Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский физико-технический институт (государственный университет)»
Физтех-школа прикладной математики и информатики
кафедра интеллектуальных систем

Направление подготовки: 03.03.01 Прикладные математика и физика (бакалавриат)
Направленность (профиль) подготовки: Компьютерные технологии и интеллектуальный анализ данных

Пространство параметров для различных априорных распределений

(Бакалаврский диплом)

Студент:
Аминов Тимур Венерович

(подпись студента)

Научный руководитель:
Зайцев Алексей Алексеевич,
к. ф.-м. н.

(подпись научного руководителя)

Москва 2020

Оглавление

Аннотация

Во время обучения нейронных сетей, оценка параметров сети — результат решения задачи минимизации функции ошибки. Зависимость функции ошибки от параметров модели на основе нейронных сетей сложная: она не является выпуклой, у нее много локальных оптимумов.

В силу этого напрямую изучать свойства данной поверхности не представляется возможным: математический аппарат для таких сложных случаев отсутствует. Существуют методы исследования сложности структуры пространства параметров нейронной сети, однако до этого никто не исследовал влияние априорного распределения на это пространство и зависимость между архитектурой сети и связанностью оптимальных значений параметров.

В этой работе мы рассмотрели влияние априорного распределения на структуру пространства параметров, а так же влияние архитектуры модели на существование изолированных экстремумов для функции потерь. Мы показали, наложение априорного распределения сглаживает пространство параметров, а так же установили что при росте числа нейронов экстремумы функции ошибки перестают быть сильно изолированными.

Глава 1

Введение

1.1 Введение

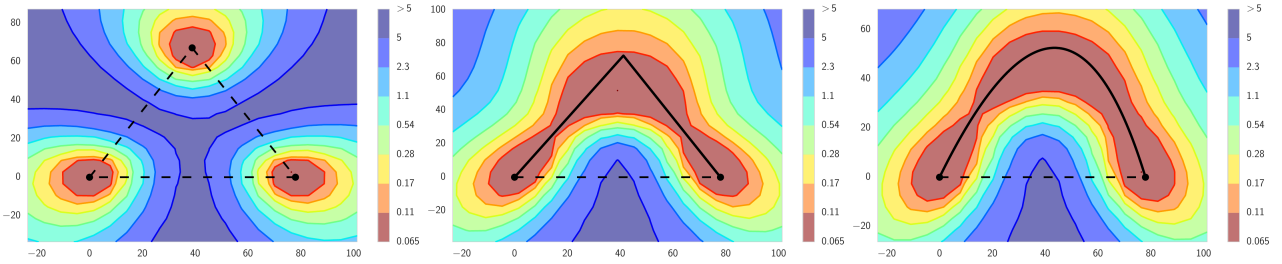
Нейронные сети являются удобным инструментом для построения моделей в широком ряде задач. Существуют различные виды нейронных сетей: Нейронные сети прямого распространения (feed forward neural networks, FF или FFNN), полносвязные нейронные сети (fully connected FC), сверточные нейронные сети (neural networks CNN), рекурентные нейронные сети (recurrent neural networks RNN). Эти модели используются для решения разных задач машинного обучения, однако во всех моделях не ясно, как устроено внутренее пространство параметров: как функция ошибки зависит от значений параметров нейронной сети.

Изучить свойства этого пространства с помощью явных математических методов не представляется возможным. Это вызвано тем, что это пространство имеет сложную структуру: функция ошибки не является выпуклой или одноэкстремальной функцией параметров модели.

Поэтому мы используем более «физические» методы исследования. Чтобы получить информацию о структуре пространства параметров модели можно строить кривые в этом пространстве с низкими значениями функции потерь вдоль нее, соединящие разные локальные минимумы функции ошибки. Такие кривые строятся, например, в работах [17], [1]. Основной идеей в этих работах является использование методов, основанных на статистическом подходе для построения данных кривых, а так же построение новых методов для решения этой задачи.

В диссертационном исследовании с помощью методов построения кривых низких значений функции ошибки мы исследуем пространство параметров нейронных сетей. А именно, мы решаем следующие две задачи:

- построение зависимости потери качества классификации вдоль кривой от числа нейронов в скрытом слое.
- построение зависимости потери качества классификации вдоль от априорного распределения для пространства параметров.



(a) Изолированные минимумы. (b) Соединение отрезками. (c) Соединение кривой сложной формы.

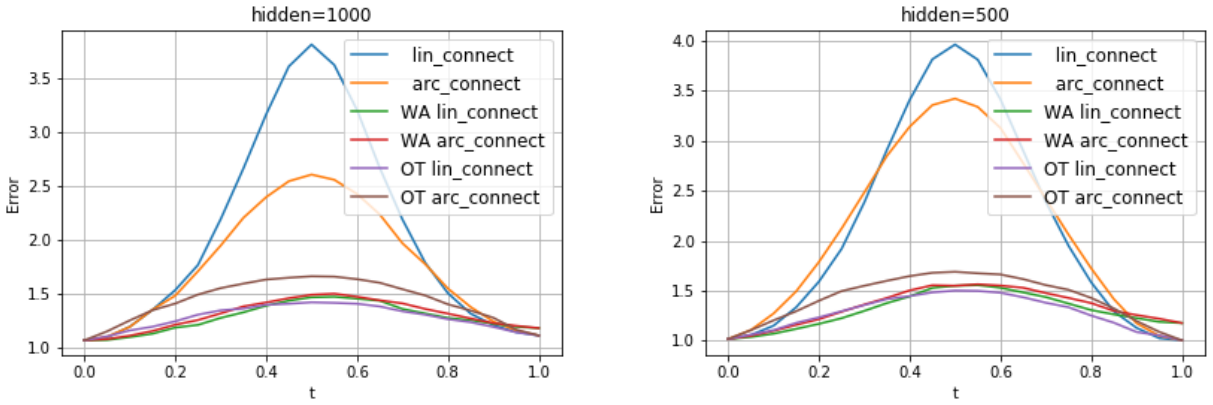
Рис. 1.1: Визуализация двумерной проекции пространства параметров и функции ошибки для него. По осям — значения проекция параметров модели, цвет соответствует значению функции ошибки. Рисунок из статьи [1]

На рисунке 1.1 из статьи [1] мы наблюдаем линии уровня функции потерь. На рисунке приведено два случая: когда локальные минимумы изолированы и когда они соединены кривой, вдоль которой значение функции ошибки почти не меняется. так же мы видим различные методы соединения двух экстремумов в этом пространтве. Уже здесь можно заметить что различные методы могут показывать различные результаты. Для некоторых способов соединения локальных оптимумов кривая проходит через области с высокими значениями функции ошибки потерь.

В нашей работе мы рассматриваем, как от спецификации модели нейронной сети зависит того, попадаем ли мы в ситуации с изолированными или с соединными кривой с низкими значениями функции ошибки локальными оптимумами. Например, рисунок 1.2 демонстрирует, как меняется функция ошибки вдоль разных траекторий, соединяющих два ее локальных минимума с и без априорным распределениям для параметров.

1.2 Обзор литературы

Глубокое обучение это раздел машинного обучения. Как и машинное обучение глубое обучение позволяет создавать модели, которые по входным данным предсказывать результат. Для того чтобы обучить модель, используются различные методы оптимизации. Каждый из этих методов оптимизации никаким образом не учитывает структуру



(a) Без априорного распределения

(b) Нормальное априорное распределение

Рис. 1.2: Сравнение различных методов построение кривых, соединяющих локальные оптимумы для различных априорных распределений и различного числа нейронов

модели: наиболее часто используемые ADAM и SGD [3] — методы первого порядка общего назначения.

Результатом работы таких алгоритмов для глубоких нейроных сетей является достаточно робастный локальный оптимум функции ошибки. Разные запуски оценки параметров обычно дают разные локальные оптимумы. Это происходит как потому, что перестановка нейронов дает такое же качество нейронной сети, так и потому, что существует много локальных минимумов функции ошибки [9], для большинства из которых функция ошибки на тестовой выборке тоже маленькая [10].

Некоторые работы исследуют, как устроено пространство параметров нейронной сети и как оно меняется с изменением сложности модели. Например, оказывается, что наблюдается такой эффект как продолжение снижения ошибки на тесте при увеличении сложности модели [2, 16]. Другим способом исследования того, как устроено пространство параметров модели является анализ топологической структуры полученного пространства [15].

В нашей работе мы смотрим на это пространство с другой точки. А именно, мы смотрим, как связаны между собой локальные оптимумы функции потерь. Считается, что для моделей с малым числом параметров локальные минимумы функции ошибки изолированы, а для переопределенных моделей локальные минимумы соединены кривыми, вдоль которых функция ошибки меняется мало [1].

Такие кривые нельзя построить аналитически в общем случае. Поэтому, для их построения используют подходы на основе явной минимизации оценки интеграла функции

ошибки вдоль кривой [1] и другие подходы, которые учитывают структуру пространства параметров [17, 9]. Разработанные метооды для рассматриваемых архитектур глубоких нейронных сетей позволяют получить кривые, соединящие локальные минимумы функции ошибки и обладающие маленькие значениями этой функции вдоль кривой.

Однако, пока не было исследовано, когда мы переходим в режим переопределенной модели с точки зрения изолированности локальных оптимумов и как на эту точку перехода влияет наличие регуляризующего априорного распределения для параметров модели. Проведение такого исследования позволило бы лучше понять, как устроено пространство параметров нейронных сетей и учесть это при разработке новых алгоритмов оптимизации параметров и создания ансмаблей моделей нейроных сетей разной сложности.

Глава 2

Теоретическая часть

2.1 Особенности нейронных сетей как моделей машинного обучения

2.1.1 Модель нейронной сети как модель машинного обучения

Введем основные понятия, связанные с нейронными сетями.

Определение 2.1.1. ***нейронная сеть*** - это модель, которая по входу $\mathbf{x}_t \in \mathbb{R}^n$, строит предсказание $\mathbf{y}_t \in \mathbb{R}^m$

Разные выходы нейронной сети дают решения разных задач машинного обучения. Для $\mathbf{y}$ из всего $\mathbb{R}^m$ мы получаем решение задачи регрессии с m-мерным выходом. Для $\mathbf{y}$ такого, что его компоненты $y_i \geq 0$ и $\sum_{i=1}^{m} y_i = 1$ мы получаем решение задачи классификации на m классов с компонентами вектора, соответствующими вероятностям соответствующих классов. В этой работе мы будем далее предполагать, что решается задача m-классовой классификации.

2.1.2 Модели многослойного и однослойного перцептрона

Для своих экспериментов мы выбрали модель многослойного перцептрона в виду ее простоты и скорости ее обучения. Определим эту модель.

Для начала введем определение однослойного перцептрона:

Определение 2.1.2. ***Однослойный перцептрон*** — модель машинного обучения, принадлежащая к классу нейронноых сетей. В этой модели сигналы от входного слоя сразу подаются на выходной слой, который преобразует сигнал и выдает результат работы модели:

$$y = \phi\left(\sum_{i=1}^{n} w_i x_i\right) = \phi(\mathbf{wx}).$$

- $\mathbf{x}$ — входной сигнал (как правило вектор линейного пространства $\mathbf{x} \in \mathbb{R}^n$)
- x_i — компоненты входного сигнала (числа)
- $\mathbf{w}$ — вектор весов ($\mathbf{w} \in \mathbb{R}^n$)
- w_i — компоненты вектора весов
- ϕ — нелинейная функция активации. В нашем случае мы используем сигмоидную функцию активации, которая покомпоненто преобразует вход по следующей формуле:

$$\phi_i = \phi(x_i) = \frac{1}{1 + \exp(-x_i)}.$$

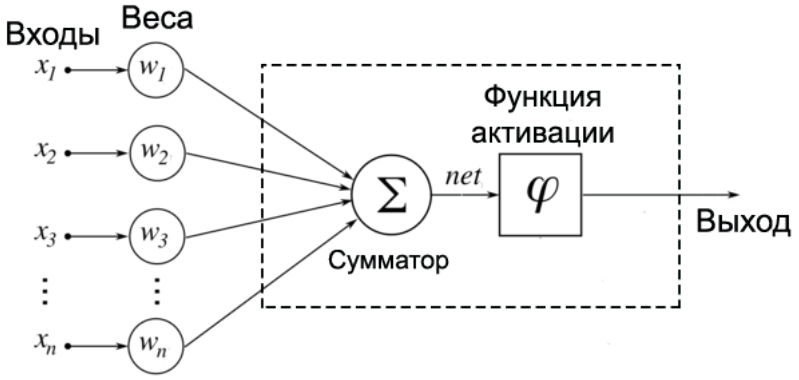


Рис. 2.1: Схема однослойного перцептрона

Определение 2.1.3. ***Многослойный перцептрон*** — сеть, в которой сигналы сначала преобразуются в скрытых слоях, а затем подаются на выходной слой

$$\mathbf{y} = (\mathbf{W}_2\phi(\mathbf{W}_1\mathbf{x}))$$

- $\mathbf{x}$ — вектор линейного пространства $x \in \mathbb{R}^n$
- $\mathbf{W}_1$ — матрица весов первого слоя $\mathbf{W} \in \mathbb{R}^{d \times n}$, где d - размерность скрытого слоя
- $\mathbf{W}_2$ — матрица весов финального слоя $\mathbf{W}_1 \in \mathbb{R}^{d_1 \times d_2}$, где d_2 — количство классов в задаче классификации
- ϕ — нелинейная функция активации. Для многослойного перцептона мы используем функцию активации ReLU:

$$ReLU(\mathbf{x}) = \max(0, \mathbf{x})$$

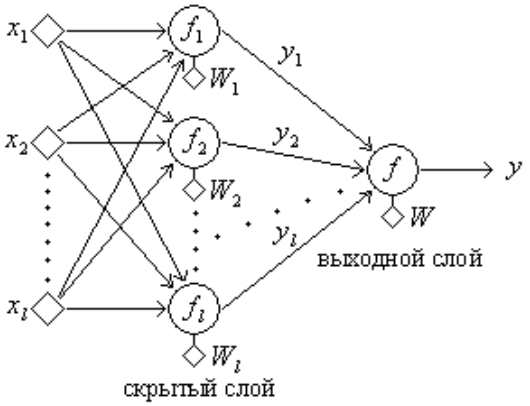


Рис. 2.2: Схема многослойного перцептрона

2.1.3 Оценка параметров нейронной сети

Под пространством параметров нейронной сети мы понимаем $\mathbb{R}^{|net|}$, где $|net|$ — это число параметов сети. В глубинном обучении мы предполагаем что в этом пространстве определена некоторая функция ошибки $\mathbb{L}$, которую мы хотим минизировать для построения модели. Это делается с помошью различных методов оптимизации, таких как ADAM [12] и SGD. В процессе применения этих методов используются градиент функции ошибки по параметрам модели.

2.2 Построение кривой с низким значением функции ошибки вдоль нее

Результатом работы процедуры оценки параметров нейронной сети, описанной в разделе 2.1.3 является вектор ее параметров, соответствующий локальному оптимуму функции ошибки $\mathbb{L}(\boldsymbol{\theta})$.

Разные запуски алгоритма оптимизации параметров нейронной сети дают оценки параметров, соответствующие разным локальным оптимумам функции ошибки. Пусть мы рассматриваем два вектора $\boldsymbol{\theta}_A$, $\boldsymbol{\theta}_B$, соответствующие разные локальным оптимумам функции ошибки. Вектора $\boldsymbol{\theta}_A, \boldsymbol{\theta}_B \in \mathbb{R}^p$, где p — размерность пространства параметров нейронной сети.

Мы хотим построить кривую, которая соединяет эти два вектора в пространстве параметров. Каждой такой кривой соответствует отображение $\psi_{\boldsymbol{\alpha}} : [0, 1] \to \mathbb{R}^p$. Мы хотим чтобы это отображение являлось непрерывным и кусочно-гладким, а так же чтобы $\psi_{\boldsymbol{\alpha}}(0) = \boldsymbol{\theta}_A$ и $\psi_{\boldsymbol{\alpha}}(1) = \boldsymbol{\theta}_B$. Вектор параметров кривой $\boldsymbol{\alpha}$ задает конкретное отображение.

Из всех таких кривых нас интересуют те, на которых значение функции ошибки

как можно меньше. Формально мы хотим, чтобы вектор параметров $\boldsymbol{\alpha}$ минимизировал интеграл $I(\boldsymbol{\alpha})$:

$$I(\boldsymbol{\alpha}) = \frac{\int \mathbb{L}(\psi_\alpha) d\psi_\alpha}{\int d\psi_\alpha} = \frac{\int_0^1 \mathbb{L}(\psi_\alpha(t)) \left\|\psi`_\alpha\right\| dt}{\int_0^1 \left\|\psi`_\alpha(t)\right\| dt} = \int_0^1 \mathbb{L}(\psi_\alpha(t)) q_\alpha(t) dt = \mathbb{E}_{t\sim q_\alpha(t)} \left[\mathbb{L}(\psi_\alpha(t))\right],$$

где $q_\alpha(t)$ — распределение для $t \in [0,1]$, $q_\alpha(t) = \left\|\psi`_\alpha\right\| \cdot \left(\int_0^1 \left\|\psi`_\alpha dt\right\|\right)^{-1}$ Заметим что, интеграл в основании дроби это просто нормировочная константа. Таким образом можно его опустить.

В итоге получаем выражение:

$$\mathbf{l}(\alpha) = \int_0^1 \mathbb{L}(\psi_\alpha(t)) dt = \mathbb{E}_{t\sim U[0,1]} \mathbb{L}(\psi_\alpha(t)$$

Для построения этой кривой существуют различные методы [1], [17]. Мы приведем те методы, которые будем использовать в данной работе. Методы отличаются выбором семейства кривых и процедурой оптимизации параметров $\boldsymbol{\alpha}$.

2.2.1 Отрезок

Очевидно что для того чтобы соединить два вектора $\widehat{\boldsymbol{\theta}}_A$ и $\widehat{\boldsymbol{\theta}}_B$ прямой линией, тогда мы получим очень простую форму для отображениия $\psi(t)$:

$$\psi(t) = (1-t)\widehat{\boldsymbol{\theta}}_A + t\boldsymbol{\theta}_B$$

Для достаточно сложных пространств параметров эта кривая не дает достаточно хороших результатов.

Например, она не сохраняет распределние праметров вдоль кривой. Покажем это. Пусть $\boldsymbol{\theta}_A, \boldsymbol{\theta}_B \sim p$, то есть оба этих вектора параметров порождены одним распределением p.

Тогда в общем случае $\psi(t) \nsim p$ для $t \in [0,1]$. Чтобы убедиться в этом рассмотрим матрицу ковариации $\Sigma_{\psi(t)}$:

$$\Sigma_{\psi(t)} = (1-t)^2 \Sigma_{\hat{\boldsymbol{\theta}}_A} + t^2 \Sigma_{\hat{\boldsymbol{\theta}}_B} = ((1-t)^2 + t^2)\Sigma_p \neq \Sigma_p.$$

Поэтому этот метод не может дать в общем случае достаточно хорошего результата.

2.2.2 Дуга

У нас есть два вектора $\boldsymbol{\theta}_A$ и $\boldsymbol{\theta}_B$.

Предположим, что эти вектора порождены многомерным нормальным распределением с одинаковой матрицей ковариации и математическим ожиданием $\mu = \mathbb{E}\boldsymbol{\theta}_A = \mathbb{E}\boldsymbol{\theta}_B$.

Рассмотрим для них кривую:

$$\psi(t) = \mu + \cos\left(\frac{\pi}{2}t\right)(\widehat{\boldsymbol{\theta}}_A - \mu) + \sin\left(\frac{\pi}{2}t\right)(\widehat{\boldsymbol{\theta}}_B - \mu),$$

Можно показать, что для такой кривой в сделанных предположениях распределение не меняется. А именно, выполнена следующая теорема [17].

Теорема 2.2.1. *Если $\boldsymbol{\theta}_A$ и $\boldsymbol{\theta}_B$ — это случайные величины из одного и того же многомерного нормального распределения p с нулевым математическим ожиданием. Тогда для любого $t \in [0, 1]$ выполнено, что*

$\psi(t) = \cos\left(\frac{\pi}{2}t\right)\boldsymbol{\theta}_A + \sin\left(\frac{\pi}{2}t\right)\boldsymbol{\theta}_B$ *будет иметь то же распределение. Кроме того, выполнено, что $\psi(0) = \boldsymbol{\theta}_A$ и $\psi(1) = \boldsymbol{\theta}_B$.*

Доказательство этой теоремы проводится с помощью метода характерестических функций.

Так как апостериорное распределение для большинства статистических моделей близко к нормальному в силу теоремы Бернштейна-фон-Мизеса [4]. Для нейронной сети у пространства параметров структура более сложная, однако, такая аппроксимация хорошо работает на практике [7].

Чтобы сделать целевое распределение ближе к нормальму, обобщим этот метод на более широкий класс распределений. Введем дополнительное отображение ν, которое преобразует исходное распредение к нормальному. Тогда кривая $\psi(t)$ будет выглядеть следующим образом:

$$\psi(t) = \nu^{-1}\left[\cos\left(\frac{\pi}{2}t\right)\nu(\boldsymbol{\theta}_A) + \sin\left(\frac{\pi}{2}t\right)\nu(\boldsymbol{\theta}_B)\right]$$

На практике отображение ν нам не известно заранее, но мы можем попытаться найти подходящее отображение с помощью обучающей выборки. Для того чтобы найти это отображение мы используем еще одну нейронную сеть, которая поддерживает обратное отображение. Сети такого вида часто используются для преобразования простых

распределений (например нормального) в сложные и мультимодальные, с сохранением возможности посчитать плотность конкретной точки. Обучение и быстрое использование этих сетей возможно благодаря использованию преобразований Якобиана, которых легко получить[5], [11].

2.3 Байесовский подход к оценке параметров модели

Мощным инструментом для наделения решения задачи машинного обучения заданным свойствами является введение регуляризации или штрафа на вектор параметров модели. Этот подход хорошо работает уже для линейных и обобщенный линейных моделей, таких как логистическая регрессия. Обычно в функцию потерь в этом случае добавляют слагаемое, которое равно коэффициенту регуляризации, умноженному на l_1 или l_2 норму вектора параметров модели. Результатом являются более устойчивые модели с теоретичеки более удачными свойствами.

У регуляризации есть естественная вероятностная интерпретация. Мы предполагаем, что вектор параметров модели — случайная величина. У этой случайной величины задано априорное распределение

Например, для модели линейной регрессии получаем компоненты нашей вероятностной модели, связанные с шумом в наблюдениях и априорным распределением. Модель наблюдения с нормальным шумом имеет вид:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\theta} + \epsilon,\ \epsilon \sim \mathbb{N}(\mathbf{0}, \sigma^2\mathbb{I}),$$

где $\mathbf{y} \in \mathbb{R}^n,\ \mathbf{X} \in \mathbb{R}^{n\times d},\ \theta \in \mathbb{R}^d$

В качестве априорного имеет смысл взять многомерное нормальное распределение, так как оно является сопряженным: Теперь дополнительно предположим что $\theta \sim p(\boldsymbol{\theta}|\alpha)$, тогда

$p(\mathbf{y}, \theta|\mathbf{X}, \alpha) = p(\mathbf{y}|\mathbf{X}, \theta)p(\theta|\alpha)$ В таком случае в явном виде получаем апостериорное распределение: $p(\theta|\mathbf{X}, \mathbf{y}, \alpha) = \frac{p(\mathbf{y}, \theta|\mathbf{X}, \alpha)}{p(\mathbf{y}|\mathbf{X}, \alpha)}$.

Аналогичным образом работает Байесовский подход и для нейронных сетей. В этом случае мы тоже вводим нормальное или Бернулевское априорное распределение для вектора параметров модели. Апостериорное распределение в этом случае не имеет явного вид, и мы вынуждены использовать методы приближенного Байесовского вывода [14].

2.3.1 Нормальное распределение

Часто в качестве априорного распредения выбирают нормальное, с нулевым матожиданием. Для этого есть две причины: во-первых, нулевое математическое ожидание смещает апостериорные распределения параметров к нулю, позволяя избежать чрезмерно больших значений, а во-вторых с помощью изменения диспресии априорного распределения мы можем регулировать силу влияния априорного распределения на итоговый результат.

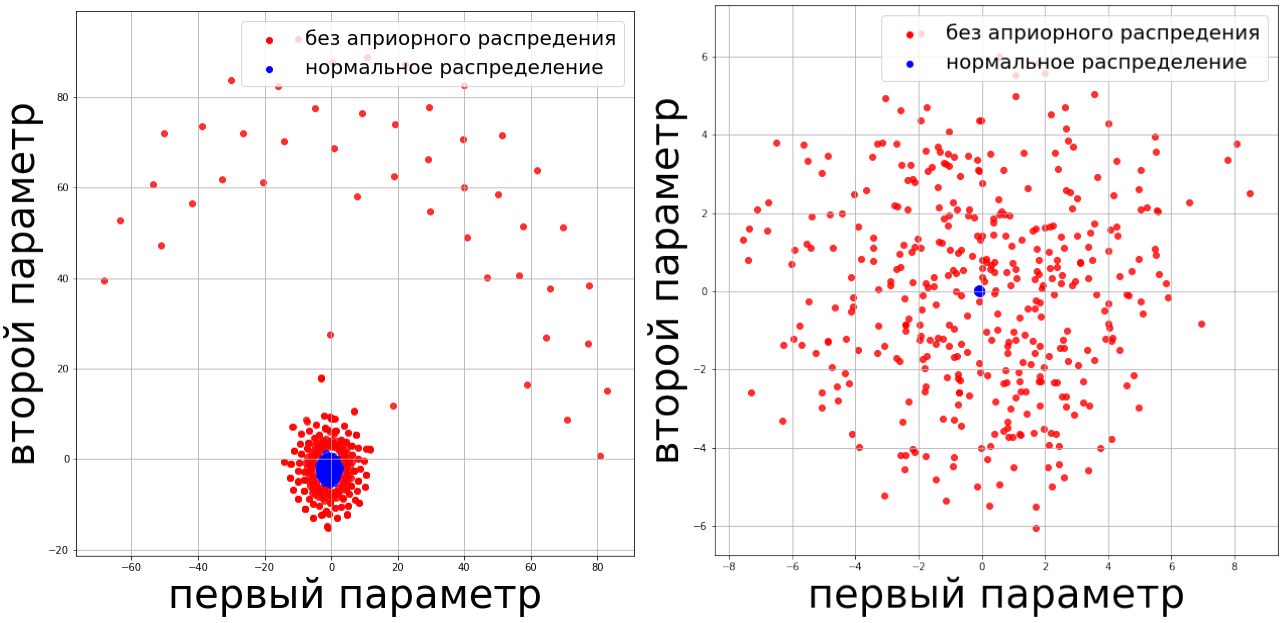
Мы рассмотрим, как расположены полученные значения параметров относительно друг друга для случая с и без априорного распределения. Для этого мы сначала получим набор векторов параметров, а затем построим их отображение в двумерное пространство.

Мы дважды обучали одну и ту же нейронную сеть с одним скрытым слоем. В первом случае мы не накладывали никаких дополнительных ограничений на параметры и получили набор векторов параметров, соответсвующих красным точкам. Для второго случая мы наложили на параметры априорное нормальное распределение и получили набор векторов параметров, соответсвующих синим точкам. После этого мы получили матрицы весов каждой модели и положили столбцы этих матриц векторами линейного пространства. После этого мы применили алгоритмы снижения размерности PCA и tSNE для того, чтобы отобразить их в двумерное пространство. По осям мы откладываем значения каждого вектора в этом двумерном пространстве.

Проиллюстрируем эти эффекты на рисунках 5.24. Мы видим, что действительно полученные значения параметров в случае использования априорного распределения сконцетрированы около одного значения, соответствующего нулевому векторму параметров.

2.3.2 Dropout регуляризация

Так же хорошо известен тот факт, что наложение распределения бернулли в качестве априорного, велечет включение такого механизма регуляризации как Dropout. [8]



(a) Сжатие с помощью tSNE.

(b) Сжатие с помощью PCA.

Рис. 2.3: Низкоразмерные представления.

2.4 Детали обучения и использования методов

2.4.1 Optimal Transportation

Мы рассматриваем две нейронных сети с параметрами $\boldsymbol{\theta}_A$ и $\boldsymbol{\theta}_B$ соответственно. В силу того, что существуют симметрии в пространстве параметров, а именно перестановка нейронов местами не изменит выхода нейронной сети, имеет смысл искать такую кривую, которая в дополнение к процедурам описанным выше будет сопостовлять отдельные нейроны двух сетей.

Для поиска такого сооответствия мы применим метод оптимального транспорта (OT) в пространстве параметров скрытых нейронов. Таким образом мы получим набор перестановок нейронов, который сопоставит их максимально близко. Для этой задачи мы используем готовую библиотеку [6].

Сам по себе оптимального транспорта не соединит два экстремума, а только переставит нейроны, пытаясь их поставить в соответствие друг с другом. Чтобы завершить построение кривой мы воспользуемся описанными выше методами Arc connection и Linear connection.

2.4.2 Weight Adjustment

Для сравнения мы рассмотрим другой метод. который основан на поэтапном соединение весов. Как мы помним модель с одним скрытым слоем записывается так:

$$\widehat{\mathbf{y}} = \mathbf{W}_2\phi(\mathbf{W}_1\mathbf{x}).$$

Положим, что $\boldsymbol{\theta}^A = (\mathbf{W}_1^A, \mathbf{W}_2^A)$ и $\boldsymbol{\theta}^B = (\mathbf{W}_1^B, \mathbf{W}_2^B)$. Тогда путь

$$\boldsymbol{\theta} = \boldsymbol{\theta}(t) = (\mathbf{W}_1(t), \mathbf{W}_2(t))),\ t \in [0, 1].$$

В этом методе предлагается менять матрицы снизу вверх, то есть

$$\mathbf{W}_1(t) = (1-t)\mathbf{W}_1^A + t\mathbf{W}_1^B.$$

Далее мы можем изменять веса второго слоя, так как мы уже имеем

$$\mathbf{W}_2(t)\phi(\mathbf{W}_1(t)\mathbf{x}) \approx \widehat{\mathbf{y}},$$

а значит решая эту систему линейных уравнений мы можем получить выражение для $\mathbf{W}_2(t)$:

$$\mathbf{W}_2(t) = \widehat{\mathbf{y}}^A \left[\phi(\mathbf{W}_1(t)\mathbf{x})\right]^+,$$

где $[.]^+$ — псевдообратная матрица.

2.5 Исходные данные и условия экспериментов

В своих экспериментах мы будем использовать открытый набор MNIST, состоящий из изображений рукописных цифр от 0 до 9 и их разметки [13]. Мы будем решать задачу классификации и в процессе оценки параметров модели минимизировать кросс-энтропию. Отслеживать качество построения кривой будем с помощью точности.

Глава 3

Результаты экспериментов

3.1 Результаты

Мы проводим эксперименты, в которых изучаем влияние априорного распределения и архитектуры сети на пространство параметров модели. Для этого мы с помощью различный методов попробуем построить кривые с низкой функцией ошибки, соединяющие локальные минимумы функции ошибки, для различных моделей и различных априорных распределений.

В этом разделе приведены результаты экпериментов в виде графиков и таблиц. Для всех графиков в этом разделе обозначения одинаковы: по оси абсцисс отложен параметр времени $t \in [0, 1]$; по оси ординат отложена ошибка классификации в процентах, то есть если в данный момент времени модель демонстрирует ошибку классификации 5%, то на графике мы отложим число 5, или нормированная ошибка классификации. Результаты дополнительных экспериментов приведены в приложениях 5.2 и 5.1.

Мы сравниваем результаты для нейронной сети с одним скрытым слоем, варьируя число нейронов в скрытом слое. Для построение кривых, соединяющих два локальных минимума в пространстве параметров, мы используем методы, описанны в разделе 2:

- lin — соединение отрезков
- arc — соединение дугой
- WA — подход с адаптацией параметров слоев, используется поверх метода lin и arc
- OT — подход с подбором соответствия между нейронами разных локальных оптимумов, используется поверх метода lin и arc

Наши эксперименты разбиты на три блока:

- В разделе 3.1.1 мы проверяем, что для всех архитектур и нейронных сетей методы более сложные, чем соединение минимумов прямой или дугой, способны найти кривую, соединияющую два локальных минимума функции ошибки.

- В разделе 3.1.2 мы исследуем, как меняется сложность пространства параметров с увеличением количества нейронов в модели. То есть, что при увеличении числа нейронов отыскать кривые с малым изменением функции потерь становится проще.

- В разделе 3.1.3 мы смотрим на то, как априорное распределение влияет на пространство параметров нейронной сети и как эта зависимость проявляется для моделей разной сложности.

3.1.1 Выбор метода для соединения кривой локальных минимумов функции ошибки

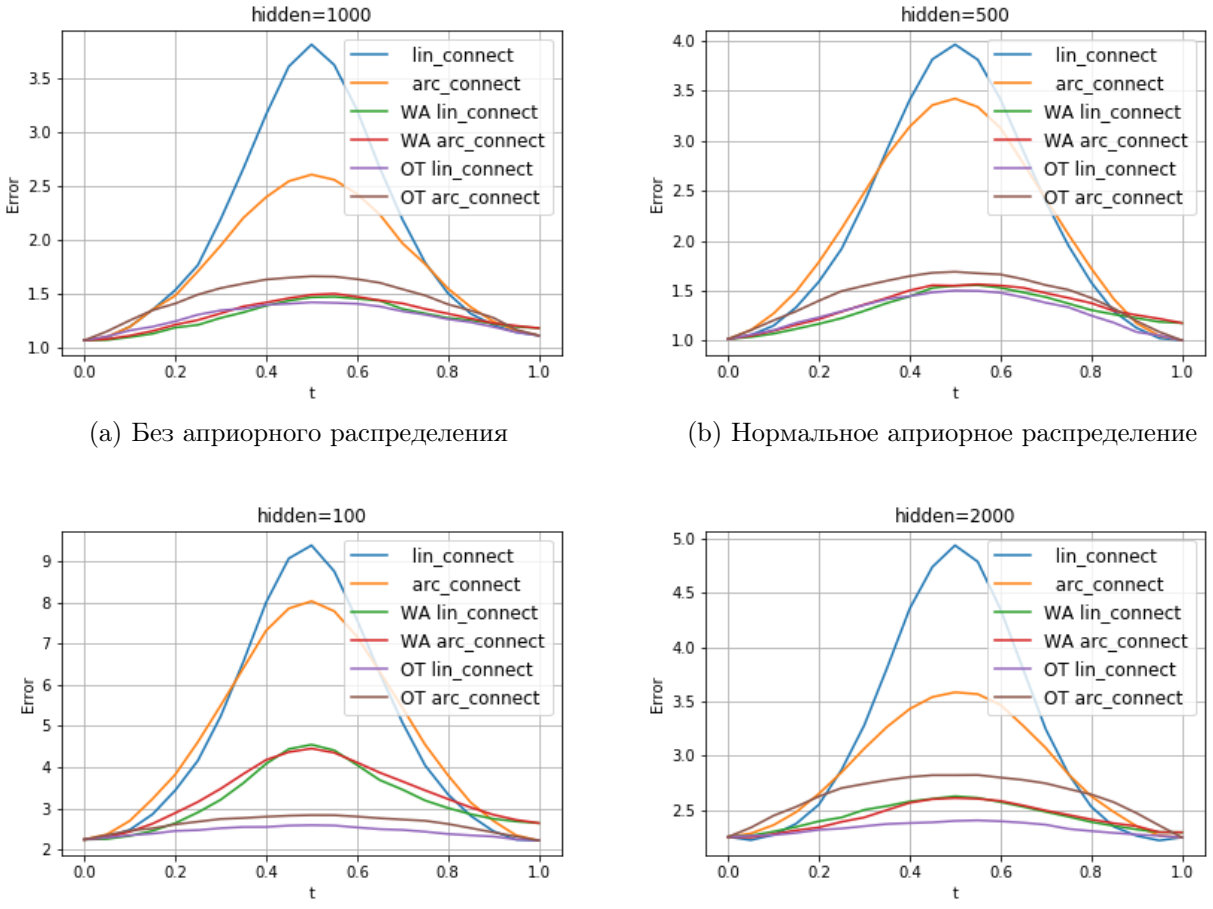
Для проверки этого утверждения предлагается сравнить качество построения кривых различнми методами. Полученные для разных априорных распределений и для разных априорных распределений приведены на рисунках 3.1 На графиках мы можем наблюдать, что качество построения кривых более сложными методами выше, ошибка вдоль кривой меньше. Этот эффект не зависит от числа нейронов и использованных априоных распределений.

Таким образом, для адекватного исследования свойств пространства параметров нужно использовать более сложные (WT lin, OT lin) подходы.

3.1.2 Отсутствие изолированности минимумов

В этом разделе посмотрим на то, как меняется динамика кривых в зависимости от числа нейронов в скрытом слое. Мы покажем что при росте числа нейронов изолированность минимумов падает. Для этого мы построим сравнение результатов работы различных методов для различного числа нейронов в скрытом слое.

Для нормировки точности мы будем делить все значения полученные в одном эксперименте на минимальное значение ошибки в этом эксперименте. Это сделано для того, чтобы наблюдать не только абсолютное изменение ошибки но и относительное, внутри данного эксперимента, и можно было сравнить кривые для разных моделей с разными ошибками.



(a) Без априорного распределения

(b) Нормальное априорное распределение

(c) Dropout 0.50

(d) Dropout 0.95

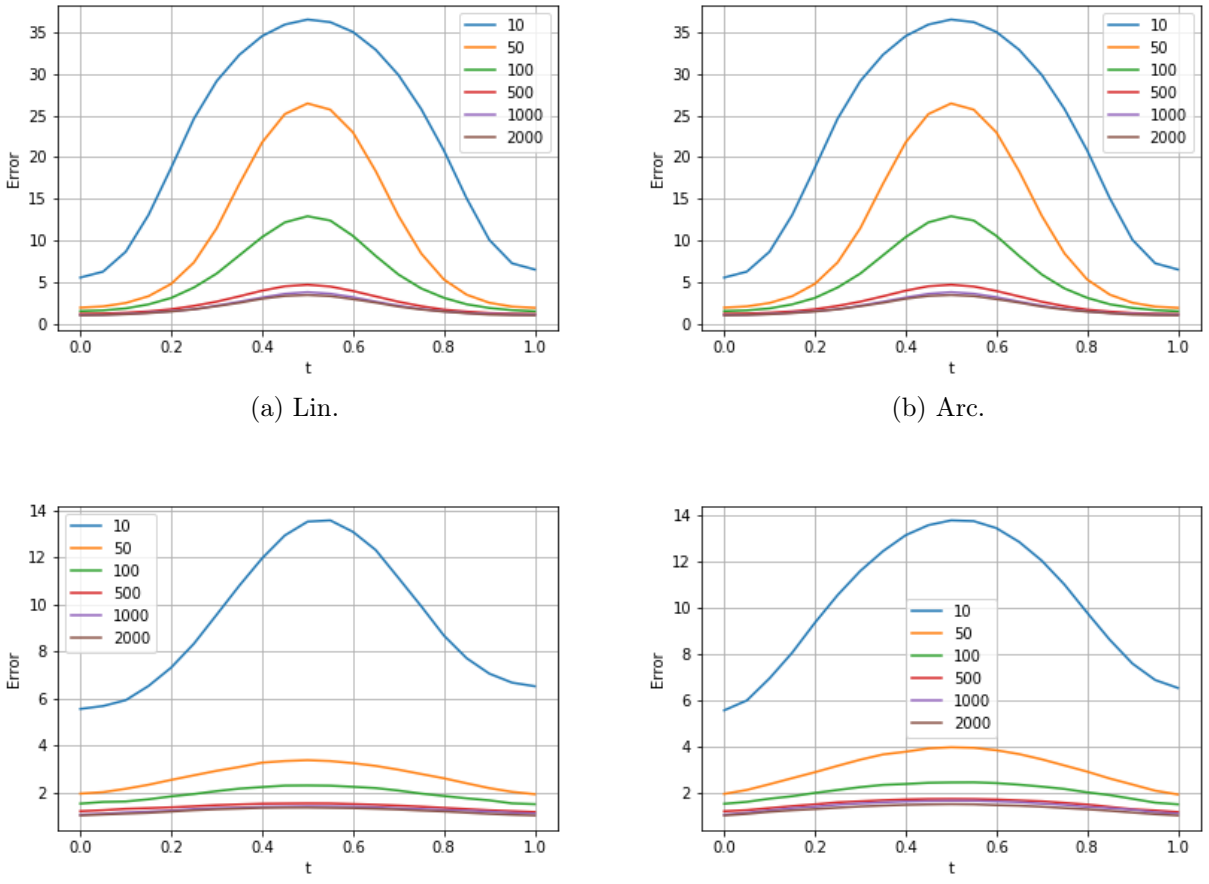
Рис. 3.1: Сравнение различных методов построение кривых, соединяющих локальные оптимумы для различных априорных распределений и различного числа нейронов

На рисунке 3.2 представлены результаты работы без нормировки. Здесь мы наблюдаем явную зависимость абсолютного показателя ошибки вдоль кривой от числа нейронов в скрытом слое. Чем больше нейронов тем меньше ошибка. Мы видим, что с увеличением числа нейронов стурктура пространства параметрво меняется и можно соединять разные локальные минимумы кривыми, вдоль которых качество модели почти не меняется.

Однако кажется что это сравнение не до конца корекно, так как обобщающая способность сети, содержащей в скрытом слое всего 10 нейронов, гораздо меньше чем та же способность для сети, содержащей 2000 нейронов.

Поэтому было принято решение отнорминовать значение вдоль оси ординат, путем деления на минимальное значение. Результат представлен на рисунке 3.3. Но и здесь мы наблюдаем ту же зависимость, кривые для большего числа нейронов лежат ниже чем для малого.

Для удобства можно проилюстрировать результат в виде таблиц 3.1 и 3.2. Наша



(a) Lin.

(b) Arc.

(c) OT+lin.

(d) OT+arc.

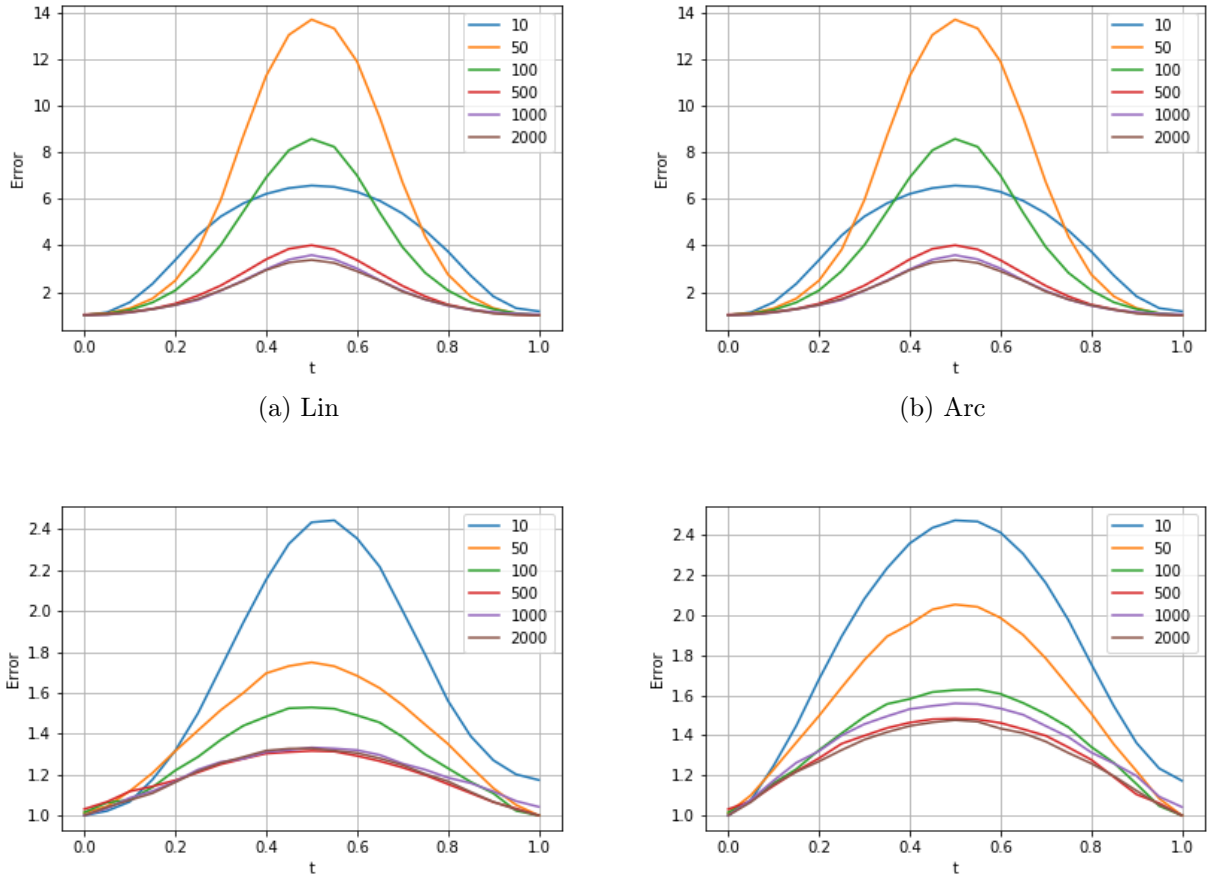
Рис. 3.2: Сравнение архитектур без проведения нормировки ошибок для разных методов построения соединящих локальные минимумы кривых

гипотеза будет означать что в каждой строке значение будет убывать слева направо, что и наблюдается на практике для большинства методов.

3.1.3 Влияние априорного распределения на структуру пространства параметров

В этой части мы хотим выявить влияние априорного распределения на пространство параметров. Для этого мы на одном графике отложим одних и тех же методов, но для различных априорных распределений параметров нейронной сети.

Полученные результаты с нормированной ошибкой приведены на рисунке 3.4. Мы видим, что во всех случаях зеленая линия, которая соответвует отсутвию априорного распределения, лежит не ниже чем все остальные, а это значит что пространство действительно становится более однордным и гладким, а минумумы менее изолированными. Кро-



(a) Lin

(b) Arc

(c) OT+lin

(d) OT+arc

Рис. 3.3: Сравнение различных архитектур нейронных сетей (с учетом нормировки внутри эксперимента)

ме того, если мы увеличиваем силу априорного распределения, увеличивая вероятность dropout с 0.5 до 0.95, кривая опускается еще ниже.

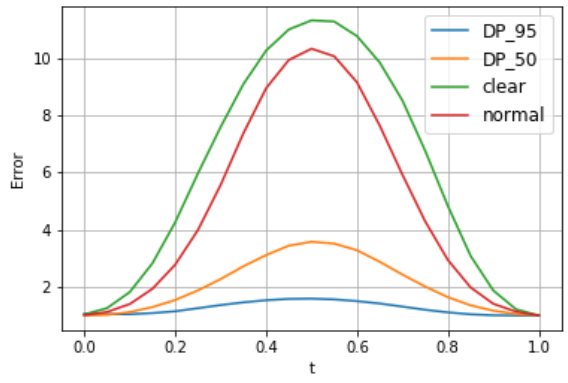
Однако, для большого числа нейронов ввдение жесткого априорного распределения приводит не только к сглаживанию пространства, но и к ухудшению качества классификации модели из-за дополнительных ограничений. Это видно на рисунке, на котором приведены ненормированные ошибки 3.5.

Число нейронов	10	50	100	500	1000	2000
Linear	45.74	24.35	10.83	4.84	3.83	4.17
Arc	35.94	21.58	9.53	4.21	3.39	2.98
Linear + Weight Adjustment	17.3	5.91	3.79	2.46	2.24	2.11
Arc + Weight Adjustment	17.11	5.77	3.71	2.43	2.23	2.11
Linear + OT	12.92	4.02	2.86	2.38	2.11	2.24
Arc + OT	13.77	4.81	3.5	2.59	2.36	2.37

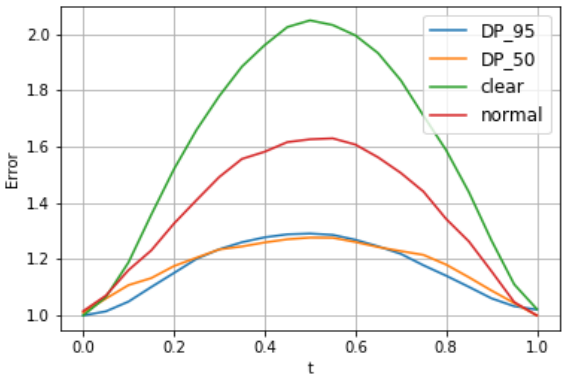
Таблица 3.1: Максимальная ошибка на кривой без использования априорного распределения (без учетом нормировки внутри эксперимента)

Число нейронов	10	50	100	500	1000	2000
Linear	8.35	12.58	7.73	4.83	4.02	4.85
Arc	6.56	11.15	6.8	4.2	3.56	3.47
Linear + Weight Adjustment	2.76	2.94	2.7	2.42	2.31	2.45
Arc + Weight Adjustment	2.65	2.87	2.65	2.39	2.30	2.45
Linear + OT	2.36	2.08	2.04	2.38	2.21	2.60
Arc + OT	2.52	2.49	2.5	2.59	2.48	2.76

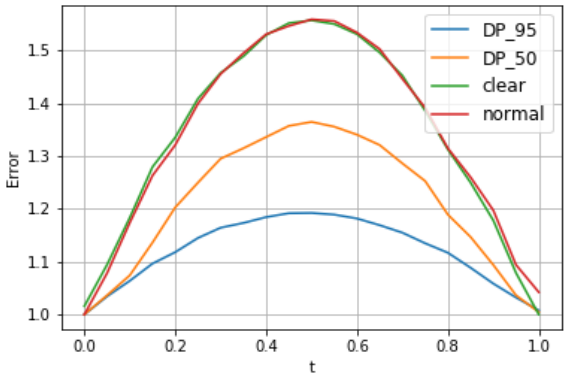
Таблица 3.2: Максимальная нормированная ошибка на кривой без использования априорного распределения



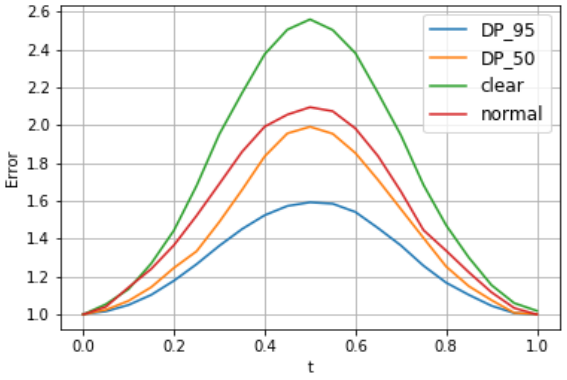
(a) Arc (hid = 50)



(b) OT+Arc (hid = 100)

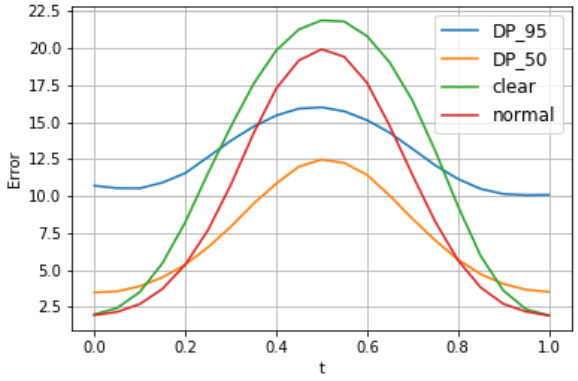


(c) OT+arc (hid = 1000).

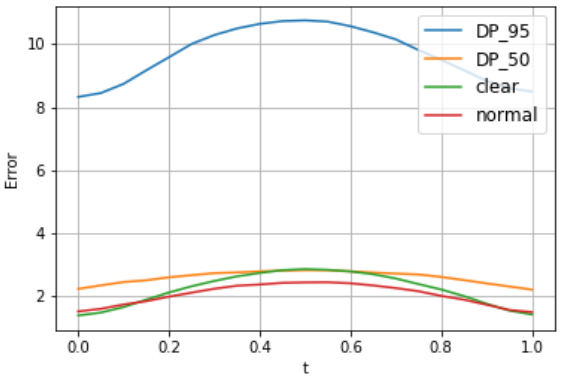


(d) Arc (hid = 2000).

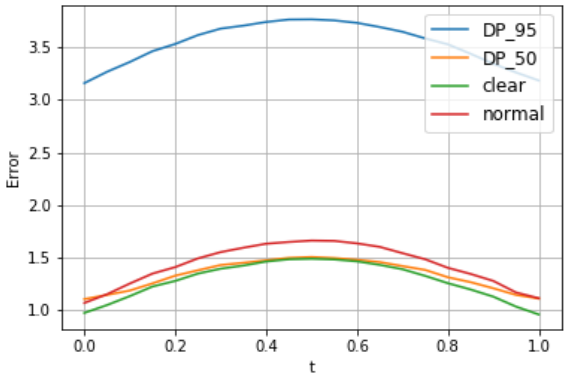
Рис. 3.4: Сравнение априорных распределений (с учетом нормировки внутри эксперимента)



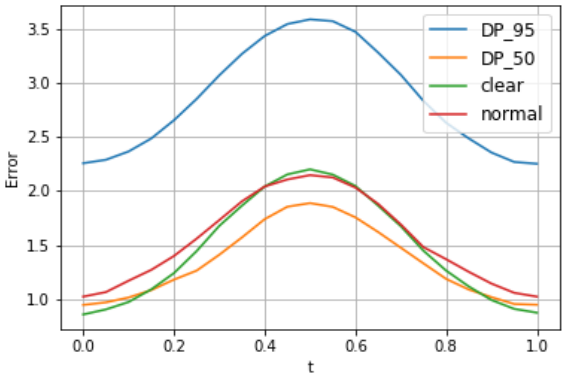
(a) Arc (hid = 50)



(b) OT+Arc (hid = 100)



(c) OT+arc (hid = 1000).



(d) Arc (hid = 2000).

Рис. 3.5: Сравнение априорных распределений (без учета нормировки внутри эксперимента)

Глава 4

Заключение

В ходе работы было изучено пространство параметров нейронных сетей с одним скрытым слоем в обычной и Байесовской постановке.

в проведенных экспериментах мы показали, что, что для всех архитектур и нейронных сетей методы более сложные, чем соединение минимумов прямой или дугой, способны найти кривую, соединияющую два локальных минимума функции ошибки, с малым изменением функции ошибки вдоль кривой.

Мы показали, что с увеличением количества нейронов локальные минимумы функции потерь перестают быть изолированы. В этом смысле мы получаем более простое пространство параметров с увеличением сложности модели.

Далее мы посмотрели на то, как априорное распределение влияет на пространство параметров нейронной сети. Оно делает пространство более гладкие и улучшает связанность локальных оптимумов, причем чем «сильнее» априорное распределение, чем более гладкое пространство мы получаем.

Глава 5

Дополнение

5.1 Таблицы

5.1.1 Ненормированные

Max accuracy loss without prior						
model	10	50	100	500	1000	2000
Linear	45.74	24.35	10.83	4.84	3.83	4.17
Arc	35.94	21.58	9.53	4.21	3.39	2.98
Linear + Weight Adjustment	17.3	5.91	3.79	2.46	2.24	2.11
Arc + Weight Adjustment	17.11	5.77	3.71	2.43	2.23	2.11
Linear + OT	12.92	4.02	2.86	2.38	2.11	2.24
Arc + OT	13.77	4.81	3.5	2.59	2.36	2.37

Max accuracy loss with prior						
model	10	50	100	500	1000	2000
Linear	36.8	26.05	12.44	4.92	4.23	4.0
Arc	34.57	19.52	8.3	3.94	3.43	2.88
Linear + Weight Adjustment	16.32	5.85	3.61	2.47	2.3	2.18
Arc + Weight Adjustment	16.16	5.76	3.53	2.42	2.33	2.2
Linear + OT	13.49	3.95	3.09	2.42	2.31	2.2
Arc + OT	13.68	4.38	3.1	2.53	2.5	2.37

Max accuracy loss with DropOut 0.5						
model	10	50	100	500	1000	2000
Linear	41.21	14.17	9.68	3.57	3.1	2.75
Arc	28.75	12.64	8.29	3.7	3.36	2.6
Linear + Weight Adjustment	20.31	7.49	4.57	2.57	2.23	2.06
Arc + Weight Adjustment	19.65	7.43	4.43	2.57	2.23	2.09
Linear + OT	15.03	4.61	3.09	2.17	2.05	2.09
Arc + OT	15.34	4.66	3.43	2.28	2.29	2.25

Max accuracy loss with DropOut 0.95						
model	10	50	100	500	1000	2000
Linear	78.04	23.01	19.05	7.99	6.09	5.2
Arc	64.97	16.73	11.35	6.29	4.99	3.8
Linear + Weight Adjustment	62.22	13.22	9.81	5.21	3.81	2.99
Arc + Weight Adjustment	59.87	13.14	10.16	5.24	3.89	2.96
Linear + OT	60.55	11.59	9.4	4.92	3.64	2.86
Arc + OT	57.16	16.44	11.34	5.28	4.01	3.02

5.1.2 Нормированные

normalazed max accuracy loss without prior						
model	10	50	100	500	1000	2000
Linear	8.35	12.58	7.73	4.83	4.02	4.85
Arc	6.56	11.15	6.8	4.2	3.56	3.47
Linear + Weight Adjustment	2.76	2.94	2.7	2.42	2.31	2.45
Arc + Weight Adjustment	2.65	2.87	2.65	2.39	2.30	2.45
Linear + OT	2.36	2.08	2.04	2.38	2.21	2.60
Arc + OT	2.52	2.49	2.5	2.59	2.48	2.76

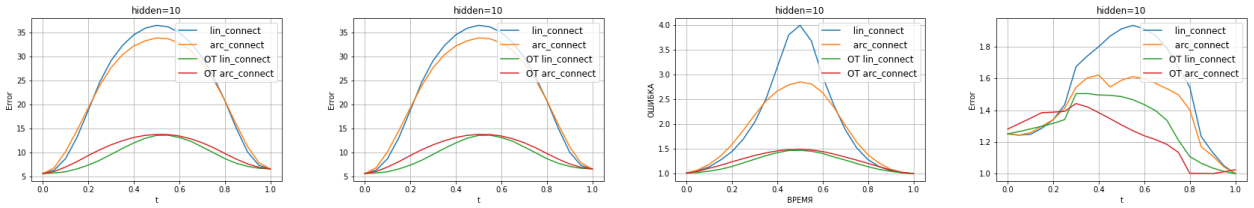
normalazed max accuracy loss with Normal prior						
model	10	50	100	500	1000	2000
Linear	6.62	13.49	8.26	4.2	3.98	3.9
Arc	6.22	10.11	5.51	3.36	3.23	2.81
Linear + Weight Adjustment	2.64	2.99	2.36	2.04	2.16	2.13
Arc + Weight Adjustment	2.45	2.94	2.31	2.0	2.19	2.15
Linear + OT	2.43	2.04	2.05	2.07	2.17	2.15
Arc + OT	2.46	2.27	2.06	2.16	2.35	2.31

normalazed max accuracy loss with DropOut 0.50						
model	10	50	100	500	1000	2000
Linear	1.94	2.28	2.29	1.8	1.95	2.34
Arc	1.62	1.66	1.36	1.42	1.58	1.69
Linear + Weight Adjustment	1.62	1.23	1.18	1.16	1.21	1.32
Arc + Weight Adjustment	1.53	1.23	1.22	1.17	1.23	1.31
Linear + OT	1.51	1.15	1.13	1.11	1.15	1.27
Arc + OT	1.46	1.63	1.36	1.19	1.27	1.34

normalazed max accuracy loss with DropOut 0.95						
model	10	50	100	500	1000	2000
Linear	1.94	2.28	2.29	1.8	1.95	2.34
Arc	1.62	1.66	1.36	1.42	1.58	1.69
Linear + Weight Adjustment	1.53	1.23	1.18	1.16	1.21	1.32
Arc + Weight Adjustment	1.49	1.23	1.22	1.17	1.23	1.31
Linear + OT	1.51	1.15	1.13	1.11	1.15	1.27
Arc + OT	1.46	1.63	1.36	1.19	1.27	1.34

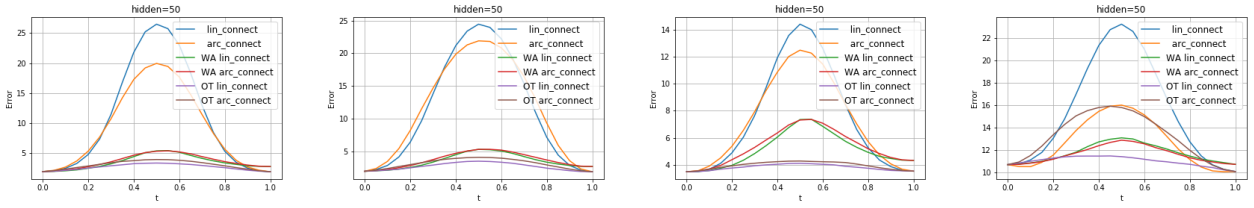
5.2 Графики

5.2.1 Ненормированные



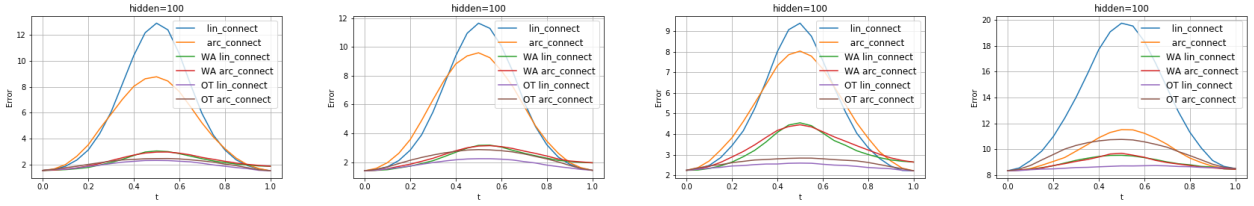
(a) Clear. (b) Normal prior. (c) DropOut 0.50. (d) DropOut 0.95.

Рис. 5.1: 10



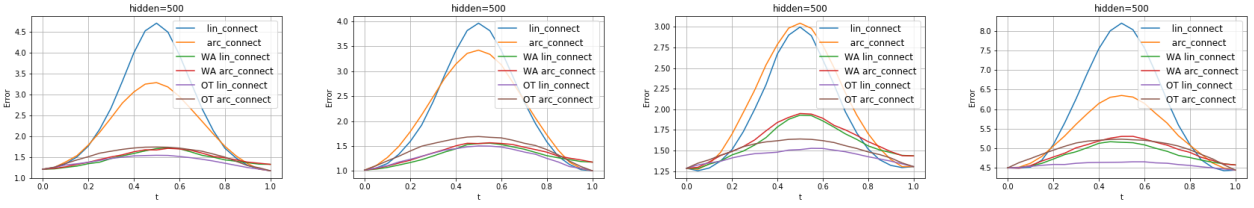
(a) Clear. (b) Normal prior. (c) DropOut 0.50. (d) DropOut 0.95.

Рис. 5.2: 50



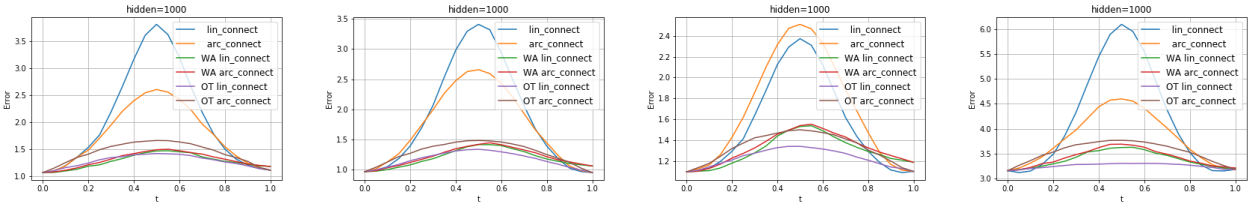
(a) Clear. (b) Normal prior. (c) DropOut 0.50. (d) DropOut 0.95.

Рис. 5.3: 100



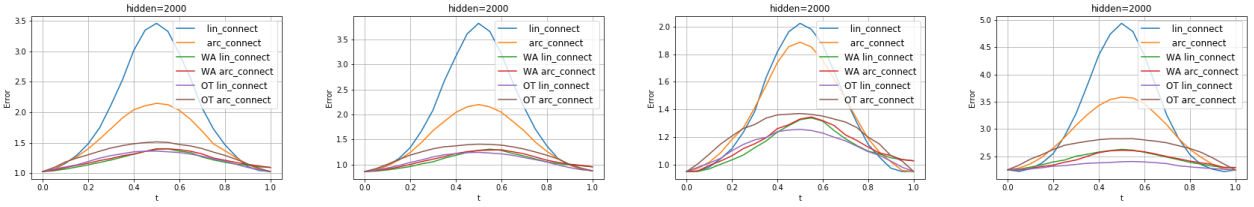
(a) Clear. (b) Normal prior. (c) DropOut 0.50. (d) DropOut 0.95.

Рис. 5.4: 500



(a) Clear. (b) Normal prior. (c) DropOut 0.50. (d) DropOut 0.95.

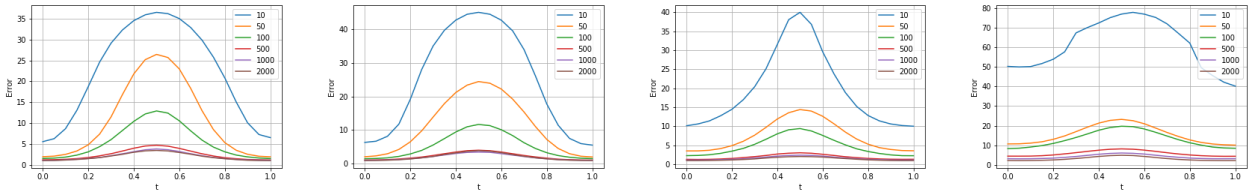
Рис. 5.5: 1000



(a) Clear. (b) Normal prior. (c) DropOut 0.50. (d) DropOut 0.95.

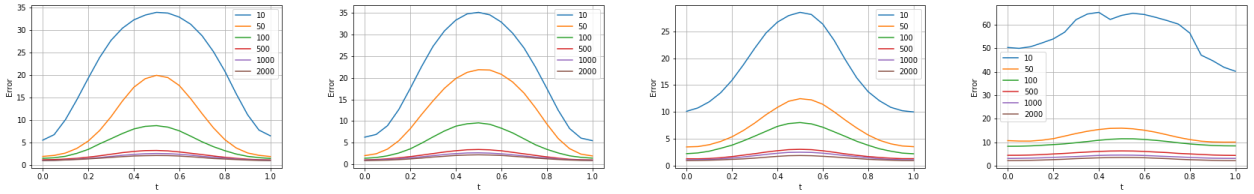
Рис. 5.6: 2000

5.2.2 Нормированные



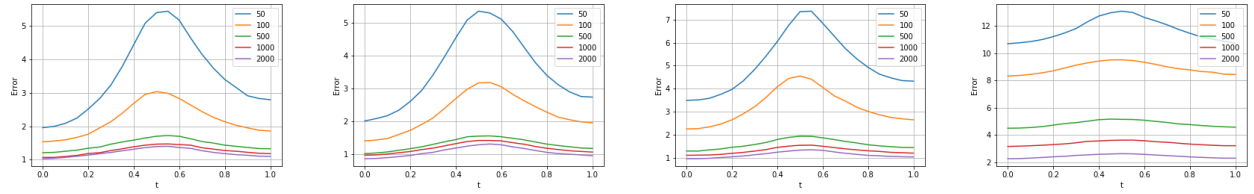
(a) Clear. (b) Normal prior. (c) DropOut 0.50. (d) DropOut 0.95.

Рис. 5.7: Сравнение LIN



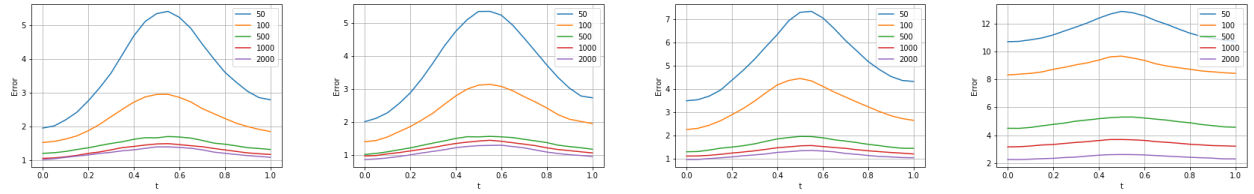
(a) Clear. (b) Normal prior. (c) DropOut 0.50. (d) DropOut 0.95.

Рис. 5.8: Сравнение ARC



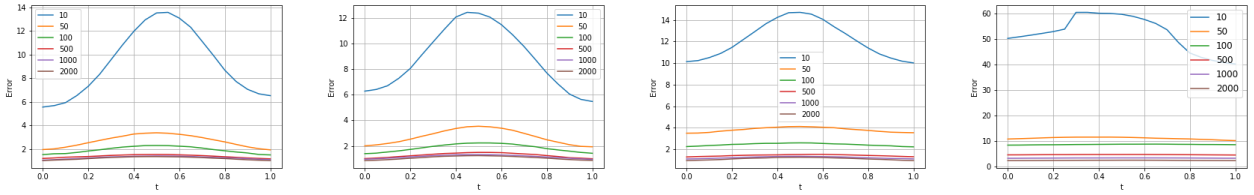
(a) Clear. (b) Normal prior. (c) DropOut 0.50. (d) DropOut 0.95.

Рис. 5.9: Сравнение WA+lin



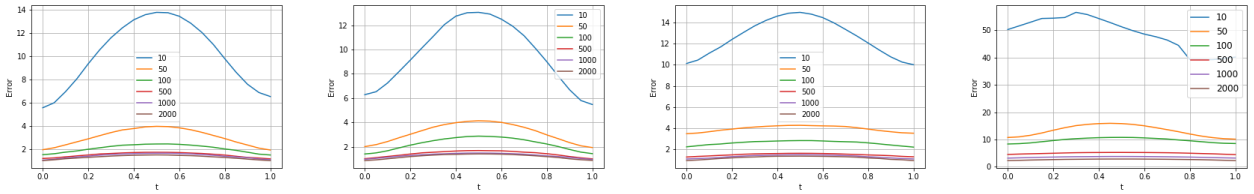
(a) Clear. (b) Normal prior. (c) DropOut 0.50. (d) DropOut 0.95.

Рис. 5.10: Сравнение WA + ARC



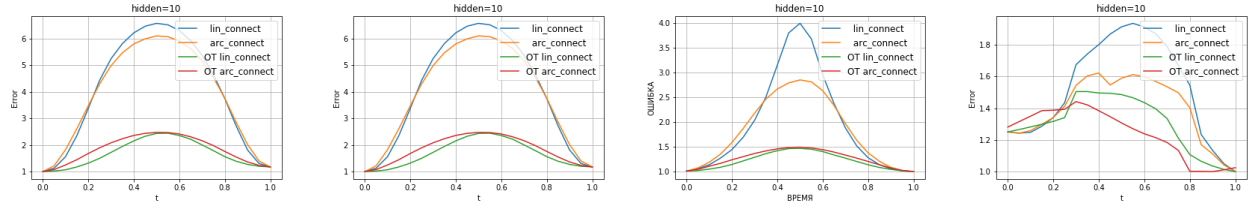
(a) Clear. (b) Normal prior. (c) DropOut 0.50. (d) DropOut 0.95.

Рис. 5.11: Сравнение OT + LIN



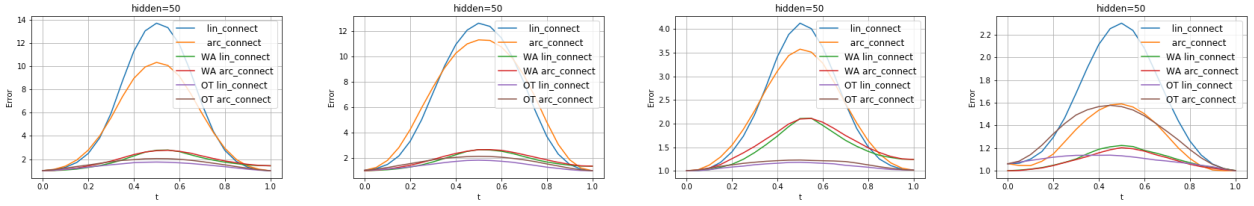
(a) Clear. (b) Normal prior. (c) DropOut 0.50. (d) DropOut 0.95.

Рис. 5.12: Сравнение OT + ARC



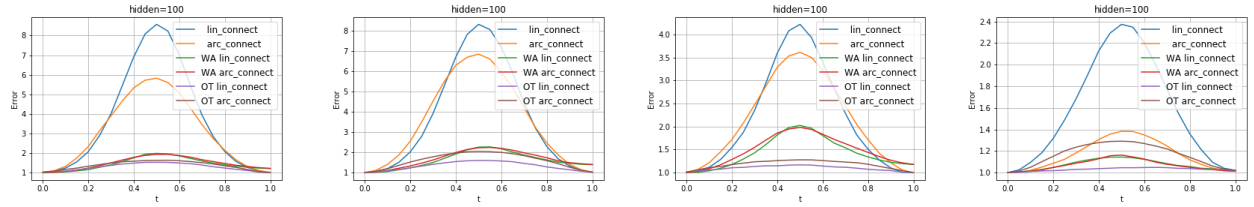
(a) Clear. (b) Normal prior. (c) DropOut 0.50. (d) DropOut 0.95.

Рис. 5.13: 10n



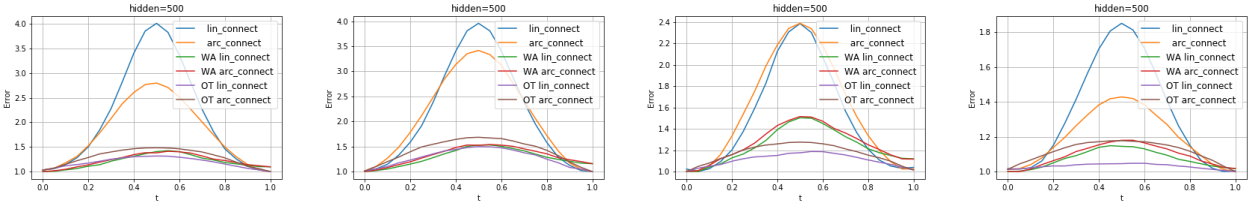
(a) Clear. (b) Normal prior. (c) DropOut 0.50. (d) DropOut 0.95.

Рис. 5.14: 50n



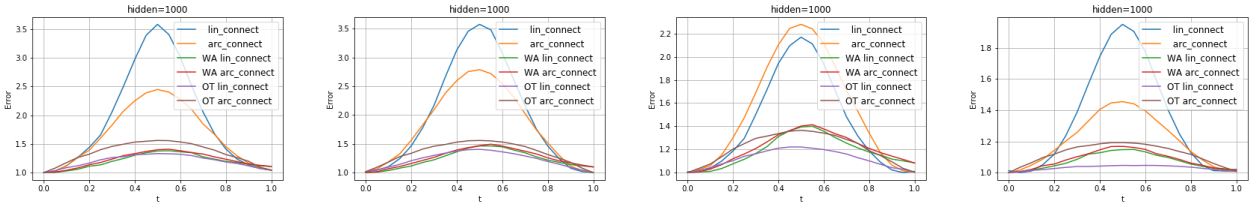
(a) Clear. (b) Normal prior. (c) DropOut 0.50. (d) DropOut 0.95.

Рис. 5.15: 100n



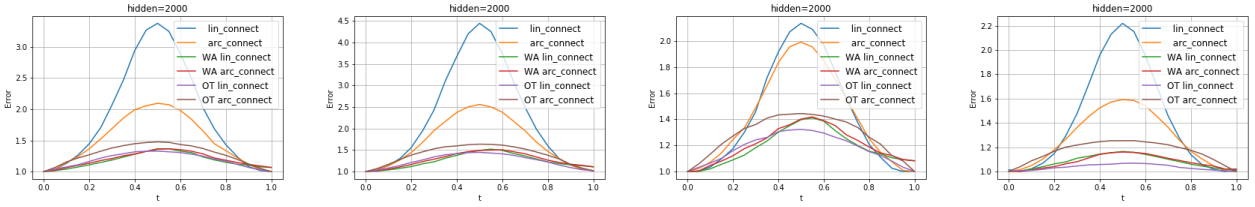
(a) Clear. (b) Normal prior. (c) DropOut 0.50. (d) DropOut 0.95.

Рис. 5.16: 500n



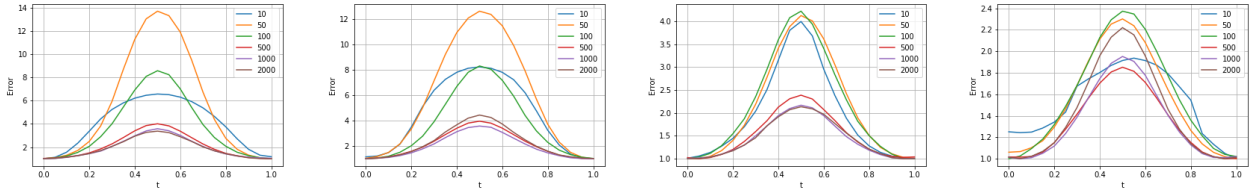
(a) Clear. (b) Normal prior. (c) DropOut 0.50. (d) DropOut 0.95.

Рис. 5.17: 1000 n



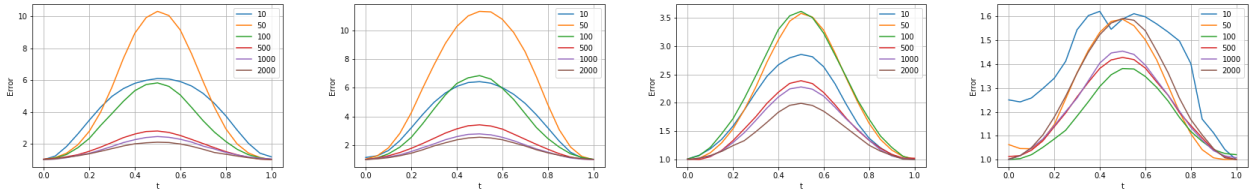
(a) Clear. (b) Normal prior. (c) DropOut 0.50. (d) DropOut 0.95.

Рис. 5.18: 2000n



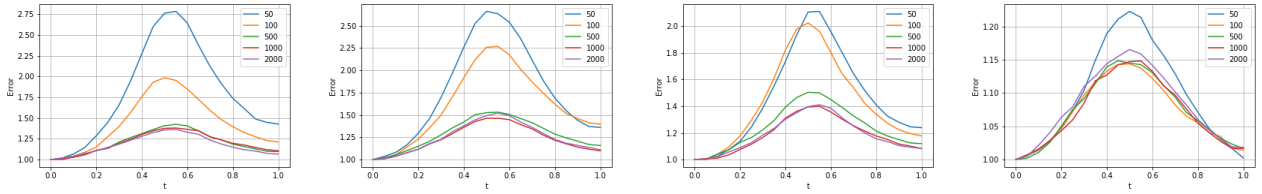
(a) Clear. (b) Normal prior. (c) DropOut 0.50. (d) DropOut 0.95.

Рис. 5.19: Сравнение LIN n



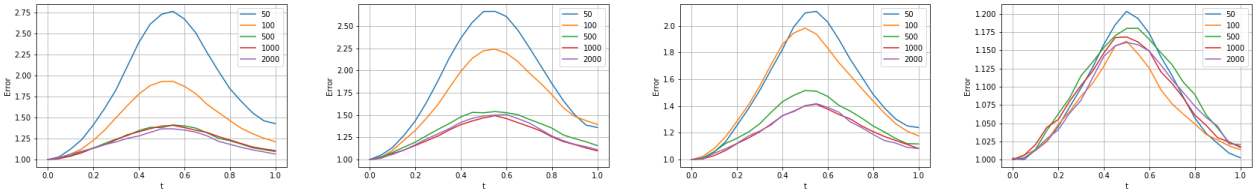
(a) Clear. (b) Normal prior. (c) DropOut 0.50. (d) DropOut 0.95.

Рис. 5.20: Сравнение ARC n



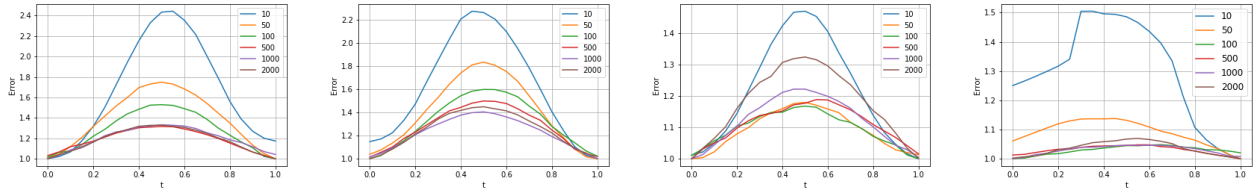
(a) Clear. (b) Normal prior. (c) DropOut 0.50. (d) DropOut 0.95.

Рис. 5.21: Сравнение WA+lin n



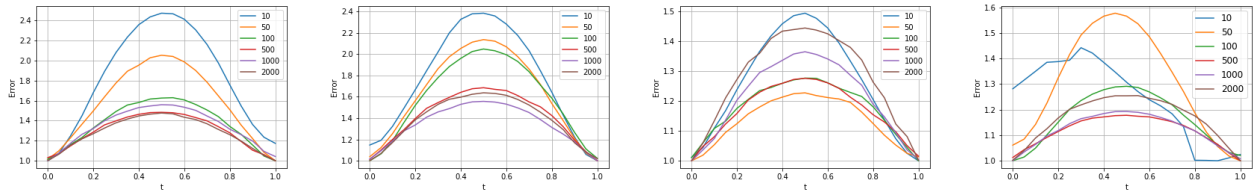
(a) Clear. (b) Normal prior. (c) DropOut 0.50. (d) DropOut 0.95.

Рис. 5.22: Сравнение WA + ARC n



(a) Clear. (b) Normal prior. (c) DropOut 0.50. (d) DropOut 0.95.

Рис. 5.23: Сравнение OT + LIN n



(a) Clear. (b) Normal prior. (c) DropOut 0.50. (d) DropOut 0.95.

Рис. 5.24: Сравнение OT + ARC n

Список литературы

[1] Martın Abadi и др. «TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems». в: *CoRR* abs/1603.04467 (2016). arXiv: 1603.04467. URL: http://arxiv.org/abs/1603.04467.

[2] Mikhail Belkin и др. «Reconciling modern machine-learning practice and the classical bias–variance trade-off». в: *Proceedings of the National Academy of Sciences* 116.32 (2019), с. 15849—15854.

[3] Léon Bottou. «Stochastic gradient descent tricks». в: *Neural networks: Tricks of the trade*. Springer, 2012, с. 421—436.

[4] E Burnaev, A Zaytsev и V Spokoiny. «The Bernstein-von Mises theorem for regression based on Gaussian processes». в: *Russ. Math. Surv* 68.5 (2013), с. 954—956.

[5] Laurent Dinh, Jascha Sohl-Dickstein и Samy Bengio. «Density estimation using Real NVP». в: *CoRR* abs/1605.08803 (2016). arXiv: 1605.08803. URL: http://arxiv.org/abs/1605.08803.

[6] R'emi Flamary и Nicolas Courty. *POT Python Optimal Transport library*. 2017. URL: https://pythonot.github.io/.

[7] Yarin Gal и Zoubin Ghahramani. «Bayesian convolutional neural networks with Bernoulli approximate variational inference». в: *arXiv preprint arXiv:1506.02158* (2015).

[8] Yarin Gal и Zoubin Ghahramani. *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning*. 2015. arXiv: 1506.02142 [stat.ML].

[9] Timur Garipov и др. «Loss surfaces, mode connectivity, and fast ensembling of dnns». в: *Advances in Neural Information Processing Systems*. 2018, с. 8789—8798.

[10] W Ronny Huang и др. «Understanding generalization through visualizations». в: *arXiv preprint arXiv:1906.03291* (2019).

[11] Diederik P. Kingma, Tim Salimans и Max Welling. «Improving Variational Inference with Inverse Autoregressive Flow». в: *CoRR* abs/1606.04934 (2016). arXiv: 1606.04934. URL: http://arxiv.org/abs/1606.04934.

[12] Diederik P Kingma и Jimmy Ba. «Adam: A method for stochastic optimization». в: *arXiv preprint arXiv:1412.6980* (2014).

[13] Yann LeCun и Corinna Cortes. «MNIST handwritten digit database». в: (2010). URL: http://yann.lecun.com/exdb/mnist/.

[14] Jaehoon Lee и др. *Deep Neural Networks as Gaussian Processes*. 2017. arXiv: 1711.00165 [stat.ML].

[15] Gregory Naitzat, Andrey Zhitnikov и Lek-Heng Lim. «Topology of deep neural networks». в: *arXiv preprint arXiv:2004.06093* (2020).

[16] Preetum Nakkiran и др. «Deep double descent: Where bigger models and more data hurt». в: *arXiv preprint arXiv:1912.02292* (2019).

[17] Ivan Anokhin (Skolkovo Institute of Science, Dmitry Yarotsky (Skolkovo Institute of Science Technology) и Technology). *Low-loss connection of weight vectors: distribution-based approaches*. 2020.