Нейронные сети. Лекции.

Лабунец Л.В. & Паймин Антон

Содержание

Лекция 02.04

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.

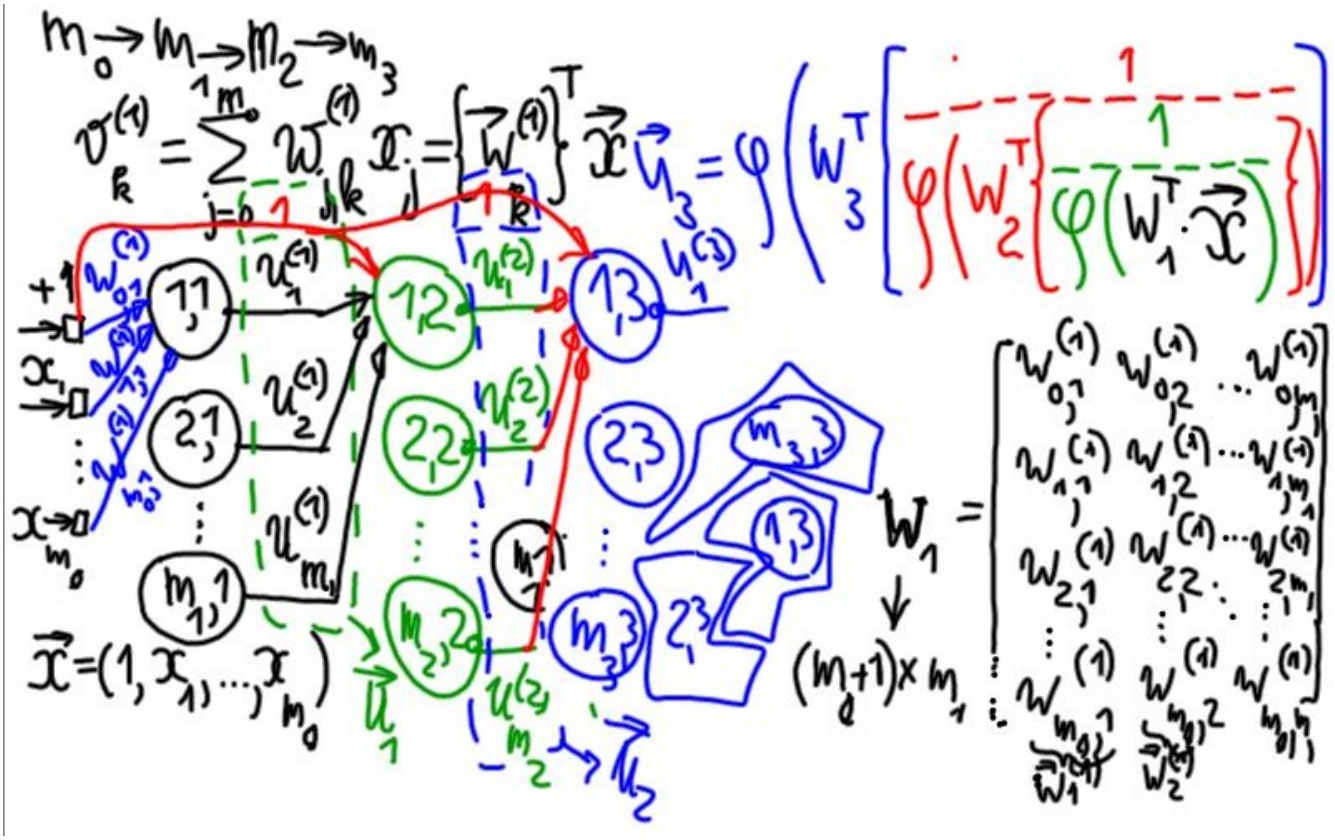

Рис. 1: Структура многослойного перцептрона

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 локального поля*):

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

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

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

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;

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

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, 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.

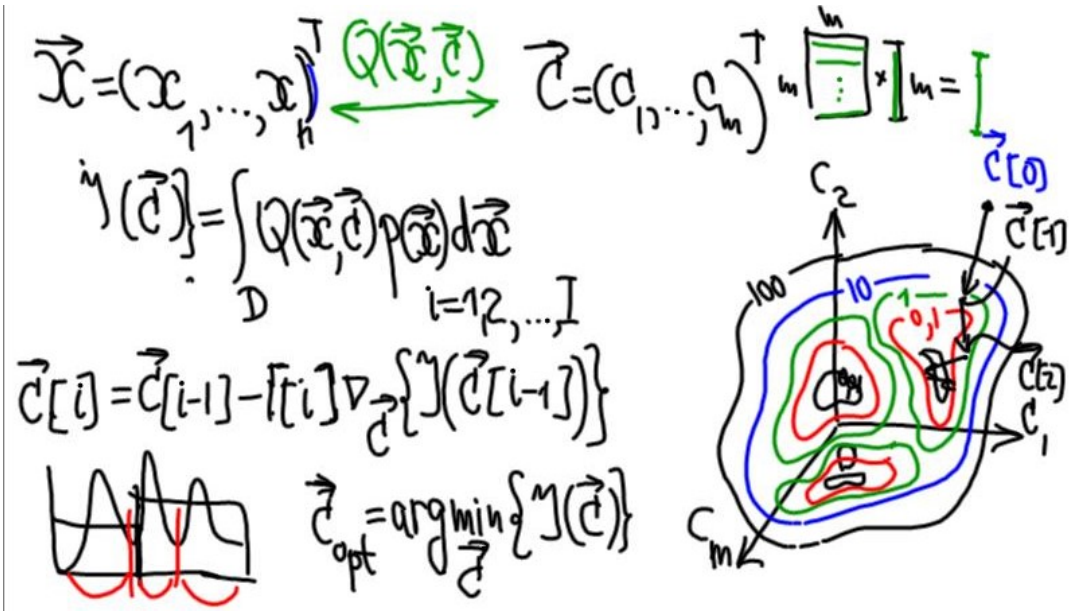

Рис. 2: Обучение нейронной сети

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 которому 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 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:

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

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 следующим выражением:

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

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

Цель обучения:

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

Существуют также методы, не требующие вычисления частной производной, — метод деформируемого многогранника 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.

Лекция 09.04 (дистанционная)

2.1 Алгоритм обучения нейронной сети для случая набора обучающих примеров

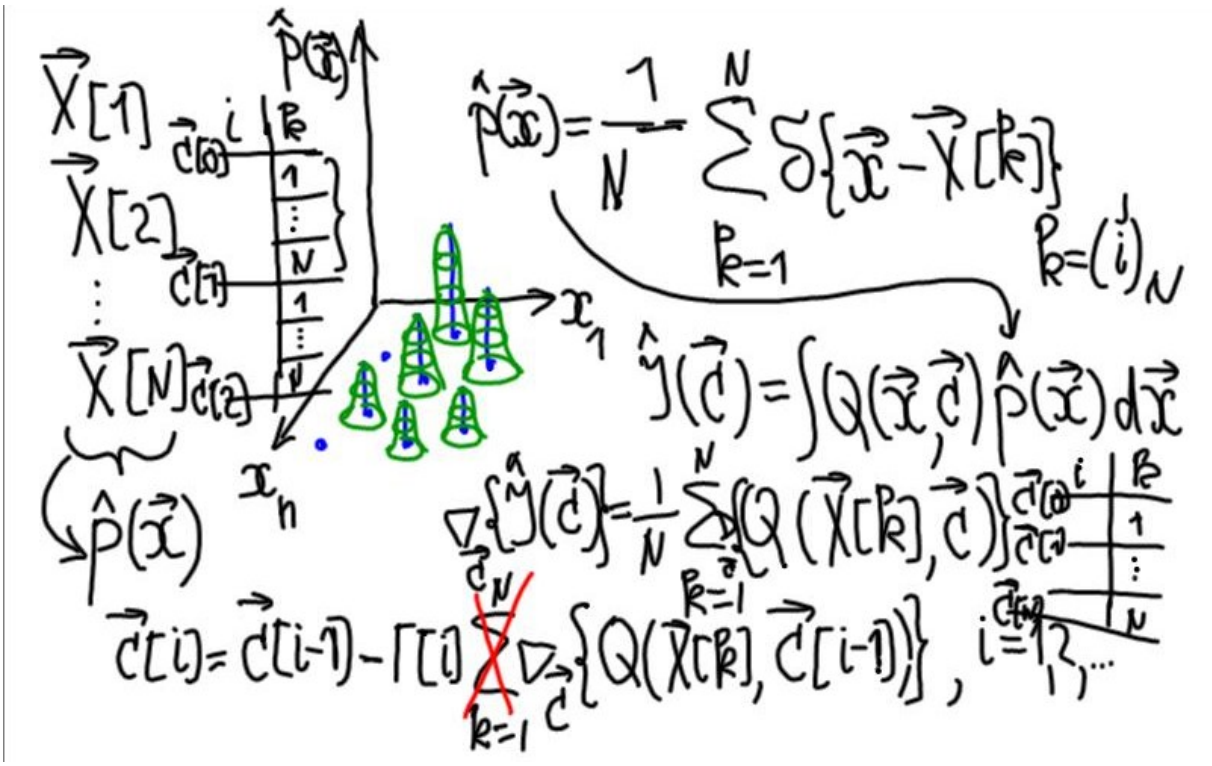

Рис. 3: Системный подход к обучению многослойного перцептрона

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 $\vec{X}$. Объём выборки - N обучающих примеров (см. левый столбец на 3), размерность пространства признаков $\{x_1, \ldots, x_n\}$ — n.

Таким образом в x-пространстве имеется набор обучающих примеров $\vec{X}[1], \ldots, \vec{X}[N]$, 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).

Ссылка на книгу David Scott, Multivariate Density Estimation

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 радиса-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 „иголок“единичного объёма в гиперпространстве. Такие „иголки“не могут иметь общие области и, таким образом, ядерная оценка при таком выборе параметров превращается в „лес иголок“и (что плохо) сглаживание экспериментальных производится не будет — *эмпирическая оценка весовой функции.*

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

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 увеличивать 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 функция 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

2.1.2 Интегрирование весовой функции среднего риска в конечном виде

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:

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

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

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

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:

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

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:

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

2.1.3 Пакетный, последовательный и комбинированный режимы обучения

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

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

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

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:

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

Таблица 1: Описание пакетного режима обучения

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

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

Таблица 2: Описание последовательного режима обучения

Вектор параметров системы	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, шумовою 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 информации, 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.

2.2 Алгоритм обучения перцептронного нейрона

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

Рассмотрим подробнее задачу *дихотомии* (разделение обучающих примеров на 2 класса). Имеется метка $+1$ и метка -1 принадлежности обучающих примеров к 2 альтернативным классам. Рассмотрим синтез алгоритма обучения перцептронного нейрона в рамках задачи дихотомии. Реализуется принцип обучения с поощрением, т.е. для настройки параметров перцептронного нейрона (биас и синаптические веса) применяются обучающие пары (представлены в 3): обучающие примеры $\vec{x}$ на входе перцептронного нейрона и желаемое указание (или метка класса) d, поскольку перцептронный нейрон решает задачу *дихотомии* .

$\vec{x}$ — Обучающие примеры на входе	d — желаемое указание или метка класса
$\vec{X}[1]$	$+1$
$\vec{X}[2]$	-1
$\vdots$	$\vdots$
$\vec{X}[N]$	$+1$

Таблица 3: Обучающие пары

Необходимо в пространстве признаков сформировать модель линейной дискриминантной границы для

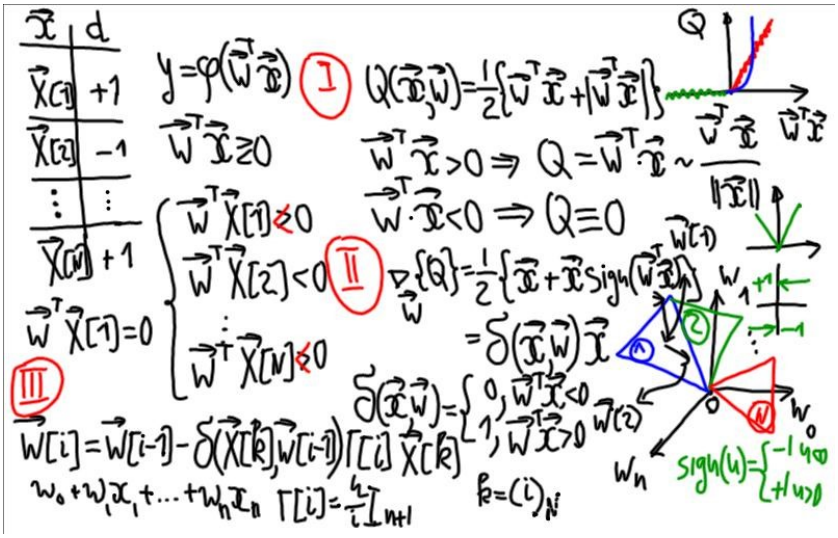

Рис. 4: Алгоритм обучения перцептронного нейрона

того, чтобы реализовать классификацию. Как было сказано выше, существуют 2 случая: линейно разделимых и линейно неразделимых обучающих примеров. Случай линейно разделимых обучающих примеров достаточно редко встречается на практике, поэтому для снижения количества неизбежных ошибочных классификаций для линейно неразделимых векторов образов была придумана *система опорных векторов*.

Рассмотрим сначала простейший случай линейно разделимых векторов образов. Для этого понадобится математическая модель перцептронного нейрона:

$$y = \varphi(\vec{w}^T \cdot \vec{x}), \tag{2.2.1}$$

причём в случае задач дихотомии функция активации φ анализирует знак сигнала на выходе сумматора нейрона (линейной дискриминантной функции) и принимает вид знаковой функции („ступенька“): $\varphi(x) = -1$ для $x < 0$ и $\varphi(x) = -1$ для $x \geqslant 0$. Геометрически представляет собой гиперплоскость.

Таким образом, задачу дихотомии удобно сформулировать в терминах решения системы линейных неравенств. С этой целью для каждой обучающей пары выпишем соответствующее неравенство:

$$\begin{cases} \vec{w}^T \cdot \vec{X}[1] > 0 \\ \vec{w}^T \cdot \vec{X}[2] < 0 \\ \vdots \\ \vec{w}^T \cdot \vec{X}[N] > 0 \end{cases} \tag{2.2.2}$$

Стандартизуем систему линейных неравенств — приведём все неравенства к знаку $<$. Для этого откорректируем все обучающие примеры для тех пар, которые обладают меткой $+1$ путём домножения их на -1. Получим:

$$\begin{cases} \vec{w}^T \cdot \vec{X}[1] < 0 \\ \vec{w}^T \cdot \vec{X}[2] < 0 \\ \vdots \\ \vec{w}^T \cdot \vec{X}[N] < 0 \end{cases} \tag{2.2.3}$$

Воспользуемся 3 этапами системного подхода Я.З. Цыпкина к синтезу алгоритма обучения перцептронного нейрона:

1. Сформулировать цель обучения;
2. Найти оценки вектора градиента функции стоимости;
3. Выписать алгоритм градиентного поиска в C-пространстве параметров системы (состоит из биаса w_0 и синаптических весов w_i).

2.2.1 Формулирование цели обучения

Выберем подходящую модель функции стоимости.

Ссылка на книгу Ту, Гонсалес — Принципы распознавания образов.

В качестве цели обучения выберем среднее арифметическое линейной дискриминантной функции и её модуля:

$$Q(\vec{x}, \vec{w}) = \frac{1}{2}\{\vec{w}^T \cdot \vec{x} + |\vec{w}^T \cdot \vec{x}|\} \tag{2.2.4}$$

Для понимания смысла выбора модели проанализируем 2 случая:

1. Неудачный выбор параметров перцептронного нейрона;
2. Удачный выбор параметров модели.

При неудачном выборе параметров перцептронного нейрона знак неравенства из 2.2.3 становится > 0:

$$\vec{w}^T \cdot \vec{x} > 0 \Rightarrow Q = \vec{w}^T \cdot \vec{x} \sim \frac{\vec{w}^T \cdot \vec{x}}{\|\vec{x}\|} \tag{2.2.5}$$

Т.е., для положительных значений линейной дискриминантной функции потери будут возрастать линейно — неудачный выбор будет „наказан“.

В случае удачного выбора параметров перцептронного нейрона:

$$\vec{w}^T \vec{x} > 0 \Rightarrow Q \equiv 0 \tag{2.2.6}$$

Это позволит реализовать 2 режима: *принцип подкрепления* (при удачном выборе параметров перцептронного нейрона) и *принцип наказания* (при неудачном выборе параметров перцептронного нейрона).

В данном случае важно рассматривать выражения не в x-пространстве, а в w-пространстве: при попадании в область неудачных выборов в w-пространстве, необходимо перейти из неё в область удачных выборов. Чем больше расстояние от текущей точки в w-пространстве от отрицательной области, тем сильнее „наказание“.

До этого „числами“являлись параметры перцептронного нейрона w_i, а „буквами“— обучающие признаки x_i. В w-пространстве всё наоборот, поскольку мы обладаем набором обучающих примеров и w_i — „буквы“(аргументы, решение системы линейных неравенств), а x_i — „числа“.

2.2.2 Получение оценки вектора градиента

Выпишем в явном виде скалярное произведение $\vec{w}^T\vec{x}$:

$$\vec{w}^T\vec{x} = w_0 + w_1 \cdot x_1 + \cdots + w_n \cdot x_n \tag{2.2.7}$$

Рассмотрим частные производные этого выражения по компонентам вектора $\vec{w}$:

$$\begin{aligned}
\frac{\partial\left(\vec{w}^T\vec{x}\right)}{\partial w_0} &= 1 \\
\frac{\partial\left(\vec{w}^T\vec{x}\right)}{\partial w_1} &= x_1 \\
&\vdots \\
\frac{\partial\left(\vec{w}^T\vec{x}\right)}{\partial w_n} &= x_n
\end{aligned}$$

Получим, что градиент произведения $\vec{w}^T\vec{x}$ по параметрам перцептронного нейрона — это обучающий пример $\vec{x}$:

$$\nabla_{\vec{w}}\{\vec{w}^T\vec{x}\} = \vec{x} \tag{2.2.8}$$

При дифференцировании модуля $|x|$ получим:

$$\frac{d|x|}{dx} = \begin{cases} 1, & x > 0 \\ -1, & x < 0 \end{cases} \Rightarrow \frac{d|x|}{dx} = \text{sign}(x) \tag{2.2.9}$$

То есть:

$$\frac{\partial|\vec{w}^T\vec{x}|}{\partial w_i} = x_i \cdot \text{sign}(w_i \cdot x_i) \Rightarrow \nabla_{\vec{w}}\{|\vec{w}^T\vec{x}|\} = \vec{x} \cdot \text{sign} \cdot (\vec{w}^T\vec{x}) \tag{2.2.10}$$

Получим градиент функции стоимости 2.2.4:

$$\nabla_{\vec{w}}\{Q\} = \nabla_{\vec{w}}\left[\frac{1}{2}\{\vec{w}^T \cdot \vec{x} + |\vec{w}^T \cdot \vec{x}|\}\right] = \frac{1}{2}\nabla_{\vec{w}}\left[\vec{w}^T\vec{x}\right] + \nabla_{\vec{w}}\left[|\vec{w}^T \cdot \vec{x}|\right] = \frac{1}{2}\left(\vec{x} + \vec{x} \cdot \text{sign}(\vec{w}^T \cdot \vec{x})\right) \tag{2.2.11}$$

Введём обозначение:

$$\delta(\vec{x}, \vec{w}) = \frac{1}{2}\left(1 + \text{sign}(\vec{w}^T \cdot \vec{x})\right) \tag{2.2.12}$$

Заметим также, что:

$$\delta(\vec{x}, \vec{w}) = \begin{cases} 0, & \vec{w}^T \cdot \vec{x} < 0 \quad \text{— удачный выбор параметров перцептронного нейрона} \\ 1, & \vec{w}^T \cdot \vec{x} > 0 \quad \text{— неудачный выбор параметров} \end{cases} \tag{2.2.13}$$

Содержательный смысл функции $\delta(\vec{x}, \vec{w})$ — счётчик ошибок, накапливающий количество ошибок при неверном выборе параметров перцептронного нейрона.

Окончательно получим:

$$\nabla_{\vec{w}}\{Q\} = \delta(\vec{x}, \vec{w}) \cdot \vec{x} \tag{2.2.14}$$

2.2.3 Алгоритм градиентного поиска

Выпишем для режима последовательного обучения:

$$\vec{w}[i] = \vec{w}[i-1] - \delta(\vec{X}[k], \vec{w}[i-1]) \cdot \Gamma[i] \cdot \vec{X}[k] \tag{2.2.15}$$

Если $\delta(\vec{X}[k], \vec{w}[i-1]) = 0$, то параметры на предыдущей итерации были выбраны удачно и ничего менять не надо — *фаза подкрепления*, т.е. $\vec{w}[i] = \vec{w}[i-1]$, коррекция отсутствует.

Если для вновь поступившего обучающего примера оценка является неудачной, т.е $\delta(\vec{X}[k], \vec{w}[i-1]) = 0$, то необходимо изменить параметры нейрона — *фаза наказания*, осуществляется движение в направлении антиградиента так, чтобы попасть в отрицательную зону.

В w-пространстве граничная гиперплоскость, соответствующая первому неравенству из 2.2.3, делящая пространство на случай удачного и неудачного выбора, проходит через начало координат, поскольку при $\forall w_i = 0$ $\vec{w}^T \cdot \vec{X}[0] = 0$. Поворот гиперплоскости (управляющие косинусы ей нормали) определяются компонентами обучающего примера:

$$\frac{\vec{X}[i]}{\|\vec{X}[i]\|}$$

Вся совокупность граничных гиперплоскостей для всех обучающих примеров проходят через начало координат, но ориентированы различным образом (см. правый нижний угол 4, синий — гиперплоскость 1-го обучающего примера, зелёный — — гиперплоскость 2-го обучающего примера, красный — гиперплоскость N-го обучающего примера).

В случае линейно разделимых векторов образов эти плоскости („нерегулярно“) будут расположены более-менее регулярно, а в идеальном случае — образуют многогранную поверхность в виде разрезанного гиперконуса (гиперпирамиды), которая делит w-пространство на 2 зоны (положительных и отрицательных значений линейной дискриминантной функции). В этом случае существует минимум одно решение системы линейных неравенств 2.2.3.

Отталкиваясь от начального предложения, необходимо сформировать посегментную траекторию градиентного поиска так, чтобы, даже находясь в положительной зоне рано или поздно, попасть в отрицательную зону w-пространства.

В случае линейно неразделимых обучающих примеров „фацеты“будут расположены нерегулярно (хаотически), так, что нельзя будет найти точку в w-пространстве, удовлетворяющей всем неравенствам 2.2.3. Как было сказано выше, эту задачу решает *машина опорных векторов*, обеспечивающая выбор параметров перцептронного нейрона с минимальным количеством нарушений.

В случае процедуры стохастической аппроксимации Роббенса-Монро шаг поиска выбирают, отталкиваясь от начального приближения в направлении антиградиента пропорционально величине η — *параметру скорости обучения*.

$$\Gamma[i] = \frac{\eta}{i} I_{n+1} \tag{2.2.16}$$

где I_{n+1} — единичная матрица размерности $n+1$, поскольку помимо n синаптических весов присутствует биас w_0.

Уменьшение шага поиска пропорциональное номеру шага обучения гарантирует сходимость метода как минимум в случае линейно разделимых векторов образов. Номер текущего обучающего примера k выбирается, как значение целочисленной переменной i по модулю N — остаток от деления нацело номера итерации на количество обучающих примеров:

$$k = (i)_N$$

Лекция 16.04 (Опоздал на полчаса)

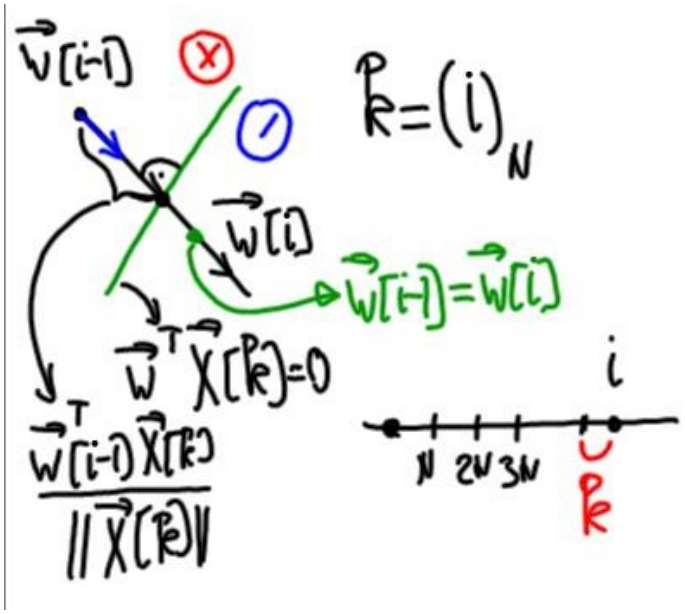

Рис. 5: Аппроксимация многомерной скалярной функции (hyperflat_7_1)

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

3.1 Алгоритм обучения перцептронного нейрона с квадратичной функцией потерь

3.1.1 Формулирование цели обучения

Рассмотрим новую модель функции потерь:

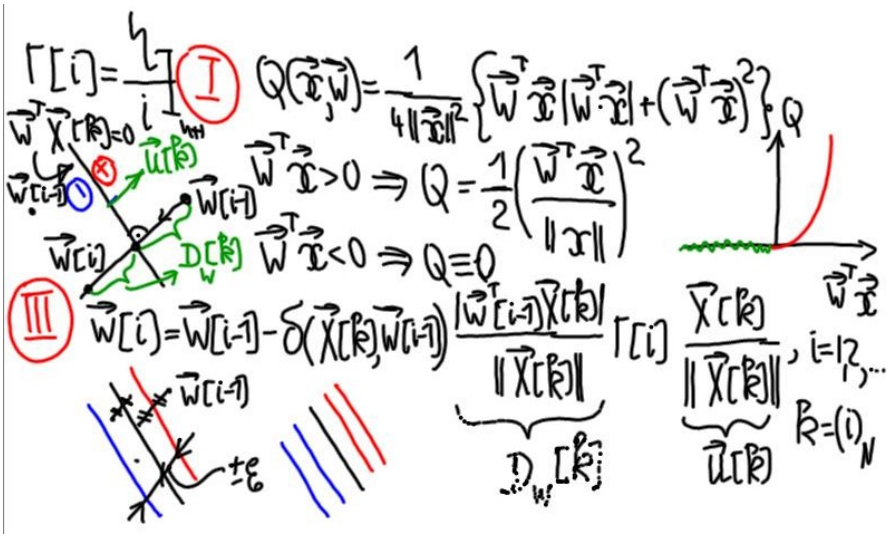

Рис. 6: Алгоритм обучения перцептронного нейрона с квадратичной функцией потерь

$$Q\left(\vec{x},\vec{w}\right)=\frac{1}{4\|\vec{x}\|^2}\left[\vec{w}^T\vec{x}|\vec{w}^T\vec{x}|+\left(\vec{w}^T\vec{x}\right)^2\right] \tag{3.1.1}$$

Для неё получим:

$$Q\left(\vec{x},\vec{w}\right)=\begin{cases}\dfrac{1}{2}\left(\dfrac{\vec{w}^T\vec{x}}{\|\vec{x}\|}\right)^2, & \vec{w}^T\vec{x}>0\\ 0, & \vec{w}^T\vec{x}<0\end{cases} \tag{3.1.2}$$

То есть, при неудачном выборе потери возрастают пропорционально квадрату расстояния текущего значения линейной дискриминантной функции от 0 — функция потерь квадратичная.

3.1.2 Вычисление оценки градиента функции ошибок

Необходимо вычислить вектор градиента квадратичной функции ошибок:

$$\nabla_{\vec{w}}\left[Q\left(\vec{x},\vec{w}\right)\right]=\nabla_{\vec{w}}\left[\frac{1}{4\|\vec{x}\|^2}\left[\vec{w}^T\vec{x}|\vec{w}^T\vec{x}|+\left(\vec{w}^T\vec{x}\right)^2\right]\right]=\frac{1}{4\|\vec{x}\|^2}\cdot\left(\nabla_{\vec{w}}\left[\vec{w}^T\vec{x}|\vec{w}^T\vec{x}|\right]+\nabla_{\vec{w}}\left[\left(\vec{w}^T\vec{x}\right)^2\right]\right) \tag{3.1.3}$$

Надо вычислить градиент функции потерь на листике

Я начал выше, но идёт тяжело

3.1.3 Алгоритм градиентного поиска

Имея текущий обучающий пример x_k и текущие параметры нейрона $\vec{W}[i-1]$, ..

ВЫписать выражение для градиентного поиска после слов

В W-пространстве нормаль $\vec{u}[k]$ вычисляется как:

$$\frac{\vec{X}_k}{\|\vec{X}_k\|} \tag{3.1.4}$$

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

$$D_W[k] = \frac{|\vec{W}^T[i-1]\vec{X}[k]|}{\|\vec{X}[k]\|} \tag{3.1.5}$$

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 "РВТ"

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 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

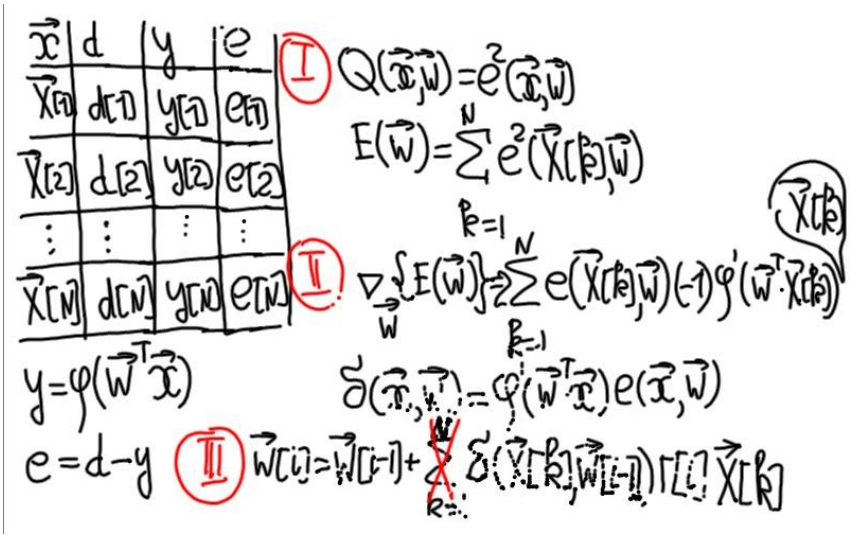

Рис. 7: Аппроксимация многомерной скалярной функции

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.

$\vec{x}$	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 4: 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:

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

Здесь $\delta(\vec{X}[k], \vec{W}[i-1])$ – локальный градиент 3.2.4. 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. 8):

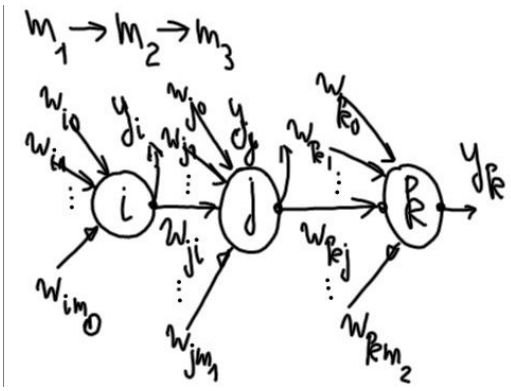

Рис. 8: Новая схема индексации нейронов

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) — 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.** 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 нелинейного 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.

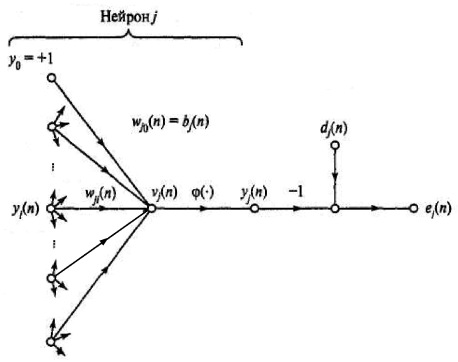

Рис. 9: Граф передачи сигнала в пределах j-го нейрона

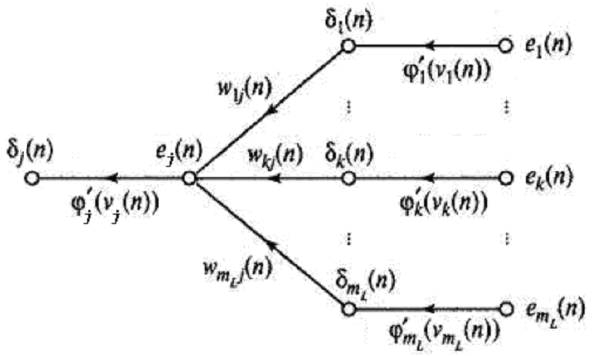

Рис. 10: Направление двух потоков сигналов в многослойном перцептроне

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:

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

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.

Лекция 23.04 (пропустил)

4.1 Практические рекомендации по улучшению сходимости алгоритма обратного распространения ошибок

4.2 Выбор начального приближения для биаса и синаптических весов

Лекция 07.05

5.1 Сеть главных компонент

5.1.1 Метод главных компонент (МГК)

Lec_ Neuro_ 04.docx

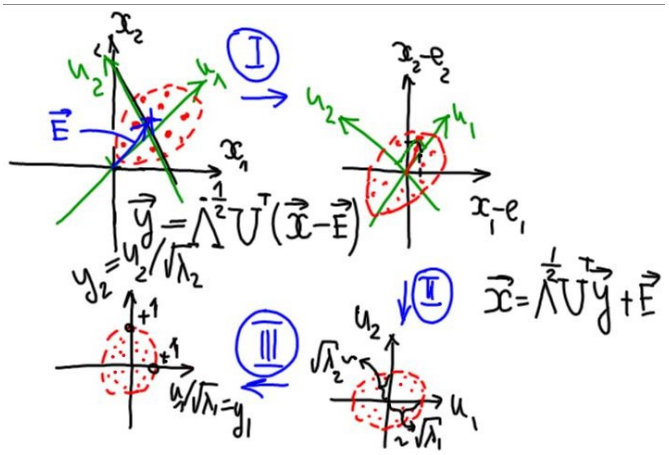

Рис. 11: Метод главных компонент

При решении задач распознавания образов на первом этапе формируется пространство исходных признаков, которое обычно обладает следующими недостатками:

1. Избыточность исходного списка признаков, поскольку он формулируется экспертным (эмпирическим) методом.
2. Значимая корреляция исходных признаков (содержание неравных долей информации об объекте анализа)

Одним из инструментов анализа морфологии такого пространства является анализ диаграммы рассеяния, которая может демонстрировать явные направления, вдоль которых степень свободы (рассеяния) значима, т.е., точки в пространстве образуют эллипсоид. Э является признаком значимой корреляции.

Для анализа пространства применяется ковариационная матрица исходных признаков, в которой прежде всего интересны собственные вектора и собственные значения.

Существует 2 группы собственных значений:

1. **Значимые** ($\gg 0$). Образуют *сигнальное подпространство*, размерность (количество степеней свободы) которого определяется рангом M ковариационной матрицы обучающих примеров. Также определяют энергию сигнального подпространства, поскольку собственные значения являются аналогом дисперсии, и количество степеней свободы.
2. **Малозначимые** (≈ 0). Образуют *шумовое подпространство*, порождённое ошибками измерений и вычислений.

Согласно МГК собственные вектора матрицы ковариации исходных признаков определяют новый базис (как было показано Пирсоном), а ошибка описания исходных признаков с помощью МГК определяется суммой малозначимых собственных значений.

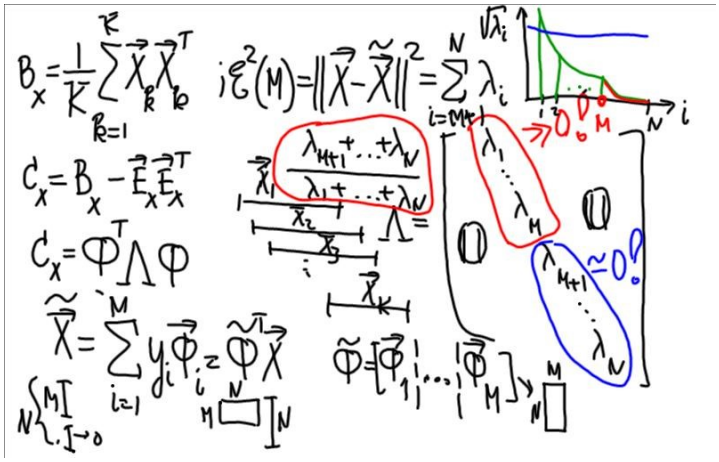

Рис. 12: Метод главных компонент (продолжение)

Относительная ошибка определяется как:

$$\frac{\lambda_{M+1} + \cdots + \lambda_{M+1}}{\lambda_1 + \cdots + \lambda_M} \tag{5.1.1}$$

То есть, задавшись допустимым значением ошибок описания, можно получить критерий отбора собственных значений.

5.1.2 Сеть главных компонент (введение)

Сеть главных компонент в процессе обучения по набору непомеченных обучающих примеров должна выдать направляющие косинусы осей сигнального подпространства (компоненты собственных векторов), а совокупность фактических реакций нейронов сети главных компонент дать оценки дисперсий (оценки собственных значений), что позволит ортогонально спроецировать исходные данные на оси сигнального подпространства. Задача упрощается в случае, если корреляция значима, поскольку тогда эффективный ранг будет на порядок меньше ранга пространства исходных признаков.

Аналогично МГК функционирует и в задачах прогнозирования.

В основе сети главных компонент лежат выборочные оценки направляющих косинусов собственных векторов сигнального подпространства и оценки собственных значений, определяющихся ковариационной матрицей (второй центральный момент). В литературе ковариация и корреляция в русской и иностранной литературе альтернативны (ковариация — correlation). Основой выборочного оценивания является корреляционное произведение.

5.1.3 Гипотеза Хебба (принцип самоусиления (корреляции))

Правило коррекции синаптических весов, устанавливающее связь между входным сигналом и его реакцией в виде корреляционного произведения в простейшей форме:

$$\Delta w_{kj}(n) = \eta \cdot y_k(n) \cdot x_j(n) \tag{5.1.2}$$

иначе называемое *правилом обучения синапса Хебба*. Здесь x_j — сигнал на входе линейного нейрона от j-го исходного признака, y_k — сигнал на выходе k-го нейрона, $\eta \in [0, 1]$ — параметр скорости обучения.

Зависимость корректирующего значения $\Delta w_{kj}(n)$ от выходного сигнала y_k представлена на 13. Угловой коэффициент для гипотезы Хебба равен $\eta \cdot x_j$. Видим, что правило Хебба не вводит ограничение на значение синаптических весов, которые могут возрастать бесконечно, что приводит к состоянию *насыщения сети* (состояние, при которой выходные значения ИНС нечувствительны к изменению исходных признаков).

5.1.4 Принцип ковариации

Для стабилизации весов необходимо ввести обратную связь и вместо *принципа самоусиления (корреляции)* ввести принцип конкурентности — *принцип ковариации*.

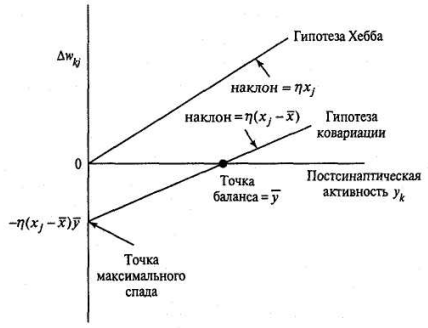

Рис. 13: Иллюстрация принципов Хебба и ковариации

Для этого правило Хебба модифицируют центрированием обучающих примеров — происходит параллельный перенос системы координат исходных признаков в центр группировки обучающих примеров (см. прошлое занятие). Тогда получаем:

$$\Delta w_{kj} = \eta\,(x_j - \bar{x})\,(y_k - \bar{y}) \tag{5.1.3}$$

где $\bar{x}$ и $\bar{y}$ — усредненные по времени значения предсинаптического x_j и постсинаптического y_k сигналов.

Теперь возможны и отрицательные корректирующие значения и существует *точка баланса*, что видно для соответствующей прямой на 13.

5.2 Метод настройки весов для сети главных компонент

Линейный нейрон, являющийся структурных элементом сети главных компонент, называют *фильтром Хебба*. Несмотря на простоту, в синаптических весах фильтра Хебба сохраняются направляющие косинусы осей в пространстве главных компонент.

Lec_ Neuro_ 14.docx

5.2.1 Правила обучения линейного нейрона

Как было отмечено выше, синаптические веса линейного нейрона по сути своей являются направляющими косинусами оси в пространстве исходных признаков.

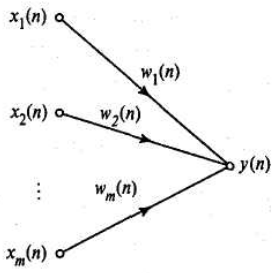

Рис. 14: Граф линейного нейрона

Рассмотрим модель линейного нейрона (представлена на 14):

$$y = \sum_{i=1}^{m} w_i x_i \tag{5.2.1}$$

где w_i — синаптические веса (всего m штук), n — итерация обучения.

Важно! Здесь отсутствует сигнал стандартного смещения $+1$ в отличие от перцептронного нейрона.

В рамках гипотезы корреляции (согласно правилу Хебба):

$$w_i(n) = w_i(n-1) + \eta \cdot y(n-1) \cdot x_i(n-1), \quad i = 1, 2, \ldots, m \tag{5.2.2}$$

где n — дискретное время, η — параметр скорости обучения.

Здесь реализован только *принцип самоусиления*, что нас не устраивает. Необходимо ввести *эффект конкуренции* между синаптическими весами: между собой они могут соревноваться, но при этом длина вектора синаптических весов всегда должна быть равна 1, т.е. геометрически синаптические веса приобретают смысл направляющих косинусов.

Для этого после каждой итерации будем проводить нормировку, если совокупность весов не образует вектор единичной длины:

$$w_i(n) = \frac{w_i(n-1) + \eta \cdot y(n-1) \cdot x_i(n-1)}{\sqrt{\sum_{i=1}^{m} \left[w_i(n-1) + \eta \cdot y(n-1) \cdot x_i(n-1)\right]^2}} \tag{5.2.3}$$

В результате нормировки правило обучения становится нелинейным. Поэтому применим методы линеаризации: аппроксимируем нелинейную зависимость рядом (Маклорена или Тейлора), степень аргумента которого будет убывать достаточно быстро. В качестве аргумента используем применим параметр скорости обучения $\eta \in [0, 1]$.

Необходимо вычислить набор частных производных по η и их значения при $\eta = 0$, чтобы получить полином, у которого отбросим степени, старше первой (для ряда Маклорена). Итого получим:

$$w_i(n) = w_i(n-1) + \eta \cdot y(n-1) \left[x_i(n-1) - y(n) \cdot w_i(n-1)\right] \tag{5.2.4}$$

Это полученное линеаризованное правило обучения синаптических весов линейных нейронов, реализующее оба принципа (самоусиления и конкуренции), иногда называют *правилом Ойа*.

Данное правило обучения в итоге выдаёт наибольшее собственное значение ковариационной матрицы обучающих примеров, а веса являются направляющими косинусами. Для понимания этого реализуем матричную формулировку алгоритма.

5.2.2 Матричная формулировка алгоритма Ойа

Для входного вектора исходных признаков $\vec{x}(n)$ и вектора синаптических весов $\vec{w}(n)$ введём следующие обозначения:

$$\vec{x}(n) = \left[x_1(n), x_2(n), \ldots, x_m(n)\right]^T \tag{5.2.5}$$

$$\vec{w}(n) = \left[w_1(n), w_2(n), \ldots, w_m(n)\right]^T \tag{5.2.6}$$

Тогда выходная реакция линейного нейрона и правило его обучения:

$$y(n) = \vec{x}^T(n) \cdot \vec{w}(n) = \vec{w}^T(n) \cdot \vec{x}(n) \tag{5.2.7}$$

$$\vec{w}(n) = \vec{w}(n-1) + \eta \cdot y(n-1) \left[\vec{x}(n-1) - y(n) \cdot \vec{w}(n-1)\right] \tag{5.2.8}$$

Подставим в правило Ойя 5.2.8 фактическую реакцию нейрона (различные представления для двух вхождений в формулу 5.2.7) и получим:

$$\vec{w}(n) = \vec{w}(n-1) + \eta \cdot y(n-1) \left[\vec{x}(n-1) \cdot \vec{x}^T(n-1) \cdot \vec{w}(n-1) - \vec{w}^T(n-1) \cdot \vec{x}(n-1) \cdot \vec{x}^T \cdot \vec{w}(n-1) \cdot \vec{w}(n-1)\right] \tag{5.2.9}$$

Здесь $\vec{x}(n) \cdot \vec{x}^T(n)$ — *элементарная корреляционная матрица*, квадратная матрица размерности $m \times m$.

Введём замены:

$$\vec{h}(\vec{w}(n), \vec{x}(n)) = \left[\vec{x}(n) \cdot \vec{x}^T(n) \cdot \vec{w}(n) - \vec{w}^T(n) \cdot \vec{x}(n) \cdot \vec{x}^T \cdot \vec{w}(n) \cdot \vec{w}(n)\right] \tag{5.2.10}$$

$$\vec{\eta}(n) = \eta \cdot y(n) \tag{5.2.11}$$

Получим:

$$\vec{w}(n) = \vec{w}(n-1) + \vec{\eta}(n-1) \cdot \vec{h}(\vec{w}(n-1), \vec{x}(n-1)), \quad n = 0, 1, 2, \ldots \tag{5.2.12}$$

Это нелинейное стохастическое разностное уравнение.

По мере обучения объёма обучающих примеров переходим к непрерывному времени и разностное уравнение превращается в систему линейных ДУ:

$$\frac{d}{dt}\vec{w}(t) = \langle\vec{h}(\vec{w}(t))\rangle \tag{5.2.13}$$

При этом используем обычную выборочную среднюю по матрице набора обучающих примеров $\vec{X}$ (по всем реализациям $\vec{x}$):

$$\langle\vec{h}(\vec{w})\rangle = \lim_{n\to\infty} E\left[h(\vec{w},\vec{X})\right] \tag{5.2.14}$$

Усредняя функцию коррекции $\vec{h}(\vec{w},\vec{x})$ по столбцам матрицы обучающих примеров:

$$\vec{h}(\vec{w},\vec{x}) = \vec{x}(n)\cdot y(n) - y^2(n)\cdot\vec{w}(n) = \vec{x}(n)\cdot\vec{x}^T(n)\cdot\vec{w}(n) - \left[\vec{w}^T(n)\cdot\vec{x}(n)\cdot\vec{x}^T(n)\cdot\vec{w}(n)\right]\vec{w}(n) \tag{5.2.15}$$

В квадратных скобках 5.2.15 — квадратичная форма (по определению положительна).

Усредняя 5.2.15 по множеству набора обучающих примеров, получим:

Усредняя функцию коррекции $\vec{h}(\vec{w},\vec{x})$ по столбцам матрицы обучающих примеров:

$$\langle\vec{h}\rangle = \lim_{n\to\infty} E\left[\vec{X}(n)\cdot\vec{X}^T(n)\cdot\vec{w}(n) - \left(\vec{w}^T(n)\cdot\vec{X}(n)\cdot\vec{X}^T(n)\cdot\vec{w}(n)\right)\vec{w}(n)\right] = \mathbf{R}\cdot\vec{w}(\infty) - \left[\vec{w}^T(\infty)\cdot\mathbf{R}\cdot\vec{w}(\infty)\right]\vec{w}(\infty) \tag{5.2.16}$$

где $\mathbf{R}$ — матрица корреляции исходных данных, $\vec{w}(\infty)$ — предельное значение вектора синаптических весов

Итого получим систему ЛДУ:

$$\frac{d}{dt}\vec{w}(t) = \langle\vec{h}(\vec{w}(t))\rangle = \mathbf{R}\cdot\vec{w}(t) - \left[\vec{w}^T(t)\cdot\mathbf{R}\cdot\vec{w}(t)\right]\vec{w}(\infty) \tag{5.2.17}$$

решение которой известно:

$$\lim_{n\to\infty}\vec{w}(n) = \vec{q}_1 \tag{5.2.18}$$

где $\vec{q}_1$ — оценка первого собственного вектора. Считаем свойство фильтра Хебба доказанным.

Для существования решения накладываются ограничения на выбор параметра скорости обучения η, который от итерации к итерации должен изменяться, чтобы выполнялись условия (*стохастическая аппроксимация Роббенса-Монро*):

$$\sum_{n=1}^{\infty}\eta(n) = \infty \tag{5.2.19}$$

$$\sum_{n=1}^{\infty}\eta^p(n) < \infty, \quad p > 1 \tag{5.2.20}$$

$$\eta(n) \to 0 \text{ при } n\to\infty \tag{5.2.21}$$

Обучение линейного нерона по правилу Ойа гарантирует в конце обучения следующие результаты:

Свойства фильтра Хебба

1. Вектор синаптических весов $\vec{w}(n)$ является оценкой первого собственного вектора $\vec{q}_1$ корреляционной матрицы $\mathbf{R}$ входных данных. СИнаптические веса также являются направляющими косинусами.

2. Дисперсия совокупности фактических реакций линейного нейрона, обученного по правилу Ойя, даёт выборочную оценку дисперсии ковариационной матрицы обучающих примеров — её наибольшее собственное значение λ_1.

Лекция 14.05 (не проводилась)

Лекция 21.05

7.1 Архитектура сети главных компонент — обобщённый алгоритм Хебба

Фильтр Хебба на основе одного линейного нейрона оценивает первую главную компоненту входного сигнала. Сеть прямого распространения с одним слоем линейных нейронов способна оценивать все главные компоненты входного сигнала произвольной размерности.

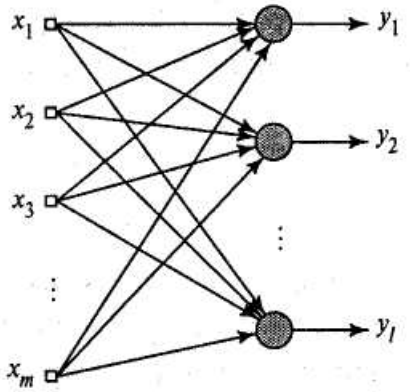

Рис. 15: Архитектура сети главных компонент

Lec_ Neuro_ 14.docx

Сеть главных компонент состоит из m входных нейронов, обозначенных x_i. Имеется единственный слой из l нейронов y_j, каждый из которых обучается правилом Ойа. В синаптических весах каждого нейрона сохраняются направляющие косинусы соответствующего собственного вектора ковариационной матрицы. Матрица этих весов формирует базис размерностью $l < m$, повёрнутый относительно базиса входных признаков.

Размерность сигнального подпространства (пространства информативных признаков) определяется рангом ковариационной матрицы данных (*в электронной версии это* l). МГК позволяет обнаружить наличие сильной корреляции в исходных признаках и, в силу этого, размерность сигнального подпространства значительно меньше пространства входных признаков ($l \ll m$) Каждый нейрон позволяет получить оценку дисперсии (собственные значения ковариационной матрицы данных) — масштаба соответствующей оси.

В основе алгоритма обучения сети главных компонент лежит *обобщенный алгоритм обучения Хебба* (Generalized Hebbian Algorithm, GHA). Алгоритм устроен следующим образом:

$$\Delta w_{ji}(n) = \eta \left[y_j(n)x_i(n) - y_j(n)\sum_{k=1}^{j} w_{ki}(n)y_k(n) \right], \quad i = 1, \ldots, m, \quad j = 1, \ldots, l \tag{7.1.1}$$

где $\Delta w_{ji}(n)$ устанавливает связь между x_i и y_j и является *величиной коррекции* пред- и постсинаптических сигналов x_i и y_j соответственно.

7.1.1 Последовательное обучение

Для упрощения анализа на первом этапе примем предположим, что реализуется последовательное во времени обучение (настройка параметров) линейных нейронов: сначала обучается 1-й линейный нейрон, за ним 2-й линейный нейрон и т.д.

Рассмотрим обучение 1-го линейного нейрона. В этом случае $j = 1$, а поскольку остальные нейроны не используются, сумма в 7.1.1 вырождается в $\sum_{k=1}^{1}$ — алгоритм вырождается в правило Ойа, реализующий 1-й и 2-й принципы самоорганизации (самоусиления [гипотеза корреляции, 1-я компонента] и конкуренции).

Результатом является формирование в пространстве признаков оси, единичный орт которой указывает направление первого главного собственного вектора ковариационной матрицы обучающего примера. При проекции этой оси на ось сигнального подпространства получим порцию полезной информации, которую даёт j-я ось сигнального подпространства. Вдоль оси с $j = 1$ степень рассеяния обучающих примеров наибольшая.

Далее из вектора обучающего примера $\vec{x}$ вычитаем фактическую реакцию, масштабированную вектором синаптических весов — проекцию его на ось сигнального подпространства:

$$\vec{x}'(n) = \vec{x}(n) - \vec{w}(n)y_1(n) \tag{7.1.2}$$

По сути, происходит "обеление"обучающего примера, из которого "удаляется"первая главная (информативная) компонента. Таким образом, второй нейрон ($j = 2$) обрабатывает уже модифицированный вектор входных компонентов $\vec{x}'(n)$. Разобьём сумму в 7.1.1 на 2 компоненты:

$$\Delta w_{2i}(n) = \eta \cdot y_2(n) \left[x_i(n) - y_1(n)w_{1i}(n) - w_{2i}(n)\right] \tag{7.1.3}$$

Введём новое обозначение:

$$x'_{2i} = x_i(n) - w_{1i}(n)y_1(n) \tag{7.1.4}$$

Тогда 7.1.1 для $j = 2$ с учётом 7.1.4 становится аналогичным правилу Ойа:

$$\Delta w_{2i}(n) = \eta \cdot y_2(n) \left[x'_{2i}(n) - w_{2i}(n)y_2(n)\right] \tag{7.1.5}$$

Удаление первой информативной компоненты 7.1.2 приводит 7.1.1 к виду, аналогичному правилу обучения Ойа.

Повторив аналогично действия, удалим вторую главную компоненту из набора обучающих примеров для перехода к обучению 3-го нейрона, т.е. в гиперплоскости, образованной 1-м и 2-м собственным векторами, имея масштабированные вектора информативных признаков, их сумму удаляем из вектора обучающих примеров.

$$\vec{x}'(n) = \vec{x}(n) - \vec{w}(n)y_1(n) - \vec{w}(n)y_1(n) \tag{7.1.6}$$

Аналогичные действия повторяем и для третьего нейрона, что позволяет перейти к обучению 4-го линейного нейрона. Т.е., в общем случае:

$$\Delta w_{ji}(n) = \eta \cdot y_j(n) \left[x'_{ji}(n) - w_{ji}(n)y_j(n)\right], \quad i = 1, \ldots, m, \quad j = 1, \ldots, l \tag{7.1.7}$$

$$x'_{ji}(n) = x_i(n) - \sum_{k=1}^{j-1} w_{ki}(n)y_k(n) \tag{7.1.8}$$

Итого получим, что в исходном пространстве сформирован базис сигнального подпространства

$$\vec{w}_1, \ldots, \vec{w}_l,$$

на который ортогонально спроецирован вектор обучающего примера, т.е. Реализована *пространственная декомпозиция*. Остаток (значение вектора обучающего примера после вычета всех главных компонент) — *ошибка реконструкции данных*, — вектор, который целиком содержится в шумовом подпространстве.

Анализ показывает, что такого рода процедура представляет собой декомпозицию исходных признаков на информативные признаки в виде произведений собственных чисел ковариационной матрицы (совпадают с y_i) на собственные вектора ковариационной матрицы $\vec{q}_i$.

Для получения оценки дисперсии ошибки реконструкции из каждого обучающего примера необходимо вычесть собственные вектора и просуммировать длины полученных векторов-остатков, либо получить относительное значение, пользуясь значениями собственных чисел ковариационной матрицы:

$$\frac{\sum_{k=l+1}^{m} \lambda_k}{\sum_{k=1}^{m} \lambda_k} \tag{7.1.9}$$

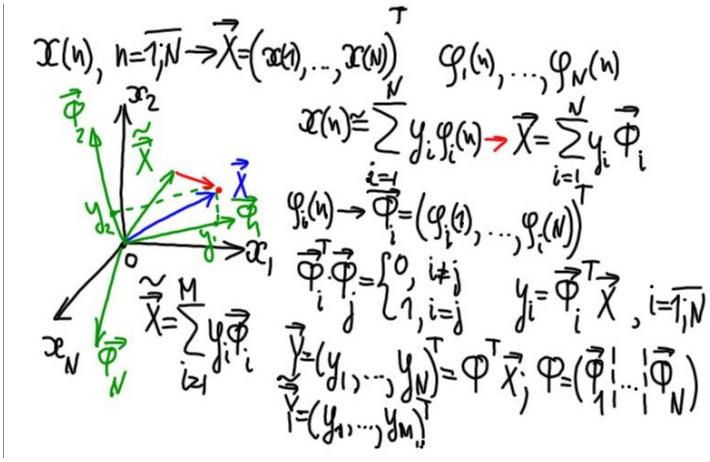

Рис. 16: hyperflat 12 1

7.1.2 Параллельное обучение

Рассмотрим новые объекты анализа.

Сформируем матрицу единичных векторов синаптических весов сети главных компонент размерности $l \times m$:

$$\mathbf{W}(n) = [\vec{w}_1(n), \vec{w}_2(n), \ldots, \vec{w}_l(n)]^T \tag{7.1.10}$$

Заметим, что:

$$\mathbf{W}^T(n) = \{w_{ji}(n)\}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, l$$

Вводится величина коррекции (*обобщенный алгоритм Хебба в матричном виде*):

$$(8.91)\Delta\mathbf{W}(n) = \eta(n)\left\{\vec{y}(n) \cdot \vec{x}^T(n) - LT\left[\vec{y}(n) \cdot \vec{y}^T(n)\right] \cdot \mathbf{W}(n)\right\} \tag{7.1.11}$$

$\vec{y}(n)$ – вектор-столбец с l компонент, $\vec{x}(n)$ – вектор-столбец с m компонент, $LT\left\{\vec{y}(n) \cdot \vec{y}^T(n)\right\}$ — оператор, устанавливающий все элементы матрицы $\vec{y}(n) \cdot \vec{y}^T(n)$, расположенные выше ее главной диагонали, в нуль (формирует нижнюю треугольную матрицу). Нижняя треугольная матрица необходима для реализации GHA. По мере обучения матрица и вектора обновляются.

Не уверен, но η — матрица

Если параметр скорости обучения $\eta(n)$ удовлетворяет условиям:

$$\lim_{n\to\infty} \eta(n) = 0, \quad \sum_{n=0}^{\infty} \eta(n) = \infty$$

то обобщенный алгоритм Хебба будет сходиться к фиксированной точке, а $\mathbf{W}(n)$ сходится к матрице, строки которой являются первыми l собственными векторами матрицы ковариации размерности $m \times m$ входных векторов размерности $m \times 1$, упорядоченных по убыванию собственных значений.

Лекция 28.05 (показаны новые наработки в 514м)