

# Нейронные сети. Лекции.

Лабунец Л.В. & Паймин Антон

## Содержание

<b>Лекция</b>	<b>2</b>
1.1 Математическая модель многослойного перцептрона . . . . .	2
1.2 Системный подход к синтезу алгоритмов обучения многослойного перцептрона . . . . .	4
1.3 Алгоритм обучения нейронной сети (метод градиентного поиска) . . . . .	5
<b>Лекция 04.09 (дистанционная)</b>	<b>6</b>
2.1 Алгоритм обучения нейронной сети для случая набора обучающих примеров . . . . .	6
2.1.1 Ядерная модель Парзена-Розенблатта . . . . .	7
2.1.2 Пакетный, последовательный и комбинированный режимы обучения . . . . .	8
2.1.3 Алгоритм обучения перцептронного нейрона . . . . .	9
<b>Лекция 16.04 (Опоздал на полчаса)</b>	<b>10</b>
3.1 Пример с квадратичной функцией потерь . . . . .	10
3.2 Аппроксимация многомерной скалярной функции . . . . .	11
3.2.1 Выбор модели функции стоимости . . . . .	12
3.2.2 Поиск градиента функции ошибок . . . . .	12
3.2.3 Применение алгоритма градиентного поиска . . . . .	13
3.3 Алгоритм обратного распространения ошибок обучения многослойного перцептрона . . . . .	13
3.3.1 Этап прямого распространения . . . . .	14
3.3.2 Этап обратного распространения . . . . .	15
<b>Новая</b>	<b>16</b>
4.1 Практические рекомендации по улучшению сходимости алгоритма обратного распространения ошибок . . . . .	16

# 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.

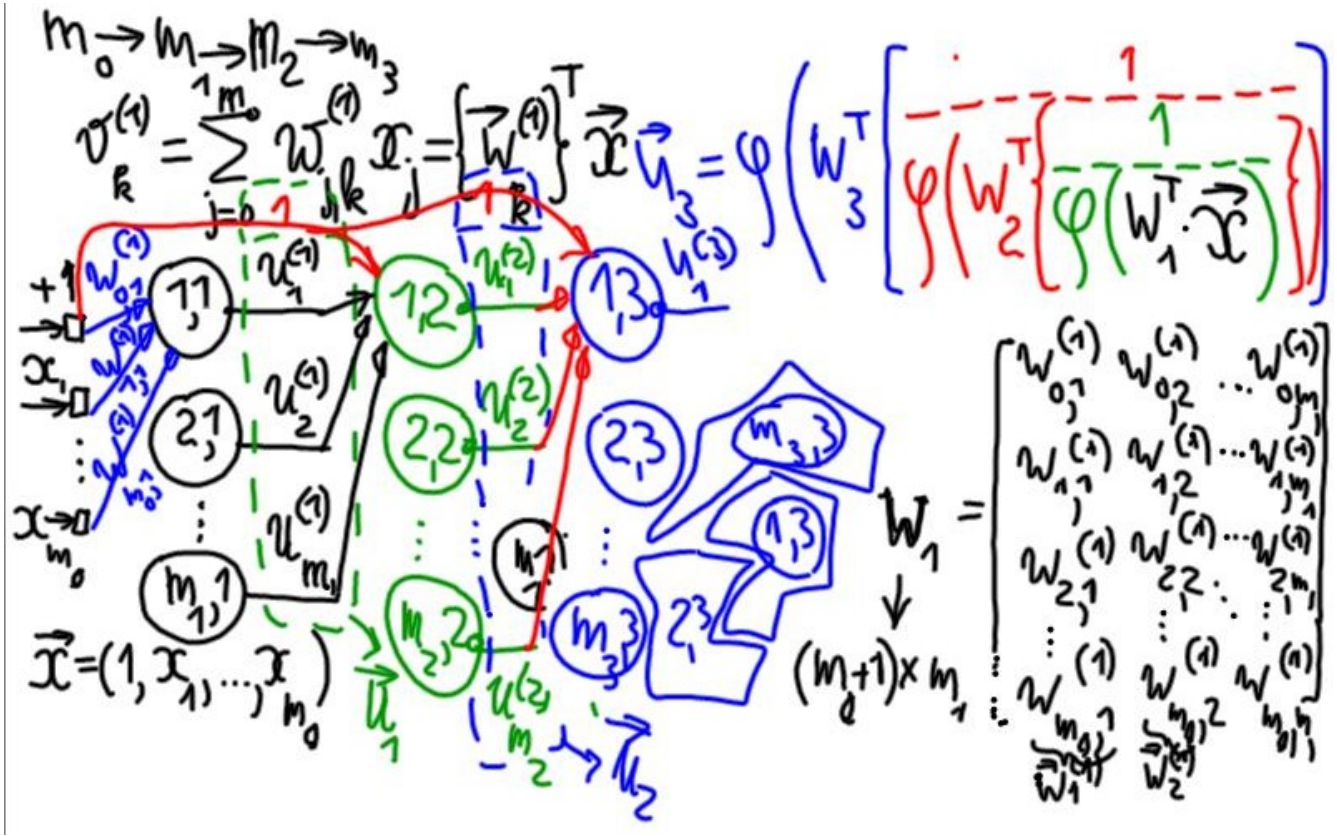


Рис. 1: Структура многослойного перцептрона

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

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

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:

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

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 локального поля*):

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

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

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

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  $W_2$ :

$$W_2 = \begin{pmatrix} w_{0,1}^{(2)} & w_{0,2}^{(2)} & \dots & w_{0,m_2}^{(2)} \\ w_{1,1}^{(2)} & w_{1,2}^{(2)} & \dots & w_{1,m_2}^{(2)} \\ w_{2,1}^{(2)} & w_{2,2}^{(2)} & \dots & w_{2,m_2}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m_1,1}^{(2)} & w_{m_1,2}^{(2)} & \dots & w_{m_1,m_2}^{(2)} \end{pmatrix} \quad (1.1.4)$$

И получим выходы сумматоров нейронов второго слоя:

**Починить расширенные вектора!**

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

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

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

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, утверждающая, что многомерная скалярная функция может воспроизводить любую закономерность сколь угодно большой сложности. Теорема об универсальной аппроксимации многослойным перцептроном утверждает, 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, соответственно, определять принадлежность данных кластерам.

## 1.2 Системный подход к синтезу алгоритмов обучения 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.

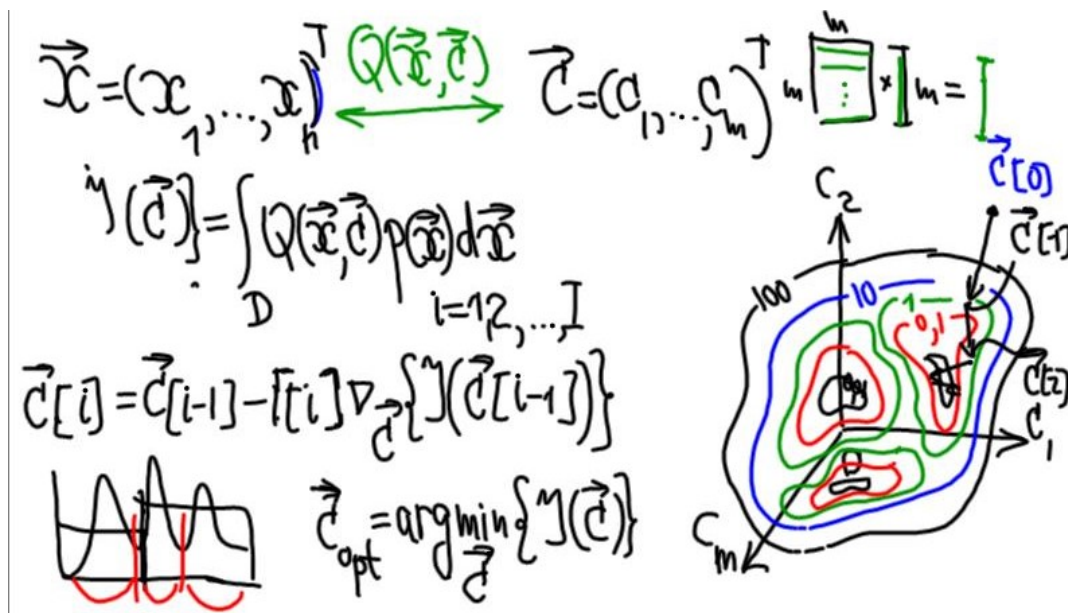


Рис. 2: Обучение нейронной сети

Будем рассматривать 3 объекта анализа

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 котором существуют эквипотенциальные гиперповерхности, все точки которых соотв. определенным значениям функционала качества. Необходимо найти область минимальных/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 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 каждого входного воздействия и просуммировать их:

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

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 обучающих 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 следующим выражением:

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

Здесь  $\Gamma[i]$  — матрица усиления, управляющая выбором величины шага поиска. Если она является диагональной, то по каждому признаку шаги будут независимы (характерно для случая гиперболоидальной топологии гиперповерхностей равных значений функции стоимости).

Цель обучения:

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

Существуют также методы, не требующие вычисления частной производной, — метод деформируемого многогранника Нелдера-Мида (симплекс-поиск).

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.

## Лекция 04.09 (дистанционная)

### 2.1 Алгоритм обучения нейронной сети для случая набора обучающих примеров

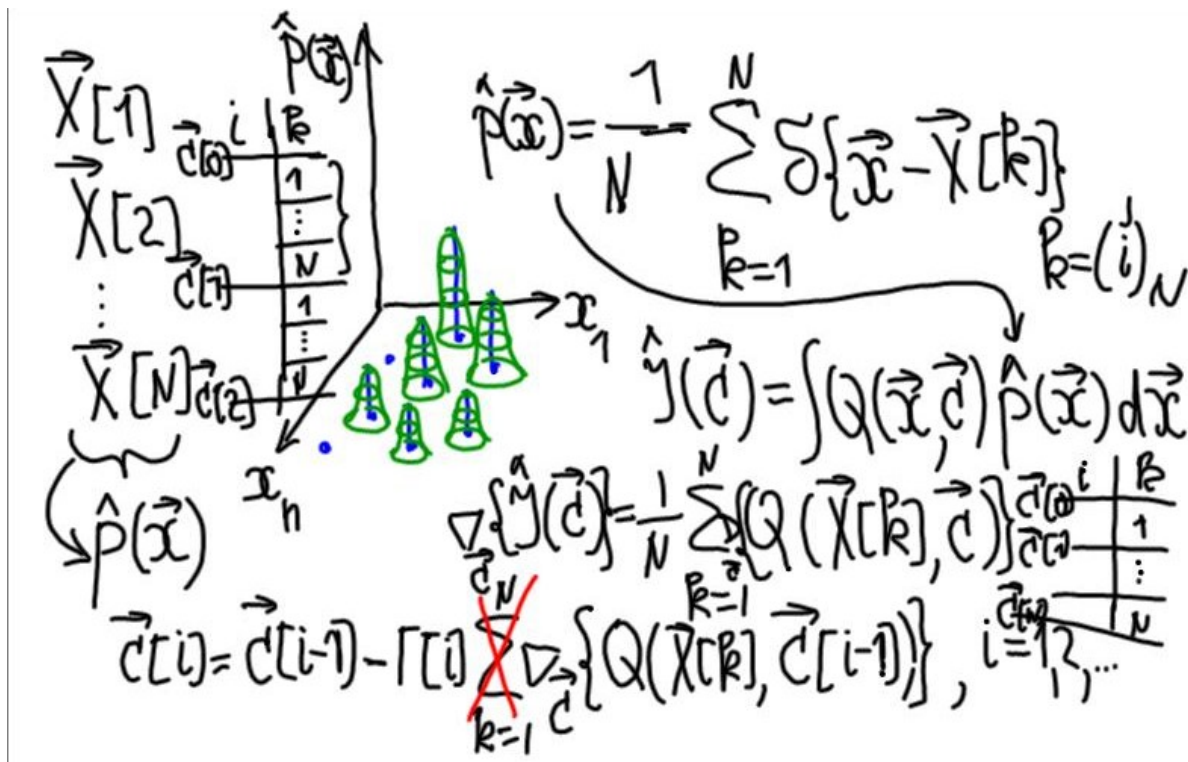


Рис. 3: Системный подход к обучению многослойного перцептрона

В подавляющем большинстве практических приложений весовая функция  $p(\vec{x})$ , входящая в выражение функционала среднего риска 1.2.1 задана неявным образом в виде набора обучающих примеров  $\hat{p}(\vec{x})$ .

Здесь и далее обучающие примеры, ранее обозначавшиеся  $\vec{x}$ , будем обозначать  $\vec{X}$ . Объем выборки -  $N$  обучающих примеров (см. левый столбец на 3), размерность пространства признаков  $\{x_1, \dots, x_n\} - n$ .

Таким образом в  $x$ -пространстве имеется набор обучающих примеров  $\vec{X}[1], \dots, \vec{X}[N]$ , визуализированных диаграммой рассеяния (график на 3). Точка задаётся радиус-вектором, набор которых задаёт набор обучающих примеров.

Для приближения функционала среднего риска к реальности, дополним его ещё одной осью  $\hat{p}(\vec{x})$ , вдоль которой будем откладывать значения оценки весовой функции, где крышечка показывает, что это оценка.

В теории искусственного интеллекта существует огромное количество непараметрических, параметрических и полупараметрических описаний распределений, в т.ч. и вероятностных. Например, примером параметрического описания многомерного распределения является модель конечной смеси стандартных распределений, для идентификации параметров которой служат ЕМ-алгоритмы. Пример полупараметрической оценки — ядерная оценка Парзена-Розенблатта, одномерная модель ПРВ в виде гистограммы, сглаженной сдвигом (с обобщением до многомерной модели).

### 2.1.1 Ядерная модель Парзена-Розенблатта

Идея заключается в ассоциировании функции ядра каждой точкой в подпространстве признаков с каждым наблюдением (обучающим примером) — „гиперхолмы“ на рисунке 3. Рассмотрим простейший случай, когда оценка обладает единственным параметром сглаживания для всех наблюдений — *неадаптивная ядерная оценка Парзена-Розенблатта*.

Такие функции обладают важными свойствами. Гиперобъём под поверхностью функции ядра всегда равен 1 (функция ядра интегрируема с 1). Также, как и у любой оценки многомерного ПРВ, присутствует параметр сглаживания (в одномерном случае — ширина разрядного интервала, в многомерном случае — гиперобъём гиперпараллелепипеда), регулирующий степень убывания высоты холма по мере удаления текущего радиуса-вектора (аргумента ПРВ) от центра холма (выбранного обучающего примера).

Если текущее значение совпадает с выбранным примером, то высота максимальна. По мере удаления от *центра опорной области* (центра холма) высота асимптотически стремится к 0. Аналогично случаю гистограммной оценки, при стремлении параметра сглаживания к 0 (масштаб функции ядра нулевой) высота будет неограниченно возрастать, а опорная область асимптотически стягиваться к 0 и приближаться к дельта-функции Дирака — набору „иглоков“ единичного объёма в гиперпространстве. Такие „иглоки“ не могут иметь общие области и, таким образом, ядерная оценка при таком выборе параметров превращается в „лес иглоков“ и (что плохо) сглаживание экспериментальных производится не будет — *эмпирическая оценка весовой функции*.

$$\hat{p}(\vec{x}) = \frac{1}{N} \sum_{k=1}^N \delta(\vec{x} - \vec{X}[k]) \quad (2.1.1)$$

Для эмпирической оценки суммируем дельта-функции, ассоциированные с обучающими примерами и масштабируем значение в зависимости от числа обучающих примеров.

*Обещано позже положительное свойство*

В ядерной оценке Парзена-Розенблатта увеличивается параметр сглаживания и опорная область расширяется по мере увеличения параметра сглаживания, а высота „холмов“ начинает уменьшаться и ближайшие обучающие примеры начинают перекрываться, происходит постепенное сглаживание совокупности обучающих примеров и получаем „гиперрельеф“.

До некоторого момента при определённом числовом диапазоне параметров сглаживания гиперрельеф сохраняет свою морфологию — наблюдается определённое количество вершин, что свидетельствует о наличии кластерной структуры в данных и характеризует статистическую устойчивость оценки ПРВ при выборе параметра сглаживания из этого диапазона.

При дальнейшем увеличении параметра сглаживания опорные области функций-ядер начинают чрезмерно увеличиваться и кластерная структура гиперрельефа начинает исчезать и в асимптотике стремится к плато — равномерному распределению по всем признакам.

Существует набор методов, позволяющих выбрать оптимальное значение параметров сглаживания, который позволяет обнаружить кластерную структуру, т.е. *многомодовый характер* гиперрельефа в  $n$ -мерном пространстве признаков.

Один из таких методов — *метод скользящей проверки* выбора оптимального параметра сглаживания. Существуют также адаптивные ядерные оценки, в которых для каждого наблюдения выбирается свой оптимальный параметр сглаживания и каждый „гиперхолм“ будет обладать своей оптимальной опорной областью (похоже на выбор характеристик масштаба модели конечной смеси стандартных распределений). В ещё более сложных оценках индивидуальными являются не только параметры сглаживания, но и модель функции ядра, например, *адаптивная фильтрационная ядерная оценка плотности распределения вероятности*. Это отдельное направление, связанное с построением оптимальных оценок одномерных ПРВ.

*Есть статья на labnet.ru - Рандомизация многомерных распределений в метрике Махаланобиса*

Конечная цель — получить возможность проинтегрировать 1.2.1 в конечном виде, чтобы получить в конечном виде оценку среднего риска, поскольку необходимо сформировать оценки компонентов вектора градиента и реализовать этот алгоритм обучения для практических приложений.

В этом случае поможет эмпирическая оценка — плата за попытку получить выражение 1.2.1 является грубая эмпирическая оценка весовой функции, которая преднамеренно игнорирует процедуру сглаживания обучающих данных.

Преимущество — фильтрующее свойство дельта-функции. При подстановке 2.1.1 в 1.2.1:



$$\hat{J}(\vec{C}) = \int_D Q(\vec{x}, \vec{C}) \hat{p}(\vec{x}) d\vec{x} = \frac{1}{N} \int_D Q(\vec{x}, \vec{C}) \sum_{k=1}^N \delta(\vec{x} - \vec{X}[k]) d\vec{x} \quad (2.1.2)$$

при смене порядка суммирования и интегрирования дельта-функция „профильтрует“ значения весовой функции для каждого обучающего примера:

$$\hat{J}(\vec{C}) = \frac{1}{N} \sum_{k=1}^N \int_D Q(\vec{x}, \vec{C}) \cdot \delta(\vec{x} - \vec{X}[k]) d\vec{x} \quad (2.1.3)$$

Произведение функции стоимости и дельта-функции, ассоциированной с конкретным обучающим примером  $Q(\vec{x}, \vec{C}) \cdot \delta(\vec{x} - \vec{X}[k])$  даст значение функции стоимости для конкретного обучающего примера.

Таким образом, эмпирическая оценка функционала среднего риска представляет собой сумму функции стоимости по всем обучающим примерам для выбранного вектора параметров системы, нормированную по количеству обучающих примеров, т.е. выборочное среднее мат. ожидания функции стоимости:

$$\hat{J}(\vec{C}) = \frac{1}{N} \sum_{k=1}^N Q(\vec{X}[k], \vec{C}) \quad (2.1.4)$$

Применение выборочного среднего означает неробастность функции, т.е. неустойчивость к аномалиям. Финальный алгоритм обучения будет обладать значимой шумовой компонентой и игнорирование влияния аномалий тоже скажется на качестве текущих оценок параметров обучения системы в процессе их настройки.

Поскольку вектор градиента есть набор частных производных, а операции суммирования и дифференцирования перестановочны, то запишем выражение для эмпирической оценки вектора градиента:

$$\nabla_{\vec{C}} \hat{J}(\vec{C}) = \frac{1}{N} \sum_{k=1}^N \nabla_{\vec{C}} Q(\vec{X}[k], \vec{C}) \quad (2.1.5)$$

Подставим полученную в конечном виде оценку градиента в синтезированный ранее алгоритм, получим адаптированный к случаю обучения системы по набору параметров алгоритм:

$$\vec{C}[i] = \vec{C}[i-1] - \Gamma[i] \sum_{k=1}^N \nabla_{\vec{C}} \{Q(\vec{X}[k], \vec{C}[i-1])\} \quad (2.1.6)$$

Пока операцию суммирования (усреднения по всем возможным значениям компонент вектора градиента) сохраняем.

### 2.1.2 Пакетный, последовательный и комбинированный режимы обучения

Проанализируем обучение системы при наличии этого усреднения. В этом случае процедура обучения реализует *пакетный режим* обучения (см. 1).

В распоряжении имеется набор из  $N$  обучающих примеров, происходит инициализация алгоритма обучения начальными значениями параметров системы. Прежде чем откорректировать начальное приближение параметров системы в соответствии с существованием этой суммы, необходимо перебрать все обучающие примеры, вычислить значения функции стоимости и частные производные для всех обучающих примеров, получить в  $C$ -пространстве набор направлений оптимальных поисков (свой для каждого примера), просуммировать эти вектора, взять усреднённое направление поиска с отрицательным знаком, гарантирующее приемлемую скорость убывания, выбрать величину шага и только после этого от  $\vec{C}[0]$  перейти к  $\vec{C}[1]$ . На следующей итерации шаги повторяются.

Пакетный режим обучения требует максимально возможной памяти.

Альтернативой пакетному режиму обучения является *последовательный режим обучения* (см. 2). Для этого опускаем операцию суммирования всех возможных направлений и после каждого обучающего примера выполняем коррекцию:

$$\vec{C}[i] = \vec{C}[i-1] - \Gamma[i] \nabla_{\vec{C}} \{Q(\vec{X}[k], \vec{C}[i-1])\} \quad (2.1.7)$$

при последовательном режиме обучения эффект шумовой компоненты существенно усиливается, поэтому на практике выбирают „золотую середину“, что согласуется с процедурами сглаживания временных рядов.



Таблица 1: Описание пакетного режима обучения

$i$ — Номер итерации обучения	$k$ — Номер обучающего примера	
$\vec{C}[0]$	1	1
		$\vdots$
		$N$
$\vec{C}[1]$	2	1
		$\vdots$
		$N$
$\vdots$	$\vdots$	$\vdots$

Таблица 2: Описание последовательного режима обучения

$i$ — Номер итерации обучения	$k$ — Номер обучающего примера	
$\vec{C}[0]$	1	1
$\vec{C}[1]$	2	2
$\vdots$	$\vdots$	$\vdots$
$\vec{C}[N]$	N	N

Выберем некоторую компоненту в параметрах системы — временной ряд в терминологии номера итерации обучения. Этот ВР будет иметь интересующую трендовую составляющую, шумовую и, возможно, циклические компоненты.

Пользуясь подходами выделения трендовых компонент ВР, используем цифровую фильтрацию: для текущего обучающего примера и нескольких предыдущих примеров (текущую и несколько предыдущих итераций), количество которых определяет интервал сглаживания. Обладая этой информацией, можно вычислить скользящую среднюю (или экспоненциальную скользящую среднюю) и реализовать подавление шумовой компоненты, выделив трендовые составляющие.

Таким образом, *комбинированный режим обучения* предполагает учёт текущего обучающего примера и нескольких предыдущих, что обеспечивает подавление шумовой составляющей и циклической компоненты.

### 2.1.3 Алгоритм обучения перцептронного нейрона

Прежде чем переходить к алгоритму обучения многослойного перцептрона, рассмотрим синтез алгоритмов обучения перцептронного нейрона, как основного структурного элемента многослойного перцептрона.

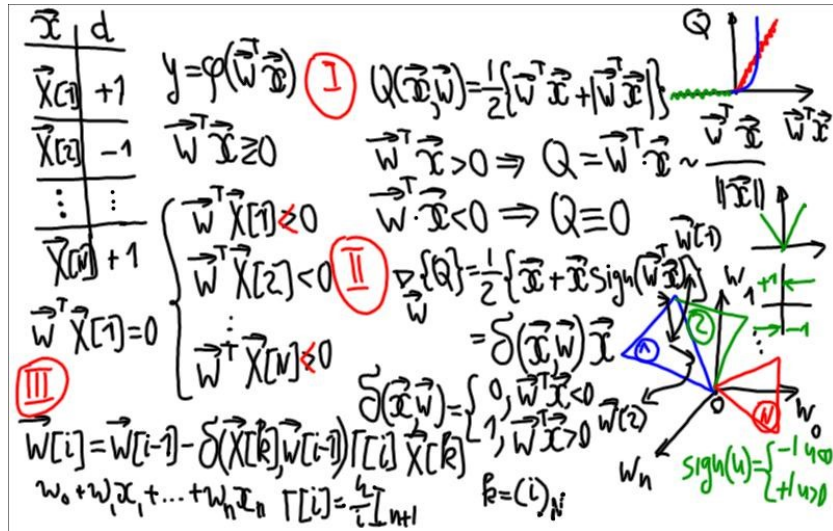


Рис. 4: Алгоритм обучения перцептронного нейрона

## Лекция 16.04 (Опоздал на полчаса)

Алгоритмы обучения перцептронного нейрона в рамках задачи дихотомии в соответствии с системным подходом реализуют 3 этапа (см. выше) В пространстве параметров обучающие примеры — это числа, гиперплоскости, проходящие через начало координат, пропорционально направляющим косинусам. Для случая линейно разделимых образов регулярная структура (разрезанная пирамида в виде facets) делит пространство параметров на + и -. В отрицательную часть необходимо стремиться.

Сегментарная траектория поиска в области отрицательных значений. Мы считаем, что система неравенств имеет решение.

Для каждого обучающего примера, фаза наказания: X-числа, W-буквы. Параметр скорости - 1 - ортогонально. Параметр скорости 2 - зеркальное отражение. Обычно берут долю

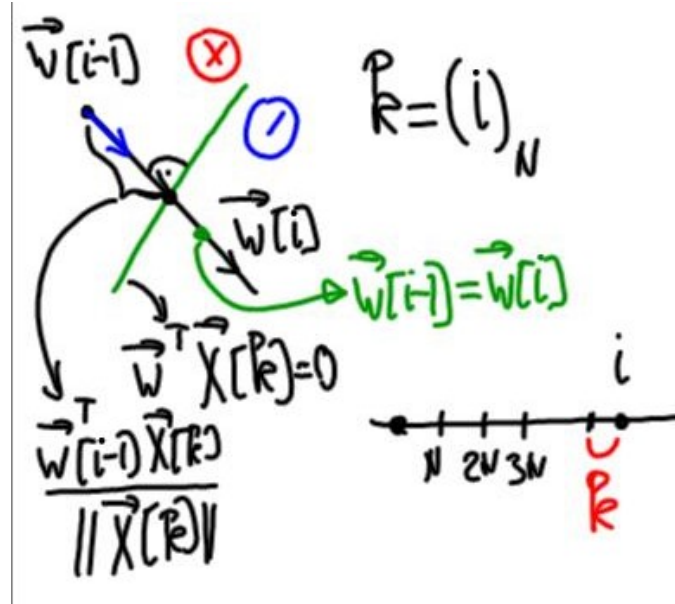


Рис. 5: Аппроксимация многомерной скалярной функции (hyperflat\_7\_1)

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

### 3.1 Пример 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:

$$\frac{\vec{X}_k}{\|\vec{X}_k\|} \quad (3.1.1)$$

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

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

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

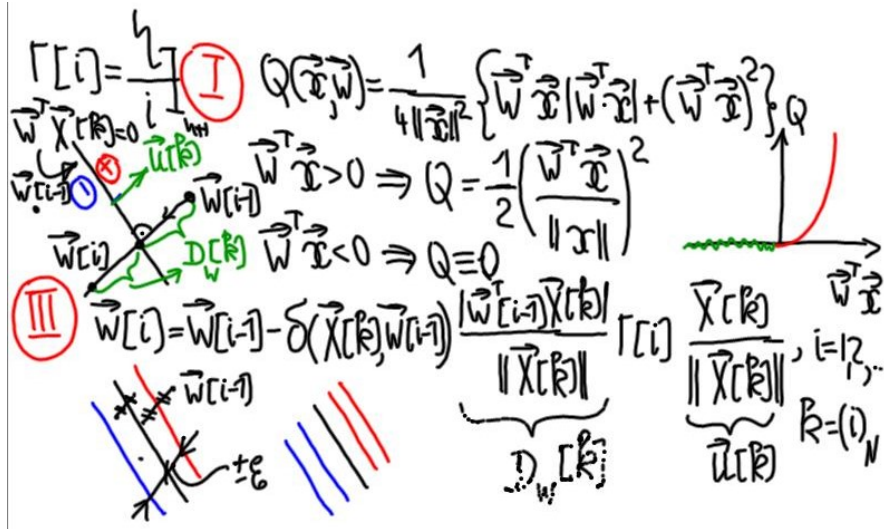


Рис. 6: ... с квадратичной функцией потерь

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

Также в книге присутствует алгоритм, в котором для гиперплоскости каждого обучающего примера выбирается некая погрешность и гиперплоскость обволакивается полосой шириной  $\pm \epsilon$  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 Владимиром Вапником в методе опорных векторов.

Булавский (НГУ) предложил симплекс-метод для решения несовместных 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

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

Handwritten notes showing a table of input-output pairs and associated equations for a neural network model.

$\vec{x}$	$d$	$y$	$e$
$\vec{x}[1]$	$d[1]$	$y[1]$	$e[1]$
$\vec{x}[2]$	$d[2]$	$y[2]$	$e[2]$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vec{x}[N]$	$d[N]$	$y[N]$	$e[N]$

Equations and notes:

- $Q(\vec{x}, \vec{w}) = e^2(\vec{x}, \vec{w})$
- $E(\vec{w}) = \sum_{k=1}^N e^2(\vec{x}[k], \vec{w})$
- $\nabla_{\vec{w}} \{E(\vec{w})\} = 2 \sum_{k=1}^N e(\vec{x}[k], \vec{w}) (-1) \varphi'(\vec{w}^T \cdot \vec{x}[k])$
- $y = \varphi(\vec{w}^T \vec{x})$
- $\delta(\vec{x}, \vec{w}) = \varphi'(\vec{w}^T \vec{x}) e(\vec{x}, \vec{w})$
- $e = d - y$
- $\vec{w}[k] = \vec{w}[k-1] + \delta(\vec{x}[k], \vec{w}[k-1]) \vec{x}[k]$

Рис. 7: Аппроксимация многомерной скалярной функции

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

Таблица 3: 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 выходное 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 функции стоимости

Выберем квадратичную функцию ошибок:

$$Q(\vec{x}, \vec{w}) = e^2(\vec{x}, \vec{w}) \quad (3.2.1)$$

Тогда средние потери будут равны:

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

### 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.

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

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

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

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):

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

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:

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

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

При введении сглаживания с помощью модели ЕМА 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 нейронами скрытых слоёв. Эту проблему решает алгоритм обратного распространения ошибок.

Упростим схему индексации нейронов (см. 8):

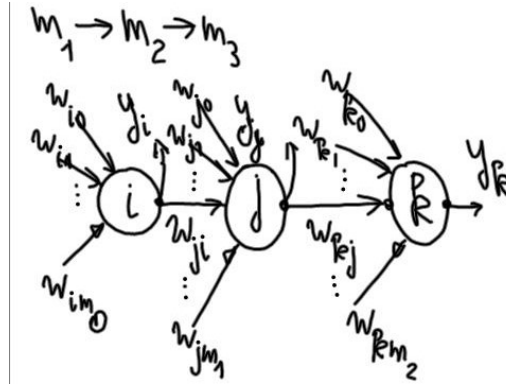


Рис. 8: Новая схема индексации нейронов

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.
2. Связь между нейронами младшего и старшего слоя (синаптический вес) —  $W_{ji}$ ,  $j$  — 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.** 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  $x_m$ , взвешиваемые синаптическим весом  $w_{im_0}$ .

На выходе сумматоров получим:

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

$n$  - номер итерации.

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

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

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

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

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

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

Аналогично на выходе сумматоров нейронов 3-го скрытого слоя:

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

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

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

Зная фактические реакции выходных нейронов, вычислим набор ошибок:

$$e_k(n) = d_k(n) - y_k(n) \quad (3.3.7)$$

Вычисляем величину локального градиента выходных нейронов:

$$\delta_k(n) = \varphi'(v_k(n)) \cdot e_k(n) \quad (3.3.8)$$

Алгоритм коррекции  $k$ -го выходного нейрона:

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

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.

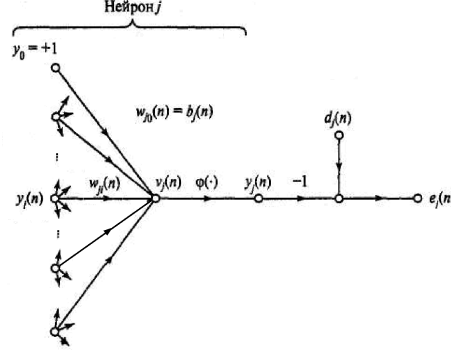


Рис. 9: Граф передачи сигнала в пределах  $j$ -го нейрона

Для нейронов выходного слоя имеем сигналы ошибки  $e_1(n), \dots, e_k(n), \dots, e_{m_3}$ , 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.

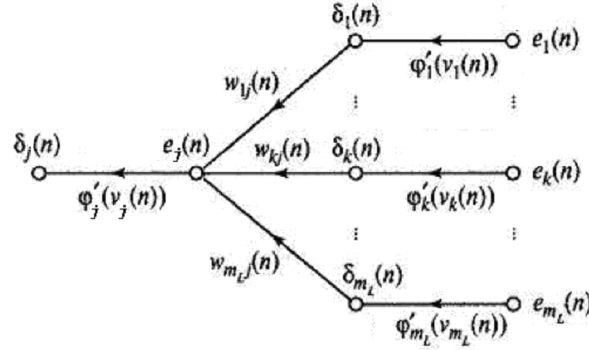


Рис. 10: Направление двух потоков сигналов в многослойном перцептроне

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:

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

*Возможно, суммирование должно быть от 0.*

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

$$\delta_j(n) = e_j(n) \cdot \varphi'(v_j(n)) \quad (3.3.11)$$

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

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



Для нейронов первого скрытого слоя:

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

*Возможно, суммирование должно быть от 0.*

$$\delta_i(n) = e_i(n) \cdot \varphi'(v_i(n)) \quad (3.3.14)$$

Для входного слоя получим:

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

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