Книга Kevin Swingler "Applying Neural Networks. A practical Guide" (перевод Ю.П.Маслобоева)
Содержание

2. Кодирование и пребразование данных

2.1. Введение

Нейронные сети способны обрабатывать данные, полученные из самых разных источников. Однако, данные должны быть представлены в определенном формате. Более того, вид представления данных оказывает существенное влияние на ход обучения сети. Следовательно, очень важно выбрать метод подготовки данных (кодирования) перед предъявлением их сети. Разработка схемы кодирования данных выполняется перед сбором данных, так как важно знать, что Вы собираетесь делать с данными еще до того, как они будут получены.

Для того, чтобы обучить многослойный перцептрон (МП) решению определенной задачи, задача должна быть сформулирована в терминах набора входных векторов и ассоциированных с ними эталонных выходных значений (эталонов). На практике эта рекомендация мало что даёт при постановке проблем, которые могут быть решены с помощью МР, так как закодировать можно всё. В этом разделе представлен обзор подходов к выбору методов кодирования для нейронно-сетевых проектов а в последующих разделах это подходы обсуждаются более подробно.

Следует различать процедуру кодирования данных, под которой понимают процесс преобразования сырых (необработанных) данных к виду, удобному для тренировки сети, и процедуру преобразования данных, под которой понимают манипулирование или сырыми данными или их кодированными эквивалентами. Преобразование может быть использовано для того, чтобы выделить некоторые аспекты данных или наиболее важные элементы данных. Это может быть сделано например для представления данных меньшим количеством размерностей или другим типом представления. Использование преобразования Фурье является одним из примеров такого Преобразования. Кодирование данных может быть использовано после преобразования, так как кодирование - это процедура, с помощью которой данные приводятся к виду, в котором они могут использоваться сетью.

2.1.1. Два принципа представления данных в нейронных сетях

При выборе представления данных для их использования в сети необходимо иметь в виду два принципа (аспекта) - количество переменных (определяет размерность данных) и содержательность данных (содержание информации, которую необходимо получить с помощью сети). Из соображений сложности и скорости обработки часто бывает необходимо минимизировать количество переменных (входных и выходных) с которыми будет работать сеть. Проблемы также могут быть связаны с разрешением предъявляемых данных: чем выше разрешение, тем сложнее должна быть сеть и, следовательно, возрастает время тренировки и количество данных, необходимых для тренировки. Это приводит нас к второму аспекту представления тренировочных данных - ясность, содержательность. При ограничении количества входных переменных, возникает необходимость обеспечить, чтобы используемые переменные содержали информацию, достаточную для того, чтобы обеспечить обучение сети.

2.1.2. Подготовка данных

При подготовке набора сырых входных данных для тренировки нейронной сети нужно выполнить следующие шаги:

1. **Классификация данных по типу**. Каждый вход или выход нейронной сети может изменяться либо непрерывным образом, либо представлять набор дискретных значений. Тип каждой переменной необходимо идентифицировать, так как каждый тип требует своего подхода при кодировании.
2. **Вычисление глобальной статистики**. Для некоторых методов подготовки данных необходимо знать статистические свойства каждой переменной, используемой для тренировки. Для непрерывных данных должны быть вычислены среднее значение, стандартное отклонение, максимум и минимум. Для дискретных данных необходимо определить количество различающихся случаев (значений, которые может принимать каждая из переменных или категорий, из которых они выбираются).
3. **Удаление выбросов**. Девяносто пять процентов данных, распределенных по нормальному закону, находятся внутри интервала ограниченного удвоенным значением стандартного отклонения в окрестности среднего значения. Исключение из рассмотрения данных, находящихся вне указанного интервала является простейшим способом удаления выбросов. Удаление выбросов способствует более успешному обучению сети. Если же эти значения являются принципиальными для исследуемой задачи, тогда данные должны быть кодированы и собраны таким образом, чтобы правильно отобразить их важность.
4. **Количество тренировочных данных**. Существует множество факторов, влияющих на количество данных, необходимых для тренировки сети. Один из главных факторов - размерность данных. Чем больше переменных, тем больше данных требуется для тренировки сети. Эта проблема может быть решена либо увеличением набора данных, либо уменьшением их размерности.
5. **Построение выборки из набора данных.** Для очень больших наборов данных не всегда можно выполнить комплексный анализ на полном наборе. В таких случаях допустимо построить случайную выборку из данных, на основании которой и сделать заключения о методе предварительной обработки данных. Выборка должна быть случайной, должна включать данные со значениями из всего диапазона изменения и включать достаточно данных, чтобы быть репрезентативной. После построения выборки необходимо сравнить среднее значение и диапазон изменения с аналогичными параметрами полного набора, чтобы убедиться в том, что выборка выполнена правильно. Следующие процедуры могут быть выполнены на выборке, а не на полном наборе данных. Тем не менее сеть необходимо тренировать на полном наборе данных.
6. **Проверка качества.** Для построения хорошо сбалансированной модели важно, чтобы распределение тренировочных данных было гладким. Желательно также выполнить статистические тесты, чтобы убедиться в том, что данные содержат информацию, достаточную для решения поставленной задачи.
7. **Понижение размерности.** Нейронная сеть с большим количеством входных элементов содержит большое количество весовых коэффициентов (весов). Сеть с большим количеством весов требует (иногда экспоненциально) большого количества тренировочных данных. Аналогично, теория дискретизации утверждает, что имеет место экспоненциальная зависимость между размерностью данных и количеством точек дискретизации для корректного описания данных. Уменьшение количества входов сети позволяет использовать меньше данных, обеспечивая приемлемый уровень сложности сети.
8. **Шкалирование данных.** Выходные данные должны быть шкалированы (преобразованы) к диапазону, который соответствует диапазону выходных значений сжимающей функции активации выходного слоя. Сигмоидальная функция, например, имеет диапазон выходных значений от 0 до 1. Часто бывает удобным привести входные данные к тому же диапазону. Шкалирование может быть линейным или нелинейным, в зависимости от распределения данных. Наиболее часто используемые методы шкалирования - линейное, логарифмическое и "мягкое" (softmax). Можно также разделить данные на несколько диапазонов с различными факторами шкалирования.
9. **Кодирование данных.** Если данные были преобразованы, может возникнуть необходимость кодировать их перед предъявлением сети. Шкалированные численные данные могут быть использованы непосредственно, либо после дополнительного преобразования. Категорийные данные всегда необходимо кодировать.

2.2. Классификация данных по типу

Все данные можно разбить на два больших класса - численные и категорийные. Численные данные это данные, которые непрерывно изменяются в некотором диапазоне и к которым применимо понятие расстояния (разности) между двумя значениями. Категориальные данные не могут быть выстроены в непрерывный ряд, каждое значение дискретно и невозможно ввести понятие расстояния.

Рассмотрим, каким образом можно кодировать числа и категории, а также каким образом можно трактовать числа как категории с использованием расширенного кодирования. Рассмотрим специальный случай, когда выбор между кодированием чисел распределенным (как числовые величины) или локальным (с расширенным кодированием) методами не столь очевиден. Это случай целых чисел. Хотя мы и отметили, что величины, которые описываются непрерывным набором значений, необходимо рассматривать как численные, однако часто данные, представленные целыми числами и удовлетворяющие указанным критериям предпочтительно кодировать как категории.

В качестве примера рассмотрим переменную, описывающую количество детей. Она может принимать только целые значения - 0,1,2,3: и кажется непрерывной. Однако, диапазон значений очень узкий и дробные значения бессмысленны. Используя схему расширенного кодирования, которая характеризуется значениями активации 0 и 1, можно преобразовать одну численную переменную в набор категориальных переменных. Очевидно, что диапазон изменения должен быть заранее известен, так как мы будем использовать одну единицу на значение, но такое представление переменной обладает рядом преимуществ.

Во-первых, очевидной становится дискретная природа разностей между числами. Изменение величины числа теперь вносит изменения более чем в одно измерение в модели, что облегчает задачу разделения входного пространства. Во вторых, если переменная является выходной переменной, сеть выдает величины, соответствующие вероятности того, что ответом является каждая из заданных возможностей. Это показывает нам, в какой степени мы можем доверять выходам сети. Наконец, исключена вероятность получения бессмысленных ответов типа - число детей = 0,3.

Неудобство представления целых чисел в виде категорий состоит в увеличении размера сети и, следовательно, требуется больше времени на тренировку сети и больше данных для тренировки.

2.3. Исходные статистические вычисления

Некоторые манипуляции с данными, которые будут обсуждаться, требуют вычисления простейших статистических мер тренировочного набора данных. Среди них: среднее значение, максимум, минимум, стандартное отклонение в случае численных данных и количество различных величин (категорий) в случае категориальных данных.

2.3.1. Удаление выбросов

Величины, которые слишком далеки от среднего значения переменной, могут оказать влияние на сеть, которое не адекватно ошибке, которую они вызывают. Этот эффект проявляется сильней, если выбросы обусловлены шумом. По этой причине желательно удалять такие выбросы до начала тренировки. Среднее значение и величина стандартного отклонения могут быть использованы для выявления и удаления выбросов из тренировочного набора данных. Выбросы - это величины, которые слишком далеки от среднего значения и которые могут приводить к непропоционально большому эффекту в процессе тренировки. Выбросы - это значения, которые лежат далее, чем за заданное число стандартных отклонений от среднего. Два стандартных отклонения охватывают 95% нормально распределённых данных, а три стандартных отклонения охватывают 99%. В качестве критерия выброса часто используется критерий 2,5.

Следует помнить, что если некоторые переменные в заданной паре вход-выход являются выбросами, то необходимо исключить из рассмотрения весь пример. Это кажется очевидным, но если переменные хранятся в разных файлах, то легко об этом забыть и удалить данные из переменной в одном файле, оставив соответствующие величины в другом файле. Последствия такой ошибки заключаются в том, что векторы становятся несогласованными, а Ваши данные становятся неправильными.

Описанная выше простейшая методика удаления выбросов ограничена тем, что она работает для каждой из переменных независимо, в то время как выброс может трактоваться как неприемлемая комбинация величин, которые, будучи взятыми по отдельности, не выходят за пределы заданного критерия, но в комбинации выходят за пределы облака нормально распределённых данных. Такие выбросы часто обнаруживаются на графике первых двух или трех главных компонент набора данных, так что если Вы всерьёз озабочены удалением выбросов и Вы уверены, что они присутствуют в данных, то необходимо выполнить такой анализ.

Если данных мало, то удаленная величина может быть заменена некоторой другой величиной: при

2.1.2. Подготовка данных

При подготовке набора сырых входных данных для тренировки нейронной сети нужно выполнить следующие шаги:

1. **Классификация данных по типу**. Каждый вход или выход нейронной сети может изменяться либо непрерывным образом, либо представлять набор дискретных значений. Тип каждой переменной необходимо идентифицировать, так как каждый тип требует своего подхода при кодировании.
2. **Вычисление глобальной статистики**. Для некоторых методов подготовки данных необходимо знать статистические свойства каждой переменной, используемой для тренировки. Для непрерывных данных должны быть вычислены среднее значение, стандартное отклонение, максимум и минимум. Для дискретных данных необходимо определить количество различающихся случаев (значений, которые может принимать каждая из переменных или категорий, из которых они выбираются).
3. **Удаление выбросов**. Девяносто пять процентов данных, распределенных по нормальному закону, находятся внутри интервала ограниченного удвоенным значением стандартного отклонения в окрестности среднего значения. Исключение из рассмотрения данных, находящихся вне указанного интервала является простейшим способом удаления выбросов. Удаление выбросов способствует более успешному обучению сети. Если же эти значения являются принципиальными для исследуемой задачи, тогда данные должны быть кодированы и собраны таким образом, чтобы правильно отобразить их важность.
4. **Количество тренировочных данных**. Существует множество факторов, влияющих на количество данных, необходимых для тренировки сети. Один из главных факторов - размерность данных. Чем больше переменных, тем больше данных требуется для тренировки сети. Эта проблема может быть решена либо увеличением набора данных, либо уменьшением их размерности.
5. **Построение выборки из набора данных.** Для очень больших наборов данных не всегда можно выполнить комплексный анализ на полном наборе. В таких случаях допустимо построить случайную выборку из данных, на основании которой и сделать заключения о методе предварительной обработки данных. Выборка должна быть случайной, должна включать данные со значениями из всего диапазона изменения и включать достаточно данных, чтобы быть репрезентативной. После построения выборки необходимо сравнить среднее значение и диапазон изменения с аналогичными параметрами полного набора, чтобы убедиться в том, что выборка выполнена правильно. Следующие процедуры могут быть выполнены на выборке, а не на полном наборе данных. Тем не менее сеть необходимо тренировать на полном наборе данных.
6. **Проверка качества.** Для построения хорошо сбалансированной модели важно, чтобы распределение тренировочных данных было гладким. Желательно также выполнить статистические тесты, чтобы убедиться в том, что данные содержат информацию, достаточную для решения поставленной задачи.
7. **Понижение размерности.** Нейронная сеть с большим количеством входных элементов содержит большое количество весовых коэффициентов (весов). Сеть с большим количеством весов требует (иногда экспоненциально) большого количества тренировочных данных. Аналогично, теория дискретизации утверждает, что имеет место экспоненциальная зависимость между размерностью данных и количеством точек дискретизации для корректного описания данных. Уменьшение количества входов сети позволяет использовать меньше данных, обеспечивая приемлемый уровень сложности сети.
8. **Шкалирование данных.** Выходные данные должны быть шкалированы (преобразованы) к диапазону, который соответствует диапазону выходных значений сжимающей функции активации выходного слоя. Сигмоидальная функция, например, имеет диапазон выходных значений от 0 до 1. Часто бывает удобным привести входные данные к тому же диапазону. Шкалирование может быть линейным или нелинейным, в зависимости от распределения данных. Наиболее часто используемые методы шкалирования - линейное, логарифмическое и "мягкое" (softmax). Можно также разделить данные на несколько диапазонов с различными факторами шкалирования.
9. **Кодирование данных.** Если данные были преобразованы, может возникнуть необходимость кодировать их перед предъявлением сети. Шкалированные численные данные могут быть использованы непосредственно, либо после дополнительного преобразования. Категорийные данные всегда необходимо кодировать.

2.2. Классификация данных по типу

Все данные можно разбить на два больших класса - численные и категорийные. Численные данные это данные, которые непрерывно изменяются в некотором диапазоне и к которым применимо понятие расстояния (разности) между двумя значениями. Категориальные данные не могут быть выстроены в непрерывный ряд, каждое значение дискретно и невозможно ввести понятие расстояния.

2.2.1. Кодирование целых чисел

о корректности произведенного понижения размерности входных данных. Практическое преимущество метода состоит в том, что не требуется применения сложных математических процедур.

Выбор категорий выходных эталонов

Количество выходных классов можно уменьшить с использованием анализа, основанного на вычислении энтропии. Поскольку выход определяет каждую категорию как дискретное событие, мера предсказуемости каждой категории может быть увеличена на основе анализа энтропии каждого выхода, полученного при анализе каждого входа. Категории с высокими значениями энтропии хуже поддаются предсказанию. Методы, основанные на анализе энтропии, более подробно будут рассмотрены в последующих главах.

Выбор входных переменных на основе анализа энтропии.

Battiti [12] показал, как можно использовать теорию информации для выбора набора входных значений нейронной сети. Вычисляя взаимную информацию между каждой входной переменной и набором эталонных выходов тренировочного набора данных, он указал на возможность выбора единственной особенности с наибольшей предсказательной способностью для выходов. Следующие особенности могут быть отобраны по двум критериям. Новая особенность может предсказать кое что о выходах, но она не может предсказать ничего существенного об уже выбранных входных переменных. Другими словами, новая переменная должна содержать часть взаимной информации с выходами, которые должны быть созданы сетью, но меньшую часть взаимной информации с переменными, уже выбранными в качестве входов сети. Используя предложенный метод можно выбрать набор независимых переменных, которые хорошо предсказывают выходные данные. Метод не оптимальный, но позволяет добиться хороших результатов за короткое время. Выполнение строгих вычислений, необходимых для выбора оптимального набора переменных неосуществимо с точки зрения возможностей компьютеров. Применение информационной теории для оптимизации данных будет описано далее (раздел 5.3.2).

2.4.2. Проектирование в пространство с меньшей размерностью

Главные компоненты

Наиболее общим методом уменьшения количества входных переменных является выделение главных компонент из исходного набора данных. После того, как вычислена ковариационная матрица, можно комбинировать пары переменных с высокими значениями ковариантности таким образом, чтобы новые переменные описывали исходные данные как можно полнее. Это эквивалентно тому, как если бы мы нарисовали регрессионную кривую на основе исходных данных и представили новую точку как расстояние вдоль этой линии. Повторив эту процедуру N раз для N - мерного набора данных, получим новый набор точек, которые хорошо описывают исходные данные, но в другой системе координат. Преимущество новых координат состоит в том, что мы можем упорядочить новые координаты по отношению к вариабельности в исходном наборе данных. Мы можем просто отбросить исходный набор данных.

Полноценное понижение размерности с помощью метода главных компонент возможно при использовании ковариационной матрицы и при этом получаем набор векторов, называемых собственными векторами, которые могут быть использованы в качестве новой системы координат, на которую будут спроектированы исходные данные. Собственные векторы служат осями новой системы координат. Каждый собственный вектор характеризуется длиной, которая называется собственным значением. Процентное отношение суммы всех собственных значений, которое является вкладом отдельной величины показывает нам процентное отношение вариации данных, которые вычислены для соответствующего вектора. Хороший алгоритм использования главных компонент описан в книге *Numerical Recipes* [77].

Morris [67] обратил внимание на то, что этот метод имеет недостаток, обусловленный тем, что это линейный метод. Так что при попытке использовать его при анализе нелинейной системы можно получить неправильные результаты. Эта проблема имеет место в тех случаях, когда имеем дело с нелинейностями во входных данных. В случае, когда входные данные линейны, но их связь с выходами является нелинейной, проблемы не существует.

Авто-ассоциативные сети

Авто-ассоциативная сеть является решением проблемы, поднятой Morris [67], и представляет собой лёгкий в использовании нелинейный метод. Этот метод включает построение многослойного перцептрона, в котором в качестве эталонных выходов используются входы. Такая сеть с одним скрытым слоем, число элементов в котором меньше чем количество входов, может быть использована для выделения (экстракции) особенностей во входных данных. Значения весовых коэффициентов скрытого слоя можно использовать как входы для основной сети. Эта процедура отличается от простого добавления дополнительного слоя в сеть, который бы действовал как экстрактор особенностей, тем, что тренировать отдельную авто-ассоциативную сеть легче. При этом количество весов основной сети снижается за счет уменьшения размерности входных данных.

Недостатки проектирования данных

Если входные данные были преобразованы к другому набору переменных, то возникают трудности в интерпретации выходных данных сети по отношению к исходным входным данным. Это особенно важно в тех случаях, когда возникает необходимость использовать выходную информацию для того, чтобы внести изменения во входные данные.

Многие типы данных несут указания на способ уменьшения размерности. Например, звук и другие волновые процессы могут быть подвергнуты преобразованию Фурье, после чего можно использовать только компоненты из заданной полосы частот. Изображения можно сегментировать и затем кодировать. Текстовые данные о людях могут быть перекодированы в меньшее количество категорий:

2.5. Шкалирование данных

Мы уже отметили то обстоятельство, что эталонные значения выходов сети с сигмоидальной функцией должны лежать в диапазоне от нуля до единицы. Причиной этого является то, что выход сигмоидальной функции лежит в указанном диапазоне. Выходы различные функций активации лежат в различных диапазонах. Например, выходные значения функции гиперболического тангенса лежат в диапазоне от -1 до 1. Выходные значения линейной функции не ограничены.

В первых двух случаях необходимо, а в последнем желательно шкалировать данные таким образом, чтобы они не выходили за пределы соответствующих диапазонов. Несмотря на то, что входные элементы обычно линейны, тем не менее желательно шкалировать данные перед тренировкой. Одно из преимуществ данной операции состоит в том, что при ее выполнении мы будем вынуждены применять некоторые базовые статистические процедуры, касающиеся распределения тренировочных данных и устранения влияния выбросов в тренировочном наборе данных.

Другая причина для выполнения шкалирования данных, или приведения к фиксированному диапазону, состоит в том, что может иметь места ситуация, когда, например, одна переменная изменяется в диапазоне от 10000 до 100000, а другая в диапазоне от 0,3 до 0,6. Очевидно, что ошибки, обусловленные влиянием первой переменной будут сильнее влиять на тренировку, чем ошибки, обусловленные второй, изменяющейся в узких пределах. Обеспечив, чтобы каждая из переменных изменялась в пределах одного диапазона, мы обеспечим равное влияние каждой из переменных на изменение весов в процессе тренировки.

В случае входных элементов представляется целесообразным обеспечить, чтобы значения, передаваемые на входы первого скрытого слоя находились в пределах диапазона входных значений функций активации соответствующего слоя. Для элементов с сигмоидальными функциями диапазон входных значений от-5 до 5. В действительности, нет острой необходимости строго выдерживать указанные диапазоны, так как весовые коэффициенты и смещения сдвигают входные значения до корректных значений, соответствующих входам сигмоидальных функций скрытого слоя. Задача значительно упрощается, если входные значения невелики и характеризуются нулевым средним. По этой причине, а также по причинам, описанным выше, представляется разумным нормировать каждую переменную к нулевому среднему и единичному стандартному отклонению.

2.5.1. Линейное шкалирование

Простейший метод сжатия данных в заданную область изменения состоит в выделении максимальных значений и делении. Если в качестве требуемого диапазона выбран диапазон от нуля до единицы, тогда каждую величину V поделим на диапазон ее изменения, чтобы получить новую

величину S. Эта процедура обладает тем преимуществом, что сохраняет соотношения между величинами. Линейное шкалирование работает хорошо в тех случаях, когда данные распределены равномерно по диапазону изменения. Применение линейного шкалирования к данным, которые распределены неравномерно, или содержат выбросы, приводит к тому, что данные оказываются распределенными в очень узком диапазоне.

Если набор данных с максимальным значением $\max(v_{1\ldots n})$, минимальным значением $\min(v_{1\ldots n})$, средним значением $\bar{v}$ и стандартным отклонением σ_v удовлетворяет следующим условиям:

$$\max(v_{1\ldots n}) \prec (\bar{v} + \lambda\sigma_v) \text{ и } \min(v_{1\ldots n}) \succ (\bar{v} - \lambda\sigma_v),$$

то может быть использовано линейное шкалирование.

Параметр λ определяет количество стандартных отклонений от среднего, при превышении которых величина считается выбросом и исключается из рассмотрения. Если выбросов нет, то линейное шкалирование может быть успешно применено. Линейное шкалирование величины v в переменную s, распределенную в диапазоне от 0 до 1 можно осуществить с помощью формулы:

$$s = \frac{v - \min(v_{1\ldots n})}{\max(v_{1\ldots n}) - \min(v_{1\ldots n})}$$

Обратное преобразование из s в v может быть вычислено в соответствии с формулой:

$$v = \min + s(\max(v_{1\ldots n}) - \min(v_{1\ldots n}))$$

Переменная может быть нормализована к нулевому среднему значению и единичному стандартному отклонению с помощью следующей формулы:

$$s = \frac{v - \bar{v}}{\sigma_v}$$

2.5.2. Мягкое (Softmax) шкалирование

Полезным методом сжатия случайно распределенных данных в пределы подходящего диапазона является использование функции, известной под названием "softmax". Эта функция похожа на стандартную сигмоидальную функцию, используемую в качестве функции активации. В нейронной сети смещения обеспечивают, чтобы данные шкалировались в нужный диапазон сжимающей функции активации. При использовании функции "softmax" степень сжатия каждой из входных величин зависит от того, насколько эти величины далеки от среднего значения (степень сжатия величин, далеких от среднего значения больше, чем степень сжатия величин, близких к среднему значению) и от стандартного отклонения набора данных (чем больше стандартное отклонение, тем больше степень сжатия).

Если $\bar{v}$ - среднее значение величины, подлежащей шкалированию, σ_v - стандартное отклонение, λ - количество стандартных отклонений от среднего, при котором данные должны попасть на линейный участок сжимающей функции, а v - величина, подлежащая шкалированию, "softmax" функция s может быть представлена в следующем виде:

$$x = \frac{(v - \bar{v})}{\lambda \frac{\sigma_v}{2\pi}}$$

$$s = \frac{1}{1 + e^{-x}}$$

Выбор величины λ зависит от важности данных, распределенных вдали от среднего. Для нормального распределения λ =2, при этом 95% данных попадают на линейный участок функции.

При λ =3 99% данных попадают на линейный участок функции.

Обратное преобразование к исходной величине:

$$v = \bar{v} + \frac{\lambda \sigma_v \log \sqrt{-1 + \frac{1}{1-s}}}{\pi}$$

Следует соблюдать осторожность при вычислении этого выражения при значениях s, близких к единице.

Величины, которые далеки от среднего значения, сжимаются сильнее в экспоненциальной зависимости от отклонения от среднего. Функция "softmax" переводит все величины, которые выходят за пределы заданной области, к значениям, близким к нулю или к единице. Если исходные данные распределены равномерно, то все значения будут находиться в пределах трех стандартных отклонений от среднего. Таким образом функция "softmax" может сжать полный диапазон изменения данных в диапазон линейного изменения линейной части сжимающей функции. Поэтому нет необходимости выбирать между линейной и "softmax" функциями, так как функция "softmax" и линейная эквивалентны в случае использования с равномерно распределенными данными.

Следует соблюдать осторожность при использовании созданной сети на новых данных, так как величины, которые будут выходить за края диапазона тренировочных данных будут сжаты в диапазон между нулём и единицей, что приведет к неправильному результату интерполяции.

Так как среднее значение и дисперсия имеют смысл только для нормального или равномерного распределения данных, этот метод не обеспечит правильных результатов для данных, распределение которых далеко от нормального.

Специальные методы шкалирования.

Если известна полная информация о распределении данных, то мы можем сделать сознательный выбор типа шкалирующей функции. Для таких источников данных, как энергетические спектры, характерно логарифмическое распределение. Большая часть значений сосредоточена вблизи среднего, но имеется небольшое количество экспоненциально больших значений. В таких случаях логарифмическая функция шкалирования позволит избежать потери больших значений. Таким способом могут быть шкалированы данные, максимальное значение которых превышает экспоненту от среднего ($\max(x) \succ e^{\bar{x}}$). Логарифмическая функция шкалирования может быть представлена в виде:

$$s = \frac{\log(v) - \log(v_{1\ldots n})}{\log(\max(v_{1\ldots n})) - \log(\min(v_{1\ldots n}))}$$

Для логарифмически шкалированной величины обратное преобразование может быть выполнено следующим образом:

$$x = \log(\min(v_{1\ldots n})) - s(\log(\min(v_{1\ldots n}))) + s(\log(\max(v_{1\ldots n})))$$

$$v = e^{x}$$

Очевидно, что такое шкалирование применимо лишь в том случае, если большие значения (которые будут сжаты в диапазон, который значительно меньше их начальных значений) не несут основной информации об исследуемом процессе. Если же большие значения являются важными, то лучшей моделью будет линейное шкалирование.

Влияние нелинейного шкалирования на точность модели

Недостатком использования нелинейных шкалирующих функций является то, что они вносят в данные новые свойства, которых не было в исходных данных. Мы обсуждали важность равномерно распределенного тренировочного набора для создания корректной, способной к

обобщению модели. Для таких случаев линейное шкалирование является очевидным выбором, так как распределение тренировочного набора данных является равномерным. Вторая особенность использования нелинейного шкалирования состоит в том, что оно увеличивает эффект дисбаланса в тренировочных данных посредством группировки редко расположенных точек данных в равномерные, менее выраженные группы. С другой стороны, при этом может быть построена более точная модель области, в которой данные разрежены.

Шкалирование к более узкому диапазону

Как мы видели ранее, значения выходов сети с сигмоидальными функциями попадают в диапазон значений от нуля до единицы. Так что даже если большие величины входных значений попадают в рабочий диапазон, производная сигмоидальной функции стремится к нулю. Это приводит к следующей проблеме. Если эталонные значения выходов сети равны нулю или единице, то эти значения могут быть достигнуты при очень больших значениях входов. Возможный метод устранения этой проблемы состоит в том, чтобы устанавливать диапазон эталонных значений от 0.1 до 0.9. Однако такое приближение нельзя использовать при решении задач категоризации. Кроме того, возникает проблема, связанная с тем, что у сети появляется возможность выдавать значения, которые выходят за пределы диапазона эталонных значений, (например 0.95).

2.5.3. Обратное шкалирование

Очевидно, что для правильной интерпретации значений, полученных на выходе сети, их необходимо привести к начальному диапазону значений, т.е. произвести обратное шкалирование. Любая шкалирующая функция должна иметь обратную с тем, чтобы можно было вычислить исходные значения. Например, экспоненциальная функция является обратной по отношению к логарифмической функции. Обратное шкалирование нет необходимости применять к входным значениям, если они используются только для получения результатов. Однако, если неоходимо провести анализ полученных результатов на основе входных данных, то обратное шкалирование входных данных необходимо.

3. Построение сети

3.1. Введение

Построение нейронной сети – экспериментальный процесс и даже в какой-то степени похож на алхимию. Несмотря на необходимость экспериментирования, при разработке сети можно сделать некоторые достаточно надежные предсказания поведения сети с теми или иными параметрами. Основная проблема при конструировании сети состоит в выборе оптимального уровня сложности. Сложность сети, как мы увидим далее, может быть оптимизирована различными способами.

3. Построение сети

3.2. Проектирование многослойного перцептрона (МП)

3.2.1. Размер сети

При проектировании нейронной сети необходимо в первую очередь решить вопрос о количестве слоев и количестве элементов (нейронов) в каждом слое. Поскольку количество входных и выходных элементов определяются свойствами входных и выходных данных, мы начнем исследование архитектуры МП с определения размера скрытого слоя.

3.2.2. Размер скрытого слоя

Имеет место компромисс между точностью и обобщающей способностью сети, который можно оптимизировать посредством выбора количества скрытых элементов для данной сети. Количество скрытых элементов с одной стороны должно быть достаточным для того, чтобы решить поставленную задачу, а с другой не должно быть слишком большим, чтобы обеспечить необходимую обобщающую способность. Можно рассматривать весовые коэффициенты как дополнительные члены или параметры функции. Так, если необходимо добавить член более высокого порядка в квадратичную модель для сложного входного сигнала, то мы должны увеличить количество весовых коэффициентов в сети. Так же, как мы можем моделировать кривую с одной точкой поворота включением в модель члена x^2 , (для двух точек поворота необходимо включение члена x^3) единственный скрытый слой способен моделировать одну точку поворота в данных. То же справедливо и для границ в задачах классификации. Чем больше кривых и точек поворота во входных данных, тем большее количество скрытых элементов требуется для построения модели.

Не существует простого способа для определения необходимого числа скрытых элементов сети. Как мы видели выше, выбор зависит от многих факторов. На некоторых из них остановимся ниже.

Ряд противоречивых требований накладывается на количество весовых коэффициентов сети. Во-первых, необходимо иметь достаточное количество данных для тренировки сети с выбранным количеством весовых коэффициентов. Количество весов частично обусловлено количеством входных и выходных элементов и, следовательно, методом кодировки входных и выходных данных. Во- вторых, необходимо иметь достаточное количество весов, чтобы сохранить информацию, заключенную в обучающем наборе данных. Верхний предел необходимого количества весовых коэффициентов зависит от того, насколько полным является информационное наполнение обучающего набора данных. Более разреженное представление будет требовать меньшего количества весов. Этот факт лежит в основе проблемы переобучения (overfitting). Мы не можем просто выбрать теоретический максимум числа весовых коэффициентов, так как в этом случае сеть научится иметь дело только с теми данными, которые предъявлялись в процессе тренировки, и, поэтому обобщающая способность сети будет слабой. В этом смысле выбор размера скрытого слоя зависит от решаемой задачи. Некоторые сети, например сеть для решения задачи сжатия данных должна иметь скрытый слой, меньший по размеру, чем входной слой.

Если для согласования сложности сети с количеством входных параметров необходимо увеличить количество обучающих данных, то необходимо либо собрать больше данных, либо применить один из методов расширения набора данных.

Как мы увидим ниже, рядом исследователей предложены формулы для вычисления ограничений на размеры скрытых слоев. Мы также увидим, насколько сильно различаются эти границы при использовании разных методов. Каждая формула является не более, чем ориентиром и не исключает использования метода проб и ошибок.

Сужающаяся или расширяющаяся сеть?

Первое решение, которое необходимо принять при выборе числа скрытых элементов, состоит в том, чтобы выбрать, будет ли сеть сужающейся (количество элементов в скрытом слое меньше, чем количество входных элементов) или расширяющейся (наоборот). Для задачи извлечения информации из входов с целью обобщения или снижения размерности массива данных, необходимо использовать сужающуюся сеть.

Хороший пример использования сужающейся сети приведен в работе Cotrell et. al. [26] по сжатию изображений. Они взяли набор изображений и подали их на вход МП с единственным скрытым слоем и выходным слоем, который содержал столько же элементов, что и входной слой. Скрытый слой содержал меньшее количество элементов, чем входной слой. Кодированная версия каждого изображения подавалась на вход сети, и точно такое же кодированное изображение использовалось в качестве эталона. Таким образом, сеть использовала меньшее количество скрытых элементов для представления изображения в

более сжатом виде. Значения выходов скрытого слоя можно извлечь и использовать как сжатую версию изображения, из которой позже можно восстановить исходную версию, воспользовавшись набором сохраненных выходных весов сети. Аналогичный подход можно использовать для вычисления главных компонент из набора данных, а также это хороший способ уменьшения размерности данных перед тренировкой МП. Например, в вышеприведенном примере сжатые версии изображений можно использовать для тренировки новой сети, которая должна осуществить классификацию изображений. Эта новая сеть будет содержать меньшее количество входов, и, соответственно, меньшее количество весов, чем сеть для обработки несжатых изображений.

Количество скрытых элементов зависит от количества входных элементов. Существует верхняя граница, которая больше, чем количество входных элементов. Эта верхняя граница может быть вычислена в предположении, что каждый входной вектор должен быть отображен скрытым слоем без потери информации. Если же необходимо выделение каких-либо особенностей, как это обсуждалось выше, то количество скрытых элементов должно быть меньше. Чем выше требования к различению близких входных векторов, тем более расходящейся должна быть сеть и тем ниже ее обобщающая способность.

3.2.3. Аналитическое вычисление размера сети

В следующих разделах будут представлены соображения, которые помогут в выборе размера сети. Не существует простой формулы, и не всегда существует единственно правильный ответ. Размер сети определяется рядом факторов, некоторые из которых можно легко учесть, а некоторые нет. В следующем разделе обсуждаются факторы первого типа, что же касается вторых, то пока мы не будем принимать их во внимание.

Взаимосвязь скрытых элементов с входными элементами

Hecht-Neilson [45] для вычисления верхней границы числа скрытых элементов использовал теорему Колмогорова, которая утверждает, что любая функция n переменных может быть представлена как суперпозиция 2n+1 одномерных функций. Эта граница h равна удвоенному числу входных элементов i плюс единица. Другими словами, нет никакого смысла выбирать количество скрытых элементов большим, чем удвоенное число входных элементов.

$$h \leq 2i + 1 \qquad (3.1)$$

Girossi and Poggio [39] обратили внимание на то, что теорема Колмогорова требует, чтобы функции выбирались для каждого конкретного случая, в то время, как в нейронной сети функции фиксированы и параметризованы. Они также обратили внимание на то, что нейронная сеть требует использования гладких функций для того, чтобы обеспечить необходимую обобщающую способность. Теорема Колмогорова этого не гарантирует. Однако Kurkova [56] считает, что, поскольку нейронная сеть является только аппроксимацией, указанные проблемы не являются существенными. Kurkova переформулировал теорему Колмогорова в терминах последовательности сигмоидных функций.

Существуют методы уменьшения размерности входных данных, если это необходимо. Сделав это, мы можем уменьшить сложность сети.

Учет количества обучающих данных

Как мы уже знаем, нейронные сети это модели, описываемые с помощью набора параметров, называемых весами, или весовыми коэффициентами. Upadhyaya and Eryurek [110] для вычисления верхней границы требуемого количества весовых коэффициентов ω использовали тот факт, что количество параметров, необходимых для кодирования P бинарных последовательностей равно $\log_2 P$. В соответствии с [110]

$$w = i \log_2 p \qquad (3.2)$$

Нейронная сеть такого размера будет кодировать любой вектор обучающей последовательности, так что ω - абсолютный максимум для числа весовых коэффициентов. Для проблем, требующих обобщения, равенство следует заменить неравенством при описании верхней границы. Эта теория применима далеко не всегда, так как бинарное представление используется достаточно редко. Количество векторов, которые подлежат кодированию с помощью нейронной сети, не всегда равно количеству тренировочных точек или количеству категорий. Более правильным будет считать в терминах количества областей, на которые должно быть разбито пространство входов для классификации каждой из тренировочных точек. Чем больше скрытых единиц содержит сеть, тем ближе мы подойдем к построению справочной таблицы пар обучающих данных.

Widrow and Lehr [118] рассмотрели тот факт, что для сети классификации количество тренировочных векторов, на которых производится обучение, отображается в числе выходных единиц сети. Уравнение

$$\frac{w}{o} \leq p \leq \frac{w}{o} \log_2 \frac{w}{o} \qquad (3.3)$$

описывает результирующие границы числа весов в сети.

По отношению к размеру обучающей последовательности и ошибке

Baum and Haussler [13] показали, вероятно, простейшую оценку для использования на практике. Полагая границы ошибок $0 \prec \varepsilon \leq \frac{1}{8}$, количество тренировочных примеров должно быть приблизительно равным количеству весов сети, умноженному на обратную величину ошибки. Например, для предельной ошибки ε =0.1 необходимо использовать обучающую последовательность в 10 раз большую количества весов. Эта зависимость описывается формулой:

$$n \geq \frac{w}{\varepsilon} \qquad (3.4)$$

Причиной того, что величина ошибки играет значительную роль, связана с соотношением между обобщающей способностью и точностью. Малая ошибка переобученной сети не может считаться успехом тренировки. Уравнение 3 показывает нам, что если мы желаем использовать большее количество весов чем то, которое может заполнить набор данных, мы должны остановиться при большей ошибке обучения для того, чтобы сохранить обобщающую способность. Это вынуждает нас жертвовать точностью в пользу обобщающей способности сети.

Так как количество входных и выходных элементов в большинстве случаев определяется задачей, легко определить выражение, описывающее количество весов в терминах количества скрытых единиц в полносвязной однонаправленной сети с одним скрытым слоем:

$$h = \frac{w}{(io)} \qquad (3.5)$$

Влияние уровня шума в обучающем наборе данных

Weigend [116] обратил внимание на то, что чем больше уровень шума в данных, тем больше риск того, что сеть смоделирует именно шум. Если данные не содержат шума, то нет никакого риска переобучения, так как нет объекта для переобучения (overfitting). При повышении уровня шума в данных эффективный размер набора данных уменьшается и шансы переобучения увеличиваются. Указанные закономерности приводят к заключению, что необходимо использовать тем меньшее количество скрытых единиц, чем больше шума в данных. Как мы увидим ниже, добавление шума к данным является одним из методов уменьшения риска выучивания любого шума, присутствующего в обучающем массиве и мы также увидим, что по отношению к сбору данных, если увеличивается уровень шума в данных, необходимо увеличивать и размер массива данных.

Заключение по количеству скрытых элементов

При выборе количества скрытых элементов:

- Никогда не выбирайте h больше, чем удвоенное количество входных элементов.
- Вы можете загрузить p образов по I элементов в каждом в $i\log_2 p$ скрытых элементов. Так что никогда не используйте больше. Если необходимо обеспечить высокую обобщающую способность, выбирайте значительно меньшее количество скрытых элементов.
- Убедитесь, что количество обучающих данных по крайней мере в $1/\varepsilon$ раз больше количества весов в Вашей сети.
- Выявление особенностей требует меньшего количества скрытых единиц, чем входов. Если Вы знаете, что размерность данных может быть уменьшена, используйте меньшее количество скрытых единиц.
- При обучении на бесструктурных входах необходимо, чтобы количество скрытых единиц было больше, чем количество входов. Если набор данных не имеет общих свойств, необходимо использовать больше скрытых единиц.
- Количество скрытых единиц, необходимых для решения задач классификации, увеличивается с увеличением числа классов. Более точно, количества областей, на которые должно быть разделено пространство входов является более важным фактором, чем количество классов само по себе.
- Имеет место взаимоисключающая связь между обобщающими свойствами (меньше единиц) и точностью (больше единиц), которая специфична для каждого приложения.
- Большая сеть требует большего времени для тренировки.

Как мы увидим, причиной ограничения количества скрытых слоев в сети является то, что необходимо сохранить обобщающую способность сети. Сложная модель не обобщает так же хорошо, как простая, но достаточная. Мы можем оценить, исходя из вышеприведенного, ряд ограничений на размер скрытого слоя и некоторые интуитивные соображения, основанные на специфике решаемой проблемы.

3.2.4. Конструктивные алгоритмы

Было реализовано большое количество попыток построения нейронных сетей с добавлением или удалением скрытых элементов. Простейший метод состоит в том, чтобы начать с количества элементов на один меньшего нижней границы и тренировать сеть до стабилизации ошибки. После этого добавить еще один скрытый элемент с малым весом и повторять процедуру до тех пор, пока ошибка на независимом тесте не начнет увеличиваться. Но на этом пути существует ряд проблем. Главная из которых состоит в том, что новые веса исказят сеть настолько, что было бы лучше начать снова, чем оставаться в выбранной области пространства весов.

Иерархическое конструирование

Существует много методов иерархического конструирования, таких как upstart algorithm [37] и метод каскадных корреляций [35]. Идея, лежащая в основе иерархических сетей заключается в том, что добавляется подчиненный узел, чтобы скорректировать ошибку основного. Таким путем может быть построено дерево бинарных заключений для двух элементов, добавляемых для корректировки двух типов возможных бинарных ошибок. Такие сети не являются многослойными перцептронами.

Добавление новых весов и замораживание старых

Refenes [79] предложил алгоритм, называемый CLS+, согласно которому сначала строим сеть с малым количеством скрытых элементов, а затем добавляем столько скрытых элементов, сколько потребуется. Для быстрой сходимости процесса обучения необходимо, чтобы старые веса замораживались, а добавлялись новые от предыдущего скрытого элемента к

добавляемому. Следует соблюдать осторожность, так как новые веса тренируются только на оставшейся части обучающей последовательности. Так что каждый скрытый элемент отвечает за подмножество обучающего массива. Refenes не ссылается на независимые тесты, когда делает заключение о том, есть ли необходимость добавлять новый элемент. Это, а также нестандартная процедура тренировки и архитектура мешают широкому использованию этого алгоритма.

3.2.5. Динамическое добавление и удаление

Bartlett [10] использует информационно-теоретическое приближение для динамического распределения элементов. Его алгоритм стартует с минимального количества элементов и тренирует до стабилизации ошибки. В этой точке в скрытый слой добавляется новый элемент с малым случайным весом и тренировка продолжается. Процесс повторяется до тех пор, пока добавление нового элемента не приведет к увеличению стабилизировавшейся ошибки. В некоторый момент в течение этой процедуры элемент может быть удален, если он не несет достаточной информации о выходных эталонах сети.
Это информационное содержание вычисляется с использованием меры энтропии Шеннона между скрытым элементом и требуемыми величинами на всем выходном слое минус избыточность этого элемента по отношению к другим скрытым элементам.

Информация, заключенная в некотором узле, x, зависит от вероятностей $P(g)$ появления каждой из n возможных выходных картин g и определяется формулой

$$H(x) = -\sum_{g=0}^{n} P(g)(\log_2(P(g))) \qquad (3.6)$$

Совместная информация между двумя элементами x и y определяется как

$$H(x,y) = -\sum_{g=0}^{n}\sum_{h=0}^{m} P(g,h)(\log_2(P(g,h))) \qquad (3.7)$$

где m – количество возможных картин h для элемента y и $P(g,h)$ - вероятность что элемент x породит картину g а элемент y породит картину h.

Затем необходимо вычислить зависимость между двумя элементами, $U(x,y)$:

$$U(x,y) = 2\left(\frac{H(x) + H(y) - H(x,y)}{H(x) + H(y)}\right) \qquad (3.8)$$

Если мы обозначим взаимосвязь между скрытым элементом x и выходным элементом y как $HO(x,y)$, а взаимосвязь между двумя скрытыми слоями как $HH(x,y)$ тогда важность узла $I(x)$ может быть вычислена как:

$$I(x) = \sum_{y} HO(x,y) - \sum_{y,y=x} HH(x,y) \qquad (3.9)$$

Т.е. сумма взаимосвязей между скрытым элементом и выходным слоем минус сумму взаимосвязей между скрытой единицей и другой скрытой единицей. Скрытая единица с наименьшей важностью будет исключен из сети.

Если выходные данные сети представляются не в дискретном виде, а в виде непрерывном, то диапазон этих непрерывных значений необходимо разбить на отрезки, например 0-0,1; 0,1-0,2 и т.д., а вероятности попадания величин в эти отрезки вычисляются с использованием уравнения (3.6). Вычисление энтропии, основанной на обучении, является процедурой, требующей много вычислительных ресурсов.

Bartlett не тестировал сеть на независимом тестовом наборе, так как ему не требовалось создавать сеть с обобщающей способностью. Это может быть исправлено посредством вычисления значимости элементов на контрольном наборе данных, а не только на тестовом.

В качестве более простого, но менее эффективного метода, можно использовать т.н. метод отсечения весов, заключающийся в удалении весов с очень малыми значениями. Такой метод входит в противоречие с целью создания сети с малыми весами для того, чтобы построить обобщающую модель. Более эффективный метод отсечения – метод оптимального повреждения мозга (Optimal Brain Damage). Этот метод основан на оценке влияния вносимых изменений на ошибку. Веса удаляются только в том случае, если их удаление приводит к незначительному увеличению ошибки.

Обсуждение "конс_труктивных" сетей

Практическое руководство [31] отмечает, что "здоровая" нейронная сеть должна обладать набором весовых коэффициентов, распределенных по нормальному закону в окрестности нуля. Другими словами, сеть, в которой большая часть весовых коэффициентов имеет большие значения, не является "здоровой" сетью. То есть, существует опасность, что алгоритм конструирования сети, особенно, если он включает процедуру удаления малых весовых коэффициентов, приведет к созданию "нездоровой сети". Один из возможных путей обойти эту проблему состоит в ограничении весовых коэффициентов и в добавлении новых как только последние достигнут некоторого порогового значения. Это может привести к насыщению

весовых коэффициентов, но это не представляет большой опасности.

Заключение по алгоритмам конструирования сети

Разработка "конструктивных" нейронных сетей в настоящее время находится в такой стадии, когда предлагается множество вариантов архитектуры сетей, но при этом каждый их этих вариантов тестируется на небольшом количестве задач или, что еще хуже, разрабатываются варианты для решения только одной проблемы. И очень сложно сделать обобщающие выводы.

Итак, при построении "конструктивной" сети мы должны выбрать:

- Будем ли мы строить иерархическую сеть или стандартный многослойный перцептрон?
- Фиксированы старые веса или нет перед добавлением новых?
- Будут ли поперечные соединения между скрытыми элементами? Если да, то это уже не будет стандартным многослойным перцептроном.
- Как мы будем принимать решение о необходимости удаления скрытого элемента, и какой элемент будем удалять?
- Каким образом мы будем устанавливать весовые коэффициенты между новыми скрытыми элементами? Будут ли они случайными или будут установлены с учетом ошибок?
- Как мы узнаем, когда нужно остановиться, и как мы будем возвращаться обратно, если окажется, что добавлено слишком много элементов?
- На чем будет основываться уверенность в том, что имеющееся небольшое количество весовых коэффициентов в самом деле соответствует оптимальной конфигурации?
- Можем ли мы ограничить увеличение некоторого количества весов?

Далее мы увидим, что наилучший путь достижения оптимального набора весовых коэффициентов состоит в том, что бы построить сеть с размером скрытого слоя, который определяется правилами и знанием некоторой априорной информации о данных, тренировать ее и тестировать ее. Если результат не оправдывает ожиданий, обратите внимание на ошибки тренировки и тестирования для того, чтобы пересмотреть размер скрытого слоя, а затем начните снова с новыми малыми случайными весами. Только таким путем Вы сможете получить хорошо сбалансированное решение и сможете избежать ошибок в выборе правильного алгоритма для решения Вашей проблемы.

Ниже перечислены простейшие признаки корректной нейронно-сетевой модели:

- Если ошибка тренировки мала, а ошибка тестирования велика, значит сеть содержит слишком много весовых коэффициентов.
- Если и ошибка тренировки, и ошибка тестирования велики, значит весовых коэффициентов слишком мало.
- Если все весовые коэффициенты очень большие, значит весовых коэффициентов слишком мало.
- Добавление весов не панацея; если Вы считаете, что весовых коэффициентов достаточно, подумайте о других причинах ошибок, например недостаточное количество обучающих данных.
- Не добавляйте слишком много весовых коэффициентов, что бы не переступить пределов, установленных в 3.2.2.
- И наконец, что очень важно, начальные весовые коэффициенты должны быть случайными и небольшими по величине (скажем между +1 и −1).

3.2.6. Обобщающая способность и точность: другие методы

Ripley [83] полагал, что количество скрытых элементов менее важно, чем обычно полагают, и что Вы можете достичь хороших результатов путем использования других методов. Это полностью согласуется с позицией, представленной в настоящей книге, которая устанавливает, что сложность модели определяется на каждой стадии проекта и, что было бы ошибкой полагать, что выбор корректного количества скрытых единиц гарантирует корректность проекта. Простейший метод убедиться в приемлемом уровне обобщающей способности состоит в том, что бы периодически тестировать сеть на независимом контрольном множестве данных и остановить тренировку, как только ошибка этого тестирования начнет возрастать. Это наиболее мощный метод, однако, при этом увеличивается как время тренировки, так и необходимое количество данных. Далее будет описан ряд более тонких методов.

3.2.7. Добавление регуляризующего члена

Еще одно техническое решение состоит в использовании регуляризующего члена в функции ошибок определенной таким образом, чтобы ограничивать сложность решения, как правило, путем ограничения величин весов. Сеть с меньшими значениями весовых коэффициентов будет обладать лучшими обобщающими свойствами. Простейший способ добавления регуляризующего члена состоит в том, чтобы изменить меру ошибки так, чтобы использование больших весовых коэффициентов рассматривалось как ошибка определенного сорта. Чем сильнее возрастает весовой коэффициент, тем больше становится ошибка, и это приводит к выбору простейшего из возможных решений для данного набора данных и, таким образом, улучшается обобщающая способность.

При внесении регуляризующего члена в функцию ошибок нейронной сети следует принять во внимание тренировочная последовательность какого размера была использована при обучении. Это связано с тем, что эффект регуляризации уменьшается пропорционально размеру тренировочной последовательности данных. Также необходимо иметь возможность контролировать степень регуляризации и увеличивать величину ошибки пропорционально количеству весов в сети.

Это рассмотрение приводит нас к новой функции ошибок, представленной уравнением:

$$reg\_error = \frac{\lambda}{p}\sum_i w_i^2$$

где λ - параметр регуляризации, p - размер тренировочной последовательности,

w_i - весовой коэффициент, i - индекс. Величина полной ошибки может быть вычислена следующим образом:

$$error_i = error_i + reg\_error_i$$

где $error_i$ - ошибка, вычисленная одним из стандартных способов.

Это уравнение вводит еще один параметр, который можно настраивать: параметр регуляризации λ . Чем больше величина λ , тем выше степень регуляризации. λ принимает значения от 0 до 1. Короче говоря, этот метод увеличивает величину ошибки пропорционально сумме квадратов весов сети. При возрастании весов возрастает ошибка.

Добавление шума эквивалентно регуляризации

В случае, если Ваше программное обеспечение не поддерживает специальных методов регуляризации, можно использовать добавление шума к тренировочным данным. В случае, если для каждого тренировочного прохода используются новые значения шумовой компоненты, у сети не будет шансов переобучиться. Чем больше добавлено шума, тем выше будет обобщающая способность сети. Bishop [16] показал, каким образом добавление шума к тренировочной последовательности эквивалентно введению регуляризации Тихонова. Регуляризационный член, использованный Bishop, основан на том факте, что добавление шума к входу приведет к изменению выхода, которое определяется структурой самой сети. Эффект влияния небольшого изменения входа (обусловленный добавлением шума) на выходы определяется производной выходных сигналов по отношения ко входным.

Преимуществом данного метода является возможность варьирования параметра λ. Тот же эффект может быть получен посредством использования функции активации во входном слое, которые возмущают входные значения на величину, пропорциональную λ перед передачей сигнала к скрытому слою. Входные элементы, как правило, не содержат функции активации, они просто передают входные величины, с учетом весовых коэффициентов, к скрытому слою. В известном смысле, любая шкалирующая функция, которая применяется к данным до их предъявления сети, эквивалентна функции активации входного элемента. Единственная разница состоит в том, что данные могут быть изменены лишь однажды, порождая новый набор данных, и при этом уменьшается количество вычислений в ходе обучения.

Если данные уже были нормированы до диапазона между 0 и 1, в этом случае λ представляет из себя отношение сигнал/шум. Хотя λ является дополнительным параметром вызывающим некоторую неопределенность при выборе, его достаточно легко контролировать. Начав с больших значений (например 0.1) уменьшаем каждый раз, когда ошибка перестает уменьшаться, до тех пор, пока ошибка на контрольном массиве не начнет возрастать. Таким образом можно получить решение.

Следует сделать два важных замечания:

1. Разные значения шума должны добавляться к каждой величине каждый раз, когда она используется сетью. Недостаточно просто создать новый набор тренировочных данных, к которому добавлен шум. Шум должен добавляться динамически в процессе обучения.
2. При добавлении шума к тренировочным данным необходимо сохранять среднее значение данных. Другими словами шум должен обладать нулевым средним. Не забудьте убрать шум при тестировании!

Сжимающая функция может быть использована в качестве функции активации входного слоя. Шум должен быть добавлен после того, как данные были нормированы, чтобы сохранить амплитуду постоянной для всех переменных.

Процедура ввода и последовательного уменьшения шума в процессе обучения сети может быть использована как средство против захвата сети локальными минимумами. Шум эффективно сглаживает мелкие пики в ошибке, которые не может преодолеть алгоритм обучения.

3.2.8. Количество скрытых слоев

Сеть, которая состоит всего из входного и выходного слоев без скрытых слоев известна как линейный перцептрон. Перцептроны способны моделировать только линейные функции и используются очень редко. И это не всегда оправдано, так как они обеспечивают универсальную линейную аппроксимацию, что часто и требуется для конкретной задачи. До тех пор, пока вы не убедитесь в том, что Ваша проблема существенно нелинейна, имеет смысл попытаться решить ее с помощью линейного перцептрона. Функция единственного скрытого слоя заключается в том, чтобы перекодировать входное представление таким образом, чтобы получить линейную карту для выходного представления. Многослойный перцептрон с единственным скрытым слоем является перцептроном на вершине линеаризирующей функции.

Kurkova [56] использовал теорему Колмогорова, чтобы показать, что любая функция может быть аппроксимирована с использованием, по крайней мере, четырех слоев. Hecht-Neilson [45] показал, что достаточно трех, но констатировал, что в реальной сети использование большего количества слоев приводит к уменьшению суммарного количества элементов в скрытых слоях. Анализ опубликованных результатов, однако, указывает на то, что для решения большинства практических задач достаточно одного, иногда двух скрытых слоев. Причина такого несоответствия теории и практики лежит, по-видимому в том, что сложность реальных проблем намного меньше, чем это теоретически возможно.

3.2.9. Функции активации

Функция активации элемента (нейрона) суммирует взвешенные входы от всех присоединенных элементов и сжимает их в заданный диапазон значений. Этот диапазон выбирается равным либо (0 ; – 1_, либо (–1; +1). При этом большие значения всегда сжимаются таким образом, что они вносят уменьшающийся вклад. По этой причине, а также потому, что сеть должна моделировать нелинейные процессы, функции активации должны быть нелинейными.

Наиболее часто используется логистическая функция, которая должна обладать следующими свойствами. Во-первых, сжимать входные значения в диапазон от 0 до 1 и, во-вторых, производная должна слабо изменяться на каждом из краев диапазона и сильно изменяться в середине диапазона. В качестве таких функций используются либо логистическая функция, либо функция гиперболический тангенс.

Логистическая функция

$$f(x) = \frac{1}{1+e^{-x}}$$

не симметрична, что приводит к некоторому увеличению времени тренировки. Производная этой функции:

$$f'(x) = \frac{1}{(e^x(1+e^x)^2)}$$

Функция гиперболический тангенс

$$f(x) = \tanh(\gamma x) \equiv \frac{e^{\gamma x} - e^{-\gamma x}}{e^{\gamma x} + e^{-\gamma x}}$$

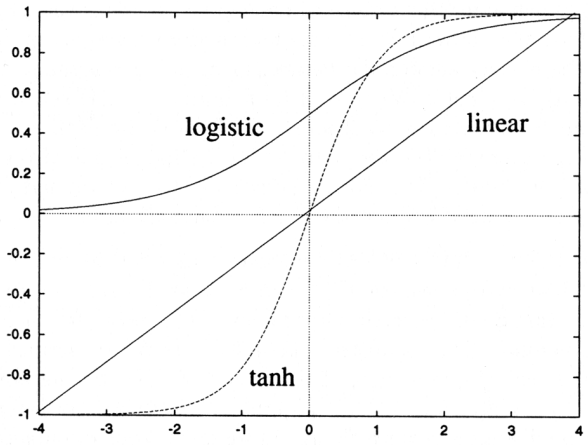
нечетная функция, что несколько увеличивает скорость тренировки.

Обе функции называются сигмоидальными функциями благодаря их S-образной форме.

Morris [67] считает, что один из скрытых элементов всегда должен быть линейным. Это связано с тем, что большинство проблем имеют линейные компоненты и очень важно промоделировать прямую линию сложением нескольких взвешенных нелинейных функций. Линейная функция активации просто суммирует входящие взвешенные величины и делит результат на некоторую константу.

Отметим, что можно изменять наклон функций, и таким образом изменять скорость, с которой они приближаются к их максимальным значениям, введением постоянного члена. Этот член влияет на резкость границ в сети классификации. Если функция с крутым наклоном после обучения обеспечивает разделение двух областей с вероятностями 0.9 и 0.1, то та же сеть, тренированная на той же задаче, но с меньшим наклоном функции активации, обеспечит разделение двух областей с вероятностями 0.7 и 0.3 при том же входе.

Линейная, логистическая и функция гиперболический тангенс показаны на рисунке.



Вертикальная ось представляет выход, а горизонтальная вход. Отметим, что функция tanh имеет ту же форму , что и логистическая функция, но изменяется в пределах от −1 до +1. Как правило, современные компьютеры вычисляют функцию tanh быстрее, чем логистическую. Другое преимущество функции tanh состоит в том, что она изменяется в диапазоне от −1 до +1. Часто бывает необходимо нормировать обучающий набор данных таким образом, чтобы среднее значение было равно 0 при единичном стандартном отклонении. Такая нормировка возможна только с функцией активации, которая способна принимать отрицательные значения. И наконец, как мы увидим далее, нечетная функция, такая как tanh, обеспечивает более быстрое обучение, чем несимметричная логистическая функция.

Аппаратное рассмотрение

Определенным преимуществом обладают сети, которые допускают аппаратную реализацию. При этом некоторые функции легко реализуются аппаратно, а некоторые с трудом. В частности, логистическая функция больше подходит для аппаратной реализации. Функция tanh может быть как вычислена, так и представлена в виде таблицы значений. Последний ввариант более быстрый, но требует дополнительных расходов на память. Некоторые коммерчески доступные нейронно-сетевые чипы поддерживают только некоторые типы функций активации.

3.2.10. Меры ошибок

Цель тренировки нейронной сети состоит в том, чтобы минимизировать ошибку, которая возникает на каждом выходном элементе

на тренировочном множестве данных. Так как ошибка может быть положительной или отрицательной, нас будет интересовать величина ошибки при измерении средней точности сети как целого. Знак ошибки важен лишь тогда, когда вычисляется индивидуальная ошибка между конкретным выходом сети и эталонным значением. Веса должны модифицироваться таким образом, чтобы величина ошибки была близка к нулю.

Существует три типа ошибок, которые нас интересуют. Первый тип – ошибка единичного выходного элемента, которая необходима для реализации процедуры обратного распространения ошибки. Второй тип – ошибка всей сети при конкретном входном сигнале, которая дает нам информацию о том, насколько правильным является ответ сети в данный момент времени. Третий тип – средняя ошибка сети, вычисленная после предъявления всего набора тренировочных данных, которая показывает насколько хорошо сеть усвоила закономерности набора тренировочных данных. Последняя ошибка представляет собой усредненное по всему набору тренировочных данных значение ошибки второго типа. Так как второй тип ошибки часто вычисляется как расстояние (модуль), то значение этой ошибки всегда положительно и ее усреднение не требует вычисления абсолютных значений.

В дополнение к двум уровням меры ошибок, описанных выше, существуют несколько индивидуальных мер ошибок, которые могут быть использованы. Каждая из них определяется конкретными требованиями к сети. Чаще всего используется мера ошибки в виде простой разности между значением эталона и значением соответствующего выхода сети для каждого входного значения из тренировочной последовательности:

$$error_i = t_i - a_i,$$

где t_i - эталонное значение для i-го выхода, a_i - текущее значение того же выхода.

Для того, чтобы вычислить ошибку всего выходного слоя, мы можем трактовать эталоны и выходы как точки в многомерном пространстве и вычислять расстояние между ними. Расстояние между двумя точками вычисляется как квадратный корень из суммы квадратов разностей координат. Ошибка выхода сети с N выходными элементами равна квадратному корню из суммы квадратов ошибок каждого выходного элемента:

$$\overline{error} = \sqrt{\sum_{i=1}^{n}(t_i - a_i)^2}.$$

Такая ошибка известна как квадратичная ошибка (RSE – root squared error). Одна из разновидностей RSE – weighted RSE – взвешенная ошибка (под термином "вес" в данном случае понимается не значения весов сети, а мера важности, присвоенная каждому из тренировочных примеров):

$$error_i = W_p(t_i - a_i).$$

С каждым примером тренировочного набора данных связывается вес W, который указывает на значимость данного примера, или на его "качество". Значения весов должны лежать между 0 и 1. RSE определяется в этом случае по формуле:

$$error_i = W_p\sqrt{\sum_{i=1}^{n}(t_i - a_i)^2}.$$

Для задач категоризации можно установить порог, так, чтобы выходы выходных элементов принимали значения либо 0 (не принадлежит данному классу), либо 1(принадлежит данному классу). Это должно быть сделано после того, как вычислена ошибка, чтобы значения меры ошибки не принимали значений или 0 или 1. Для вычисления меры ошибки недостаточно знать "да" или "нет", необходимо вычислить размер отклонения.

В некоторых случаях представляется необходимым определить ошибку в терминах конкретной задачи, для решения которой сконструирована сеть.

3.2.11. Установки значений скорости обучения и момента

Скорость обучения, η и момент, α определяют характер модификации весовых коэффициентов в процессе обучения. Оба

параметра используются при применении правил обучения. η определяет долю вычисленной ошибки, которая дает вклад в изменение весов. Момент α , связан с величиной предыдущей корректировки весового коэффициента. Значение каждой корректировки весовых коэффициентов запоминается для использования в последующих циклах. Правило модификации весов с учетом моментов включает член, пропорциональный предыдущему изменению веса.

Высокие значения η обеспечивают быстрое обучение, но увеличивают риск отклонения от решения с последующими осцилляциями вокруг него. Низкие значения η устраняют эту проблему, но приводят к замедлению процесса обучения. Высокие значения α уменьшают риск захвата локальными минимумами, но увеличивают риск проскакивания решения с последующими осцилляциями аналогично ситуации с высоким значением η .

Начиная с большого значения η (скажем 0.75) уменьшают его до (скажем) 0.25 и затем до 0.1 если сеть начинает осциллировать. Hertz [46] предлагает устанавливать для α величину 0.9 чтобы подавить осцилляции при высоких значениях η . При этом эффективное значение скорости обучения будет $\eta/(1-\alpha)$. Большинство примеров в настоящей книге

используют значения $\alpha = 0.9$ и $\eta = 0.25$. Только в тех случаях, когда очевидно, что требуются другие значения, или

если возникали проблемы с обучением сети мы пытались использовать другие значения.

Противоречия, возникающие при попытках одновременного устранения проскакивания решения и захвата локальными минимумами очевидны, но следующие простые эвристические привила будут полезны:

1. Если ошибка уменьшается медленно, но равномерно, можно увеличить η и α.
2. Если ошибка осциллирует в окрестности некоторой точки, уменьшение η поможет сети достичь этой точки. Увеличение η или α может заставить сеть преодолеть осцилляции и продолжить обучение, но не всегда.
3. Если ошибка не изменяется, значит, по всей видимости Вы достигли решения. Всегда сохраняйте копию весовых коэффициентов в этой точке прежде, чем Вы попытаетесь изменить параметры в попытке заставить сеть преодолеть плато и продолжить обучение.

3. Построение сети

3.3. Тренировка нейронных сетей

После того, как нейронная сеть создана и набор данных для тренировки подготовлен, сеть можно тренировать. На практике, как мы увидим далее, необходимо построить и тренировать несколько вариантов нейронных сетей на подготовленном наборе тренировочных данных, прежде чем будет получено удовлетворительное решение.

3.3.1. С чего начать

Несмотря на то, что в предыдущих разделах приведены правила, которыми следует руководствоваться при принятии различных решений относительно параметров сети, существует этап проектирования сети, который требует использования метода проб и ошибок. Мы достигли этого этапа. При построение нейронной сети для любой, даже весьма тривиальной задачи, потребуется построить несколько нейронных сетей различной сложности, испытать несколько вариантов выбора точки остановки при тренировке сети, стартовать с различных начальных значений весовых коэффициентов. Каждую из этих сетей следует сохранить, протестировать, проанализировать с тем, чтобы наконец, выбрать наиболее подходящую. Мы уже рассмотрели как можно преодолевать локальные минимумы путем выбора значения момента, как сохранять копии весов перед тем, как сделать это. Каждый такой проход порождает новую сеть, и могут возникнуть сложности, если файлов результирующих весов нет под рукой.

Единственное окончательное решение может быть выбрано из множества сетей, созданных в процессе тренировки, или же выбор может быть сделан на основе экспертной оценки.

3.3.2. Режимы тренировки: последовательный или пакетный

Процедура предъявления сети всего набора тренировочных данных называется эпохой. Так как задачей нейронной сети является уменьшение средней ошибки на всем наборе тренировочных данных, такая процедура является удачным вариантом получения суммарной ошибки по всем выходным элементам и всем тренировочным данным. После завершения одной эпохи вычисляется единственная усредненная ошибка и сеть модифицируется в соответствии с этой ошибкой. Этот режим тренировки известен как пакетный режим обучения. Альтернативный к пакетному режиму, т.н. последовательный режим (pattern mode), заключается в том, что модификация весов сети производится после предъявления каждого из примеров тренировочной последовательности.

Выбор между двумя вариантами не прост, и разница в эффективности того или иного режима часто зависит от специфики решаемой задачи. При выборе режима тренировки следует руководствоваться следующими соображениями:

- Пакетный режим требует меньшего количества модификаций весов и, поэтому, является существенно более быстрым.
- При использовании пакетного режима обеспечивается более точное вычисление изменений в весовых коэффициентах.
- Пакетный режим относится только к подстройке весов; необходимо обеспечить обратное распространение ошибки. Каждый элемент должен поэтому накапливать свое текущее среднее; при этом к сети добавляется требование обладать возможностями накопления.
- Пакетный режим более подвержен захвату в локальные минимумы по сравнению с последовательным режимом.
- При проектирование сети часто необходимо тренировать несколько вариантов сетей на одних и тех же данных, прежде чем будет найдено удовлетворительное решение. Тренировка в различных режимах хороший метод разнообразить условия тренировки при выборе оптимального решения.

3.3.3. Ускорение процесса тренировки

Мы уже рассмотрели, каким образом можно выбрать параметр η перед тренировкой сети. Также можно динамически подстраивать этот параметр в процессе тренировки для того, чтобы в какой-то степени ускорить процесс. Часто желательно, хотя и не необходимо, с каждым весом сети связать отдельный параметр. Это увеличивает время вычислений, создает сложности для программирования и стабильности сети и, поэтому, такой способ может быть использован только в тех случаях, когда другие методы не приводят к успеху. Следующие наблюдения могут помочь при ускорении тренировки сети.

- Если данный вес последовательно изменяется в одном и том же направлении для нескольких последовательных предъявлений, можно увеличить скорость обучения для данного веса. Естественно, это требует использования независимых параметров обучения для каждого веса.

- Наоборот, если веса изменяются в различных направлениях на последовательно предъявляемых примерах, их параметры обучения должны быть уменьшены.
- Если поверхность ошибок очень плоская, тогда скорость обучения может быть увеличена. Это простейшее правило может быть применено когда используется только единственный параметр обучения.
- Когда ошибка большая, мы можем позволить большие изменения весов.
- Как только ошибка уменьшится, мы должны выполнять более тонкую подстройку.

Отметим, что использование момента автоматически обеспечивает эффект, достигаемый в первых двух разделах, без изменения параметров обучения.

Наблюдения, сделанные выше, могут применяться понемногу. Другими словами, параметры обучения могут изменяться от предъявления к предъявлению, но любое единичное предъявление будет оказывать только слабое влияние на скорость обучения. Такой контроль вводится использованием еще одного параметра, который устанавливает размер шага изменения параметра обучения. Представляется разумным задать ограничения на параметры обучения. Они не могут достигать нулевых значений, так как в этом случае процесс обучения остановится.

Многие программные пакеты моделирования нейронных сетей предусматривают возможность динамической подстройки параметра обучения. Следует запомнить, что те же самые механизмы, которые ускоряют движение сети к решению, также могут приводить к попаданию и застреванию сети в локальном минимуме. Как мы уже отмечали, один из возможных тактических подходов в решении этой проблемы состоит в том, чтобы запомнить веса, увеличить параметры обучения до максимума и медленно уменьшать их в надежде, что сеть "выпрыгнет" из локального минимума и продолжит обучение. Если же такая тактика полностью разрушит решение, тогда оно может быть восстановлено на основе сохраненных весов.

Другие заметки по ускорению

Было показано, что сети с нечетными функциями активации обучаются быстрее, чем сети с несимметричными функциями активации. Функция является нечетной, если $f(-x) = -f(x)$. Предварительное кодирование и шкалирование тренировочных данных приводит к увеличению скорости тренировки по сравнению с тренировкой на исходных данных, так как в этом случае используется сеть меньшего размера. Время может быть уменьшено, если мы начнем с простейшей модели, постепенно усложняя ее до достижения оптимального соотношения точности и обобщающей способности.

И наконец, стоит рассмотреть, как Вы загружаете процессор. Каждый процессор работающий в среде Windows допускает возможность тренировки сети в фоновом режиме, когда на компьютере запущены другие приложения. Это увеличивает время тренировки и замедляет работу других программ. Нейронная сеть в процессе тренировки очень сильно загружает процессор, практически не оставляя возможностей для параллельной работы других программ. Поэтому очень желательно, чтобы для тренировки нейронной сети использовать отдельную машину.

Оптимальной тактикой является использование ночного времени и выходных дней для тренировки нейронных сетей.

3.3.4. Когда останавливать тренировку

Есть много причин для того, чтобы иметь простое правило как узнать, когда необходимо остановить тренировку сети. Во-первых, алгоритм обратного распространения не гарантирует сходимости. Другими словами, нельзя ожидать достижения такой конфигурации весов, которая обеспечит абсолютный минимум ошибки. Во вторых, наименьшая возможная ошибка для зашумленных данных не нуль, и мы не можем заранее знать ее величину. И последнее, процедура обратного распространения разработана для уменьшения ошибки тренировки, в то время как наша цель состоит в уменьшении ошибки обобщения.

С учетов вышеприведенных замечаний, мы можем констатировать, что нейронная сеть обучена (т.е. достигнута такая точка, далее которой нельзя достигнуть никаких улучшений), если удовлетворяется один или более из следующих критериев:

- Усредненная ошибка тренировки достигла заранее заданной величины.
- Средняя ошибка тренировки не уменьшается, или уменьшается незначительно.
- Средняя ошибка независимого теста начинает возрастать, указывая на начало переобучения.

3.3.5. Меры "здоровья" сети

Наряду с наблюдением за поведением ошибки сети, можно сосредоточить внимание на "здоровье" сети с точки зрения распределения значений весовых коэффициентов.

Гистограммы весовых коэффициентов

Мы видели, что для достижения обобщающей способности сети, необходимо, чтобы значения весовых коэффициентов были малыми. Этот факт позволяет нам сформулировать простой критерий "здоровья" сети, который известен как гистограмма весов. Гистограмма весов показывает количество весов сети, чьи значения попадают в заданный интервал.

Построение гистограммы весов

Гистограмма может быть представлена в стандартном виде как столбцовая диаграмма. Для построения гистограммы необходимо вычислить диапазон изменения весов сети, разбить его на поддиапазоны (достаточно на 10) и вычислить количество весов, которые попадают в каждый из поддиапазонов.

Анализ гистограммы весов

Гистограмма "здоровой" сети имеет плавный максимум в области малых амплитуд (в окрестности нуля), указывая, что большинство весовых коэффициентов невелики. Ясно, что маловероятно, чтобы некоторые веса обращались в нуль. Гистограмма

с пиками на краях диапазона изменения указывает на вероятность переобучения. Величины весов зависят от вида функции активации. Взглянув на рис. можно заметить, что выход сигмоидальной функции близок к нулю или к единице при диапазоне входных значений более, чем 5.

Изменяемость гистограмм весов

Внимательный читатель (или читатель, знакомый с гистограммами) отметит, что нетренированная сеть с малыми случайными весами демонстрирует "здоровую", хотя и слишком узкую гистограмму. Поэтому представляет интерес оценка дисперсии гистограммы в процессе обучения сети. Веса, которые не двигаются от их начальных состояний, не могут давать большого вклада в решение (если, конечно, они не оказались правильными при инициализации). Если ни один из весов сети не изменился по отношению к начальному состоянию, значит сеть не обучилась. Начальные веса должны быть случайными величинами в диапазоне от −1 до +1. "Здоровая сеть должна иметь веса с максимальной амплитудой около 10. На рис. представлены образцы хороших и плохих гистограмм.