representation.get(i);
            if (PRNG.nextBoolean() == true) {
                value -= Double.MIN_VALUE;
            } else {
                value += Double.MIN_VALUE;
```

```
            }
            representation.set(i, value);
        }

        /*
         * Construct new chromosome after mutation.
         */
        return ((WeightsChromosome) original).
                newFixedLengthChromosome(representation);
    }
}
```

WeightsChromosome.java

```
package org.apache.commons.math3.genetics;

import org.encog.neural.networks.BasicNetwork;
import org.encog.neural.networks.training.Train;

import java.util.List;

/**
 * Chromosome with double values of the weights from the artificial neural
 * network.
 */
public class WeightsChromosome extends AbstractListChromosome<Double> {

    /**
     * Lazy initialization and buffering for the fitness value.
     */
    double fitness = -(Double.MAX_VALUE - 1);
    /**
     * Reference to external neural network object.
     */
    private BasicNetwork network = null;
    /**
     * Reference to external training strategy object.
     */
    private Train train = null;

    /**
     * Constructor used to initialize the chromosome with array of values.
     *
     * @param representation Values as array.
     * @param network        Neural network reference.
     * @param train          Neural network training strategy.
     * @throws InvalidRepresentationException Rise an exception if the values
     *                                        are not valid.
     */
    public WeightsChromosome(Double[] representation,
                             BasicNetwork network, Train train)
            throws InvalidRepresentationException {
        super(representation);
        this.network = network;
        this.train = train;

        if (network == null) {
            throw new RuntimeException("Neural network should be provided for the fitness evaluation.
        }

        if (train == null) {
            throw new RuntimeException("Training object should be provided for the fitness evaluation
        }

        update();
    }
```

```
/**
 * Constructor used to initialize the chromosome with list of values.
 *
 * @param representation Values as list.
 * @param network        Neural network reference.
 * @param train          Neural network training strategy.
 * @throws InvalidRepresentationException Rise an exception if the values
 *                                        are not valid.
 */
public WeightsChromosome(List<Double> representation,
                         BasicNetwork network, Train train)
        throws InvalidRepresentationException {
    super(representation);
    this.network = network;
    this.train = train;

    if (network == null) {
        throw new RuntimeException("Neural network should be provided for the fitness evaluation.
    }

    if (train == null) {
        throw new RuntimeException("Training object should be provided for the fitness evaluation
    }

    update();
}

/**
 * Constructor used to initialize the chromosome with list of values.
 *
 * @param representation Values as list.
 * @param copy           Deep copy flag.
 * @param network        Neural network reference.
 * @param train          Neural network training strategy.
 */
public WeightsChromosome(List<Double> representation, boolean copy,
                         BasicNetwork network, Train train) {
    super(representation, copy);
    this.network = network;
    this.train = train;

    if (network == null) {
        throw new RuntimeException("Neural network should be provided for the fitness evaluation.
    }

    if (train == null) {
        throw new RuntimeException("Training object should be provided for the fitness evaluation
    }

    update();
}

/**
 * Fitness value update after chromosome representation change.
 */
private void update() {
    if (network == null) {
        throw new RuntimeException("Neural network should be provided for the fitness evaluation.
    }

    if (train == null) {
        throw new RuntimeException("Training object should be provided for the fitness evaluation
    }

    /*
     * Load weights from the internal representation into the network
     * structure.
```

```
         */
        List<Double> weights = getRepresentation();
        for (int layer = 0, index = 0; layer < network.getLayerCount() - 1; layer++) {
            int bias = network.isLayerBiased(layer) ? 1 : 0;
            for (int from = 0; from < network.getLayerNeuronCount(layer) + bias; from++) {
                for (int to = 0; to < network.getLayerNeuronCount(layer + 1); to++, index++) {
                    network.setWeight(layer, from, to, weights.get(index));
                }
            }
        }

        /*
         * Iterate over the training set in order to calculate network error.
         */
        train.iteration();

        /*
         * Total ANN error is used as fitness value. The bigger the fitness, the better the chromoson
         */
        fitness = -train.getError();
    }

    /**
     * {@inheritDoc}
     */
    @Override
    public double fitness() {
        return fitness;
    }

    /**
     * {@inheritDoc}
     */
    @Override
    protected void checkValidity(List<Double> values)
            throws InvalidRepresentationException {
        if (network == null) {
            //TODO throw new RuntimeException("Neural network should be provided for the fitness eval
            return;
        }

        /*
         * Length of the values should match the number of weights in the neural
         * network structure.
         */
        int counter = 0;
        for (int layer = 0; layer < network.getLayerCount() - 1; layer++) {
            int bias = network.isLayerBiased(layer) ? 1 : 0;
            for (int from = 0; from < network.getLayerNeuronCount(layer) + bias; from++) {
                for (int to = 0; to < network.getLayerNeuronCount(layer + 1); to++) {
                    counter++;
                }
            }
        }

        if (values == null || counter != values.size()) {
            // TODO Report the size problem.
        }
    }

    /**
     * {@inheritDoc}
     */
    @Override
    public AbstractListChromosome<Double> newFixedLengthChromosome(
            List<Double> values) {
        return new WeightsChromosome(values, true, network, train);
```

```
    }

    /**
     * Chromosome representation getter.
     *
     * @return List with chromosome values.
     */
    public List<Double> getRepresentation() {
        return getRepresentation();
    }
}
```

AnnErrorMinimizationProblem.java

```
package org.moeaframework.problem.misc;

import org.encog.neural.networks.BasicNetwork;
import org.encog.neural.networks.training.propagation.Propagation;
import org.moeaframework.core.Solution;
import org.moeaframework.core.variable.EncodingUtils;
import org.moeaframework.core.variable.RealVariable;
import org.moeaframework.problem.AbstractProblem;

import java.util.List;

/**
 * ANN error minimization problem.
 */
public class AnnErrorMinimizationProblem extends AbstractProblem {

    /**
     *
     */
    private BasicNetwork network;

    /**
     *
     */
    private Propagation train;

    /**
     *
     */
    private List<Double> initial;

    /**
     * @param solution
     * @param network
     * @param train
     */
    public AnnErrorMinimizationProblem(List<Double> solution, BasicNetwork network, Propagation train
        this(solution.size(), 1);
        initial = solution;
        this.network = network;
        this.train = train;
    }

    /**
     * {@inheritDoc}
     */
    private AnnErrorMinimizationProblem(int numberOfVariables, int numberOfObjectives) {
        super(numberOfVariables, numberOfObjectives);
    }

    /**
     * {@inheritDoc}
     */
```

```
    private AnnErrorMinimizationProblem(int numberOfVariables, int numberOfObjectives, int numberOfCo
        super(numberOfVariables, numberOfObjectives, numberOfConstraints);
    }

    /**
     * {@inheritDoc}
     */
    @Override
    public void evaluate(Solution solution) {
        if (network == null) {
            throw new RuntimeException("Neural network should be provided for the fitness evaluation.
        }

        if (train == null) {
            throw new RuntimeException("Training object should be provided for the fitness evaluation
        }

        /*
         * Load weights from the internal representation into the network
         * structure.
         */
        double[] weights = EncodingUtils.getReal(solution);
        for (int layer = 0, index = 0; layer < network.getLayerCount() - 1; layer++) {
            int bias = network.isLayerBiased(layer) ? 1 : 0;
            for (int from = 0; from < network.getLayerNeuronCount(layer) + bias; from++) {
                for (int to = 0; to < network.getLayerNeuronCount(layer + 1); to++, index++) {
                    network.setWeight(layer, from, to, weights[index]);
                }
            }
        }

        /*
         * Iterate over the training set in order to calculate network error.
         */
        train.iteration();

        /*
         * Total ANN error is used as fitness value. The bigger the fitness, the better the chromoson
         */
        solution.setObjective(0, -train.getError());
    }

    /**
     * {@inheritDoc}
     */
    @Override
    public Solution newSolution() {
        Solution solution = new Solution(initial.size(), 1, 0);
        for (int i = 0; i < initial.size(); i++) {
            solution.setVariable(i, new RealVariable(initial.get(i), -Double.MAX_VALUE + 1, Double.MA
        }
        return solution;
    }
}
```