Нейронные сети. Лекции.

Лабунец Л.В. & Паймин Антон

Содержание

Лекция

1.1 Математическая модель многослойного перцептрона

Рассмотрим на примере 3-слойной архитектуры (в количество слоёв входят только слои, реализующие нелинейное отображение). Рассмотрим сигналы, поступающие на вход нейронов, но для избежания путаницы, будем визуализировать связи только для первых нейронов соответствующих слоёв. Рассмотрим эти связи и соотв. им параметры.

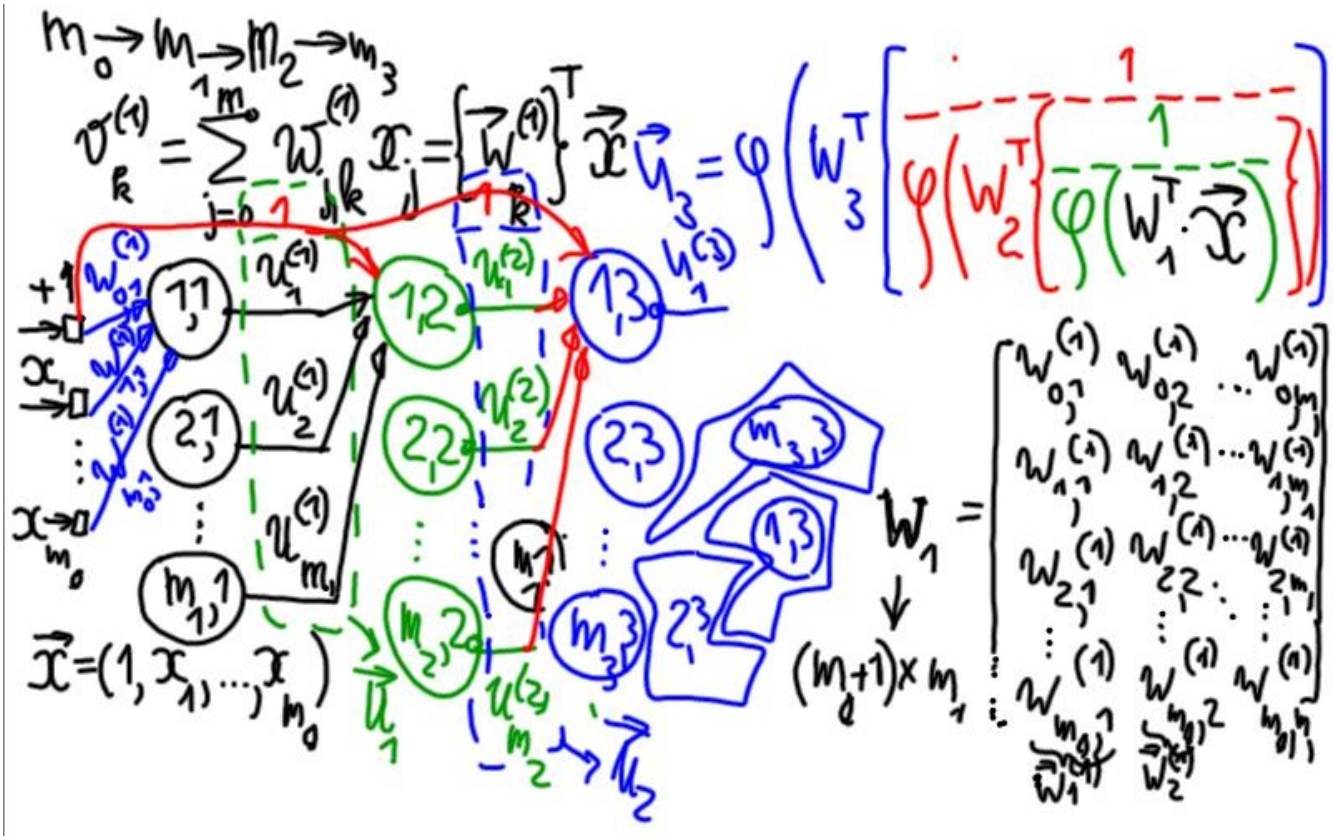

Рис. 1: Структура многослойного перцептрона

3-слойный перцептрон содержит 0-й слой - слой источников (черные квадратики). Имеется m_0 входов, на которые поступают признаки $\vec{x} = (1, x_0, ..., x_{m_0})$, 3 скрытых слоя.

Примем следующие правила индексации скрытых нейронов и нейронов выходного слоя: первый эл-т индекса - номер нейрона, второй - номер слоя: $m_{i,j}$ - i-й нейрон j-го скрытого слоя.

Выпишем веса нейронов i-го слоя в виде матрицы W_i, столбец которой $\vec{w}_i^{(k)}$ является вектором весов i-го нейрона k-го слоя. Количество строк матрицы - количество входов слоя „+1“(на вход поступает расширенный вектор), количество столбцов - количество нейронов слоя. Элемент матрицы $w_{i,j}^{(k)}$ - вес i-го входа j-го нейрона k-го слоя. У первого скрытого слоя имеется m_0 входов (слой источников) и он содержит m_1 нейронов.

$$W_1 = \begin{pmatrix} w_{0,1}^{(1)} & w_{0,2}^{(1)} & \dots & w_{0,m_1}^{(1)} \\ w_{1,1}^{(1)} & w_{1,2}^{(1)} & \dots & w_{1,m_1}^{(1)} \\ w_{2,1}^{(1)} & w_{2,2}^{(1)} & \dots & w_{2,m_1}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m_0,1}^{(1)} & w_{m_0,2}^{(1)} & \dots & w_{m_0,m_1}^{(1)} \end{pmatrix} \tag{1.1.1}$$

Рассмотрим первый слой. На вход каждого нейрона слоя поступает сигнал смещения +1, взвешиваемый *биасом* $w_{0,i}^{(1)}$. Также на вход этого нейрона поступают входные признаки x_i, взвешиваемые *синаптическими весами* $w_{1,j}^{(1)}$ до $w_{m_0,j}^{(1)}$ - веса i-го признака j-го нейрона в первом слое. Для экономии места введём вектор

синаптических весов конкретного нейрона:

$$\vec{w}_j^{(k)} = \begin{pmatrix} w_{0,j}^{(k)} & w_{1,j}^{(k)} & \dots & w_{m_0,j}^{(k)} \end{pmatrix}^T \tag{1.1.2}$$

Индекс в скобках - номер слоя, нижний индекс - номер нейрона. Т.е., $\vec{w}_1^{(1)}$ - вектор синаптических весов 1-го нейрона 1-го слоя.

На выходе i-го нейрона k-го слоя формируется сигнал $u_i^{(k)}$. Выпишем формулу для вычисления сигнала на выходе 1-го нейрона 1-го слоя $u_1^{(1)}$. Опять, в скобках - номер слоя, индекс - номер нейрона. Для этого необходимо посчитать сигнал на выходе сумматора k-го нейрона (*индуцированного локальноо поля*):

Не очень удачно были выбраны индексы, но оставил, как на презентации

$$v_k^{(1)} = \sum_{j=0}^{m_0} w_{j,k}^{(1)} \cdot x_j = [\vec{w}_k^{(1)}]^T \cdot \vec{x} \tag{1.1.3}$$

$$v_2^{(1)} = [\vec{w}_2^{(1)}]^T \cdot \vec{x}$$

где j - номер нейрона, k - номер слоя, $w_{j,k}^{(1)}$ - компоненты ..., $\vec{x}$ - расширенный вектор.

Из приведённых выражений можем получить общее: $W_k^T \cdot \vec{x}$ - совокупность элементов на выходе сумматоров всех нейронов k-го слоя.

Полученные на выходе сумматора значения подвергаются нелинейному преобразованию с помощью функции активации $\varphi(W_k^T \cdot \vec{x})$. Получаем совокупность сигналов на выходе функций активации нейронов. Для первого слоя: $\varphi(W_1^T \cdot \vec{x})$, для второго: $\varphi(W_2^T \cdot \vec{x})$ и т.д.

Для получения из матрицы W_1 матрицы W_2 рисуем таблицу:

Количество входов и выходов скрытых слоёв (без учёта "расширенности"векторов):

	1 слой	2 слой	3 слой
Количество входов	m_0	m_1	m_2
Количество выходов	m_1	m_2	m_3

Выпишем матрицу W_2;

$$W_1 = \begin{pmatrix} w_{0,1}^{(2)} & w_{0,2}^{(2)} & \dots & w_{0,m_2}^{(2)} \\ w_{1,1}^{(2)} & w_{1,2}^{(2)} & \dots & w_{1,m_2}^{(2)} \\ w_{2,1}^{(2)} & w_{2,2}^{(2)} & \dots & w_{2,m_2}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m_1,1}^{(2)} & w_{m_1,2}^{(2)} & \dots & w_{m_1,m_2}^{(2)} \end{pmatrix} \tag{1.1.4}$$

И получим выходы сумматоров нейронов второго слоя:

Починить расширенные вектора!

$$\vec{u}_2 = W_2^T \cdot [\frac{1}{\varphi(W_1^T \cdot \vec{x})}] \tag{1.1.5}$$

В формуле 1 в якобы числителе - добавленный первый элемент, чтобы получить расширенный вектор-столбец.

$$\vec{u}_3 = \varphi(W_3^T [\frac{1}{\varphi(W_2^T \cdot [\frac{1}{\varphi(W_1^T \cdot \vec{x})}])}]) \tag{1.1.6}$$

Для 4-го слоя получим:

Выписать матрицу с рисунка для 4 слоя, поправить формулу и выписать её

Увеличение слоёв нецелесообразно из-за рекурсивной структуры модели многослойного перцептрона (последовательность вложенных блоков), что приводит к уменьшению скорости вычисления, увеличению занимаемой памяти и накопления ошибок. Также по мере увеличения параметров модели, возникает *эффект переобучения*, а по мере увеличения количества слоёв, количество параметров увеличивается экспоненциально.

Возможно применение альтернативных архитектур - например, сети радиальных базисных функций. Также необходимо применение методов регуляризации, например, метод *хирургии мозга* - исключения из каждого слоя неинформативных нейронов или целых слоёв. Существуют методы обучения, подразумевающие цикличность. Более детально см. "Нейронные сети. Полный курс"Саймона Хайкина (?).

Также существует теорема, доказанная Колмогоровым, утв., что многомерная скалярная функция может воспроизводить любую закономерность сколь угодно большой сложности. Теорема об универсальной аппроксимации многослойным перцептроном, утв., что архитектура многослойного перцептрона *теоретически* позволяет с любой заданной точностью синтезировать многомерную по количеству входов векторную функцию, что позволяет строить модели нелинейных дискриминантных границ в пространстве признаков и, соотв., определять принадлежность данных кластерам.

1.2 Системный подход к синтезу алгоритмов обучения многослойного перцептрона

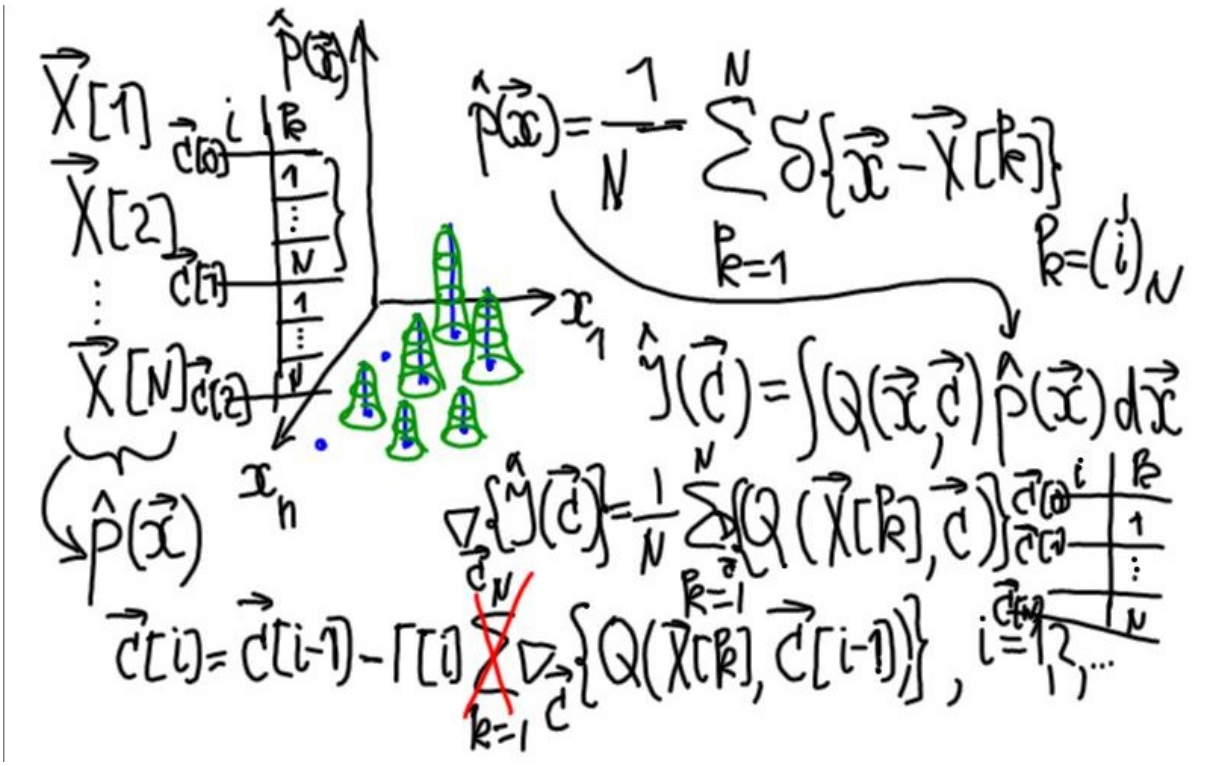

Рис. 2: Системный подход к обучению многослойного перцептрона

Согласно системному подходу к синтезу алгоритмов обучения многослойного перцептрона, необходимо выполнить 3 шага:

1. Выбор цели обучения;
2. Оценка градиента средних потерь целевой функции по параметрам нейронной сети;
3. Использование алгоритмом поиска оптимальных параметров.

Предполагается наличие функционала качества работы системы, оптимальное значение которого и ищется.

Будем рассматривать 3 объекта анализа

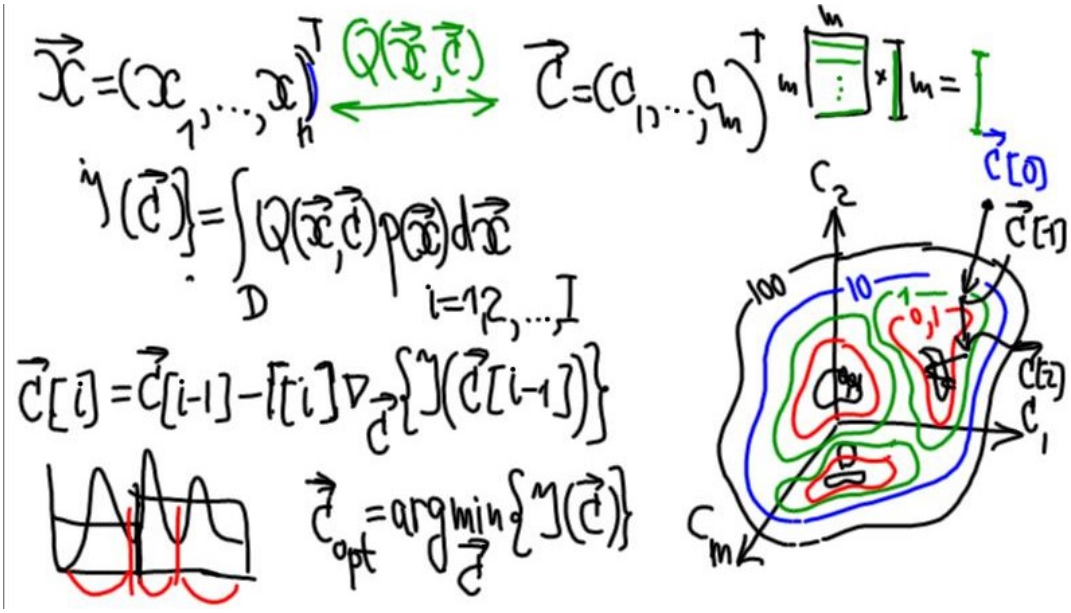

Рис. 3: Обучение нейронной сети

1. Пространство входных воздействий $\vec{x}$ - сигналы, поступающие на вход системы (обучающие и контрольные примеры), и соответствующее пространство входных воздействий, в котором существует область D все возможных входных воздействий;

2. Вектор параметров системы $\vec{C}$ и пространство, ему соответствующее, в которому существуют эквипотенциальные гиперповерхности, все точки которых соотв. определенным значениям функционала качества. Необходимо найти область мимальных/максимальных значений этого функционала качества;

3. Функция стоимости (потерь, приобретений) $Q(\vec{x}, \vec{c})$, определяющая потери/приобретения, которые можно измерить, если на вход системы поступает воздействие $\vec{x}$, а система находится в в состоянии $\vec{C}$.

Набор этих 3 объектов позволяет математически строго сформулировать выбор цели обучения в виде функционала средних потерь (среднего риска). Сделать это несложно, необходимо протестировать область D всех входных воздействий: зафиксировать $\vec{x}$ и выбрать параметры системы $\vec{C}$. Для каждого зафиксированного выбора параметров системы необходимо вычислить потери по выбранной для кажого вхожного воздействия и просуммировать их:

$$J(\vec{C}) = \int_D Q(\vec{x}, \vec{C}) p(\vec{x}) d\vec{x} \tag{1.2.1}$$

Полученное $J(\vec{C})$ - средний риск (средние потери), $p(\vec{x})$ - весовая функция, сглаживающая аномальные значения (плотность распределения вероятностей входных воздействий).

На практике возможны 2 ситуации:

1. Нетипичный случай полной априорной определённости - полное знание входных воздействий (весовой функции $p(\vec{x})$, Типа распределения и тд.);

2. Типичный случай - функция задана набором обучающий примеров

Во втором случае необходимо выполнить предварительную выборочную оценку весовой функции $p(\vec{x})$ по набору обучающих примеров.

Однако, будем считать весовую функцию $p(\vec{x})$ известной, тогда возможно посчитать функционал ошибок $J(\vec{C})$.

1.3 Алгоритм обучения нейронной сети (метод градиентного поиска)

1. Выбрать начальное приближение параметров нейронной сети $\vec{C}[0]$ (0 - номер итерации), и оценить потери;

2. Выбрать оптимальное направление поиска (вектор антиградиента $-\nabla\vec{C}$, указывающий направление, в котором потери убывают с максимальной скоростью);

3. Выбрать шаг поиска (используется процедура стохастической аппроксимации Роббенса-Монро, в соответствии с которой с каждой итерацией длина шага уменьшается обратно пропорционально номеру итерации);

   В полученной точке выполняется подтверждение сходимости (сравнение итераций), вычисление нового направления и шага.

Процедура описывается следуюшим выражением:

$$\vec{C}[i] = \vec{C}[i-1] - \Gamma[i]\nabla\vec{C}\{J(\vec{C}[i-1])\} \tag{1.3.1}$$

Цель обучения:

$$\vec{C}_{opt} = \arg\min_{\vec{C}}\{J(\vec{C})\} \tag{1.3.2}$$

Существуют также методы, не требующие вычилсения частной производной, - метод деформируемого многоранника Нелдера-Мида (симплекс-поиск).

Прикладное нелинейное программирование, Химмельблау

Обучение сводится к решению задачи оптимизации. Существуют ограничения на значения параметров системы в виде равенств или неравенств.

Результаты выполнения алгоритма зависят от выбора начального приближения, что приводит к необходимости выбора нескольких начальных приближений для обнаружения всех локальных минимумов. Существенное преимущество в этом случае обеспечивают генетические алгоритмы.

1.4 Алгоритм обучения нейронной сети для случая набора обучающих примеров

Дистанционная

Опоздал на полчаса

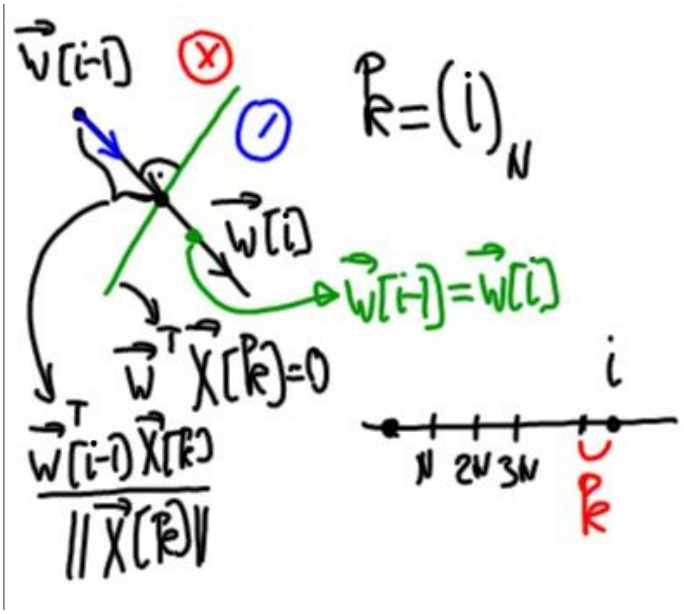

Рис. 4: Аппроксимация многомерной скалярной функции

Геометрическая интерпретация Параметр скорости обучения η выбирают, как .. Переходя от одного к другому обучающему примеру, приближаемся к поверхности разрезанной пирамиды.

3.1 Пример с квадратичной функцией потерь

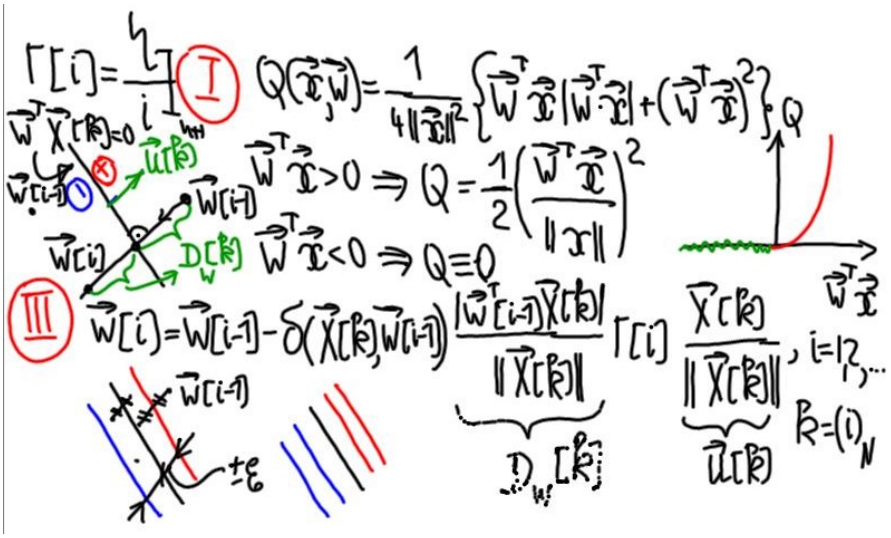

Рис. 5: . . . с квадратичной функцией потерь

Фиксируем потери, пропорциональные квадрату расстояния до гиперплоскости обучающего примера, — потери возрастают квадратично. Методика рассмотрена на предыдущем (дистанционном) занятии.

Вычислить градиент функции потерь на листике

Имея текущий обучающий пример x_k и текущие параметры нейрона $\vec{W}[i-1]$, ..
В W-пространстве нормаль $\vec{u}[k]$ вычисляется как:

$$\frac{\vec{X}_k}{\|\vec{X}_k\|} \tag{3.1.1}$$

Положим, что для k-го обучающего примера параметры неудачны. В этом случае опускаем перепендикуляр к гиперплоскости и вычисляем расстояние до неё:

$$D_W[k] = \frac{|\vec{W}^T[i-1]\vec{X}[k]|}{\|\vec{X}[k]\|} \tag{3.1.2}$$

Долю масштаба определяют компоненты матрицы $\Gamma[i] = \frac{\eta}{i} I_{n\times m}$, где $I_{n\times m}$ - единичная матрица.

Харман, Математические основы математической вычислительной томографии, глава "PBT"

Также в книге присутствует алгоритм, в котором для гиперплоскости каждого обучающего примера выбирается некая погрешность и гиперплоскость обволакивается полосой шириной $\pm\varepsilon$ в связи с присутствием погрешностей в "реальных"примерах.

Для простоты анализа здесь был выбран режим последовательного обучения, однако ранее были обсуждены также алгоритмы *смешанного обучения.*

Рекомендовано выписать этот алгоритм для простой и экспоненциальной (задача со звёздочкой) скользящей средней

Здесь рассмотрен случай линейно разделимых разделимых векторов образов, исходя из предположения, что система линейных неравенств имеет минимум одно решение. Иначе будет минимум одна зона в W-пространстве, характеризуемая количеством ошибочных классификаций и необходим алгоритм, строящий *оптимальную гиперплоскость*.

Для *оптимальной гиперплоскости* количество неизбежных ошибочных классификаций минимально. Такой метод был реализован Владимиров Вапником в *методе опорных векторов*.

Булавский (НГУ) предложил симплекс-метод для решения несовеместных систем линейных неравенств — поиска компромиссного решения. Метод применим для случая линейно неразделимых векторов образов и является альтернативой методу опорных векторов. Метод не получил широкого распространения, имеется только препринт.

Препринт есть у ЛЛВ

3.2 Аппроксимация многомерной скалярной функции

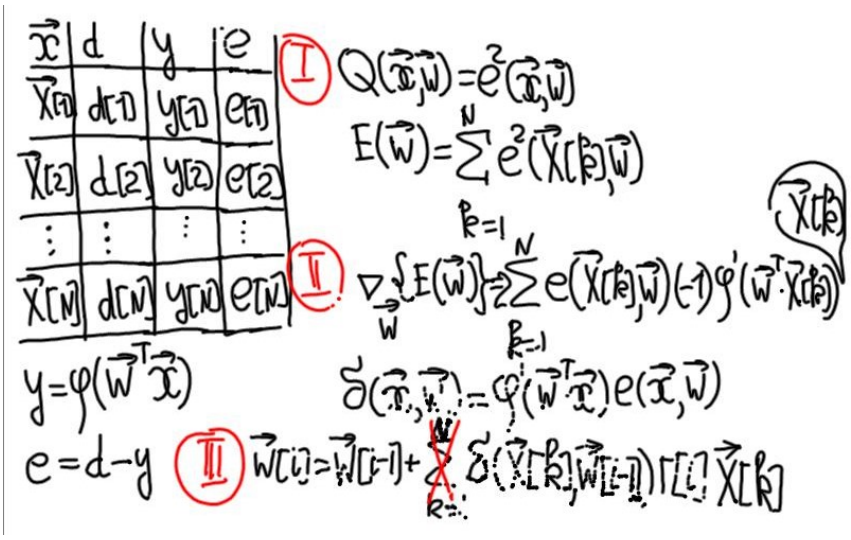

Рис. 6: Аппроксимация многомерной скалярной функции

Рассмотрим альтернативную дихотомии задачу - аппроксимацию многомерной скалярной функции, т.е. синтез закономерности, скрытой в данных, с помощью многослойного перцептрона. В данном случае стоит воспринимать прецептрон как устройство воспроизведения математической модели.

$\vec{x}$	d	y	e
$\vec{X}[1]$	$d[1]$	$y[1]$	$e[1]$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vec{X}[N]$	$d[N]$	$y[N]$	$e[N]$

Таблица 1: Табличная форма задания многомерной скалярной функции

Имеется табличная форма задания многомерной скалярной функции (3.2), набор обучающих примеров в виде совокупности факторов (входных аргументов) $\vec{x}[i]$, d — желаемое выходной значение скрытой закономерности, $y = \varphi(\vec{W}^T\vec{x})$ — фактическое значение (реакция) перцептрона и ошибка $e = d - y$.

Необходимо синтезировать математическую модель с помощью перцептронного нейрона. $\vec{x}$ — обобщенный вектор образов с $+1$ в первой позиции, $\vec{W}$ — обобщенный вектор синаптических весов с биасом в первой позиции.

Будем выполнять задачу согласно этапам системного подхода.

3.2.1 Выбор модели фунции стоимости

Выберем квадратичную функцию ошибок:

$$Q(\vec{x}, \vec{W}) = e^2(\vec{x}, \vec{W}) \tag{3.2.1}$$

Тогда средние потери будут равны:

$$E(\vec{W}) = \sum_{k=1}^{T} e^2(\vec{X}[k], \vec{W}) \tag{3.2.2}$$

3.2.2 Поиск градиента функции ошибок

Необходимо найти в W-пространстве такую точку, которая минимизировала бы суммарную энергию ошибок $E(\vec{W})$ 3.2.2.

$$\nabla_{\vec{W}}\{E(\vec{W})\} = 2\sum_{k=1}^{N} e(\vec{X}[k], \vec{W}) \cdot (-1) \cdot \varphi'(\vec{W}^T \cdot \vec{X}[k])\vec{X}[k] \tag{3.2.3}$$

В первую очередь интересно произведение первой производной функции активации и функции ошибки, которую обозначим $\delta(\vec{x}, \vec{W})$:

$$\delta(\vec{x}, \vec{W}) = \varphi'(\vec{W}^T \cdot \vec{x}) \cdot e(\vec{x}, \vec{W}) \tag{3.2.4}$$

и будем называть *локальным градиентом*. Он введён, поскольку при выполнении поиска выгоднее находиться в области ненулевых значений функции активации (иначе итерация "пропадает").

3.2.3 Применение алгоритма градиентного поиска

Распишем алгоритм градиентного поиска (пакетный режим):

$$\vec{W}[i] = \vec{W}[i-1] + \sum_{k=1}^{N} \delta(\vec{X}[k], \vec{W}[i-1]) \cdot \vec{\Gamma[i]} \cdot \vec{X}[k] \tag{3.2.5}$$

Рекомендовано выписывать алгоритм слева направо - скаляр, матрица, вектор.

Для усиления шумового эффекта игнорируется усреднение по всем примерам (удаление знака суммы) и выполняется пересчёт после каждого нового значения - последовательное обучение:

$$\vec{W}[i] = \vec{W}[i-1] + \delta(\vec{X}[k], \vec{W}[i-1]) \cdot \vec{\Gamma[i]} \cdot \vec{X}[k] \tag{3.2.6}$$

Режим последовательного обучения реализует *оптимизирующий алгоритм Уидроу-Хоффа* 3.2.6, иначе называемый *дельта-правилом*.

При введении сглаживания с помощью модели EMA получим *обобщённое дельта-правило*.

3.3 Алгоритм обратного распространения ошибок обучения многослойного перцептрона

Lec_ neuro_ 08.docx

Основным параметром алгоритма обучения Уидроу-Хоффа является ошибка реакции, которая недоступна при обучении нейронов скрытых слоёв. Эту проблему решает алгоритм обратного распространения ошибок.

Упростим схему индексации нейронов (см. 7):

Для этого примем следующие соглашения:

1. Для обозначения 1-го скрытого слоя используем символ i, нейроны 2-го скрытого слоя обозначим символами j, выходные нейроны обозначим символами k, l - символ линейных нейронов 0-го слоя.

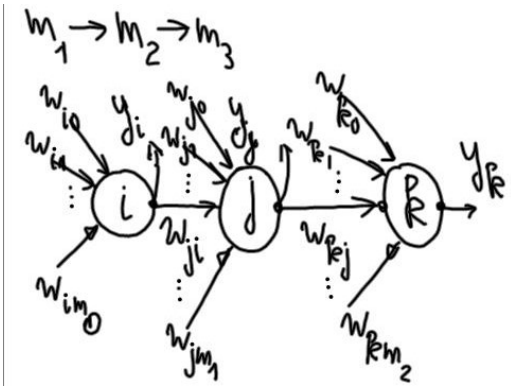

Рис. 7: 1233456

2. Связь между нейронами младшего и старшего слоя (синаптический вес) — Wji, j — символ нейрона старшего слоя, i — символ нейрона младшего слоя, для биаса второй индекс 0.

3. Во входном слое m_0 нейронов (функциональных входов), во 2-м слое m_1 нейронов, в 3-м слое m_2 нейронов.

Алгоритм обратного распространения содержит 2 вычислительных этапа

1. **Этап прямого распространения.** В соответствии с математической моделью многослойного перцептрона можем распространить все сигналы в прямом направлении — посчитать фактические реакции нейронов скрытых слоёв и выходные реакции нейронов.

   В финале этого этапа обладаем выходными реакциями нейронов, на основе которых можно вычислить сигналы ошибок и локальные градиенты, а значит, применить дельта-правило и скорректировать параметры нейронов выходного слоя.

2. **Этап обратного распространения.** (пунктирные линии рис 4.2 ворда). На этом этапе распространяются сигналы ошибок для нейронов скрытых слоёв.

3.3.1 Этап прямого распространения

Этап начинается с расчёта фактических реакций на выходе нейронов 1-го скрытого слоя.

Рассчитаем сигнал на выходе сумматора текущего i-го нейрона на выходе 1-го скрытого слоя, на вход которого поступает сигнал смещения „+1“, взвешиваемый биасом, и входные признаки x_m0, взвешиваемые синаптическим весом w_{im_0}.

На выходе сумматоров получим:

$$v_i(n) = \sum_{L=0}^{m_0} w_{il}(n)x(n) \tag{3.3.1}$$

n - номер итерации.

Подвергаем его нелинейному преобразованию и получаем реакции на выходе нейронов 1-го скрытого слоя:

$$y_i(n) = \varphi(v_i(n)), \quad i = \overline{1, m_1} \tag{3.3.2}$$

Посчитаем сигналы на выходе сумматоров нейронов 2-го скрытого слоя:

$$v_j(n) = \sum_{i=0}^{m_1} w_{ji}(n)y_i(n) \tag{3.3.3}$$

На выходе нейронов 2-го скрытого слоя после нелийненого преобразования:

$$y_j(n) = \varphi(v_j(n)), \quad j = \overline{1, m_2} \tag{3.3.4}$$

Аналогично на выходе сумматоров нейронов 3-го скрытого слоя:

$$v_k(n) = \sum_{j=0}^{m_2} w_{kj}(n) y_j(n) \tag{3.3.5}$$

На выходе нейронов 3-го скрытого слоя после нелинейного преобразования:

$$y_k(n) = \varphi(v_k(n)), \quad k = \overline{1, m_3} \tag{3.3.6}$$

Зная фактические реакции выходных нейронов, вычислим набор ошибок:

$$e_k(n) = d_k(n) - y_k(n) \tag{3.3.7}$$

Вычисляем величину локального градиента выходных нейронов:

$$\delta_k(n) = \varphi'(v_k(n)) \cdot e_k(n) \tag{3.3.8}$$

Алгоритм коррекции k-го выходного нейрона:

$$w_{kj}(n) = w_{kj}(n-1) + \frac{\eta}{N} \cdot \delta_k(n) \cdot y_j(n), k = \overline{1, m_3} \tag{3.3.9}$$

Этап прямого распространения закончен.

3.3.2 Этап обратного распространения

В соответствии с методом потоковых графов, разработанным Станиславом Осовским и описанным в книге „Нейронные сети для обработки информации“, любую сеть можно представить в виде набора графов, имеющих узлы (сигналы) и рёбра (направления распространения сигналов и коэффициенты усиления). Если рёбра сходятся в одном узле, то это означает суммирование соответствующих сигналов.

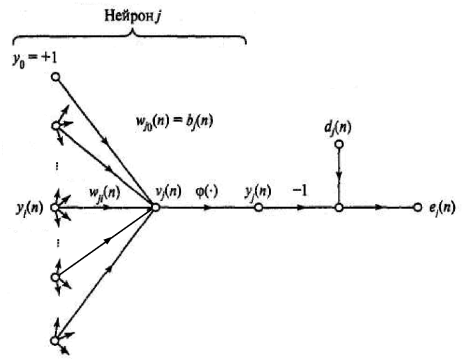

Рис. 8: Граф передачи сигнала в пределах j-го нейрона

Для нейронов выходного слоя имеем сигналы ошибки $e_1(n), \ldots, e_k(n), \ldots, e_{m_3}$, которые умножаются на вычисленные значения функции активации $\varphi'(v_k(n))$ и получаем значения локальных градиентов выходных нейронов $\delta_k(n)$, а затем используем алгоритм обратного распространения ошибок.

Основной результат заключается в правиле вычисления ошибок для нейронов младшего слоя. Показано, что сигнал ошибки может быть вычислен в виде взвешенной суммы локальных градиентов нейронов n-го слоя:

$$e_j(n) = \sum_{k=1}^{m_3} w_{kj}(n) \delta_k(n) \tag{3.3.10}$$

Возможно, суммирование должно быть от 0.

Вычислив ошибку для нейронов второго слоя, вычислим для него локальный градиент, т.е. распространим ошибку на шаг назад:

$$\delta_j(n) = e_j(n) \cdot \varphi'(v_j(n)) \tag{3.3.11}$$

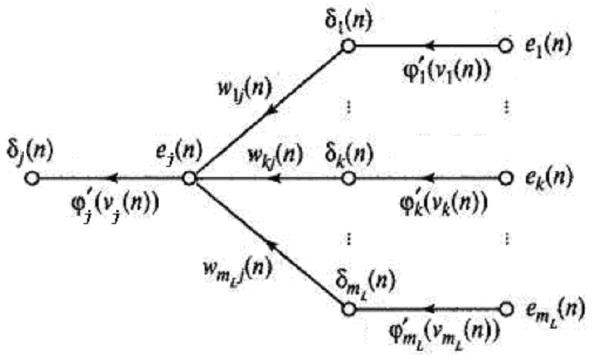

Рис. 9: Направление двух потоков сигналов в многослойном перцептроне

Значит, возможно произвести „редактирование“синаптических весов нейронов второго слоя. Т.е.:

$$w_{ji}(n) = w_{ji}(n-1) + \frac{\eta}{N} \cdot \delta_j \cdot y_i(n) \tag{3.3.12}$$

Для нейронов первого скрытого слоя:

$$e_i(n) = \sum_{j=1}^{m_2} w_{ji}(n)\delta_j \tag{3.3.13}$$

Возможно, суммирование должно быть от 0.

$$\delta_i(n) = e_i(n) \cdot \varphi'(v_i(n)) \tag{3.3.14}$$

Для входного слоя получим:

$$w_{il}(n) = w_{il}(n-1) + \frac{\eta}{N} \cdot \delta_i \cdot x_l \tag{3.3.15}$$

Произошло распространение ошибок и локального градиента в обратном направлении, что позволило применить обычное дельта-правило.

Новая

4.1 Практические рекомендации по улучшению сходимости алгоритма обратного распространения ошибок