

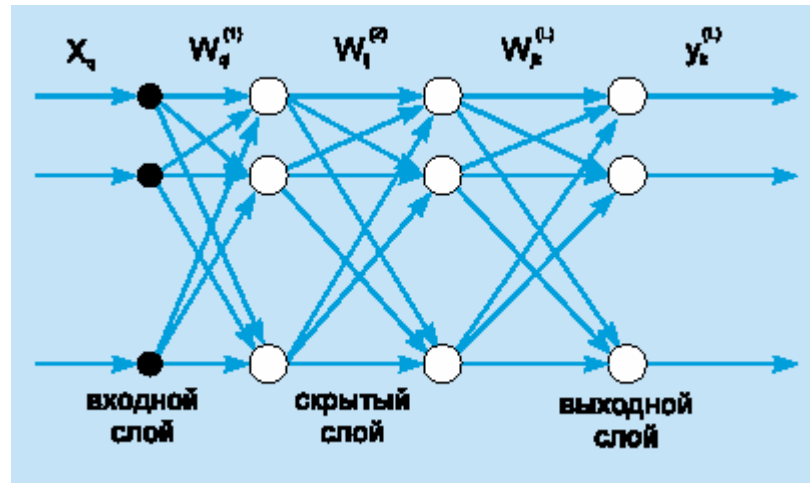
Машинное обучение

Лекция 2.

Часть 1.

Многослойные персептроны
(MLP) и обучение обратным распространением
ошибки

Персептроны



Все решаемые задачи сводятся к двум:

- **Классификация** (дискретный набор выходных значений)
- **Регрессия** (непрерывные выходные значения)

Regression & Classification – Common Algorithms



Regression

Simple Linear Regression

Multiple Linear Regression

Polynomial Linear Regression

Support Vector Regression

Decision Tree Regression

Random Forest Regression

Neural Network

Classification

Logistic Regression

K Nearest Neighbors

Support Vector Machine

Naïve Bayes

Decision Tree Classification

Random Forest Classification

Neural Network

Regression Vs Classification - Differences



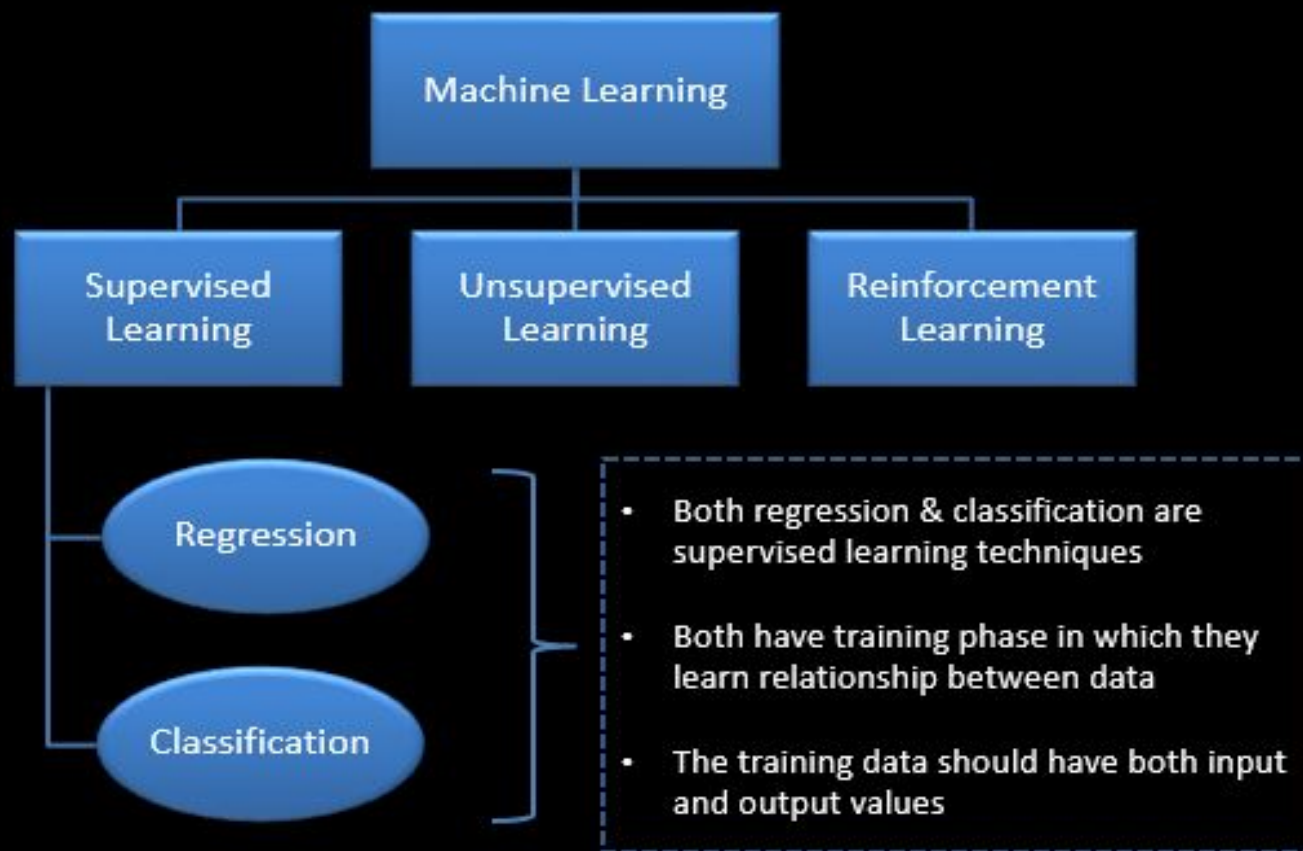
Regression

Age	Income	Loan Amount
21	20000	0
37	55000	150000
29	35000	120000
23	17000	550000
34	70000	250000
47	84000	0
25	30000	90000

Classification

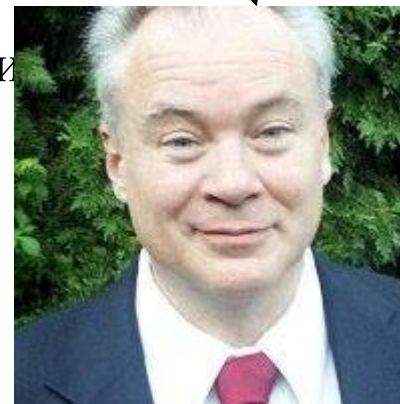
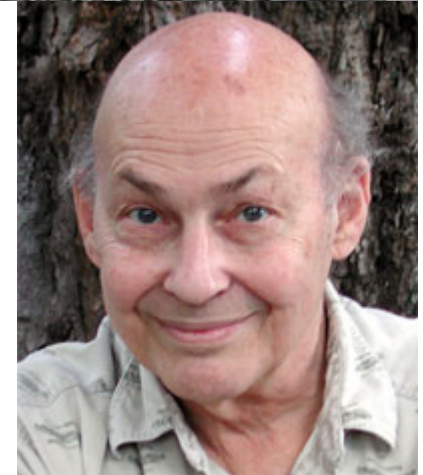
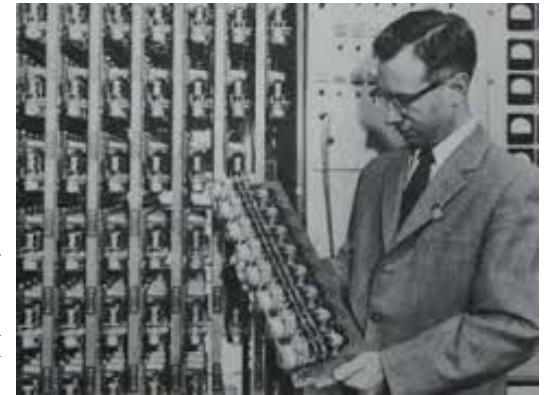
Age	Income	Loan Status
21	20000	Rejected
37	55000	Approved
29	35000	Approved
23	17000	Rejected
34	70000	Approved
47	84000	Rejected
25	30000	Approved

Regression Vs Classification - Similarities



История персептронов

- Персептроны Розенблатта (аппаратная реализация) в 1950-х годах
 - Ф.Розенблатт. Принципы нейродинамики. Персептроны и теория механизмов мозга. – М.: Мир, 1965.
- Их критика в
 - М.Минский, С.Пейперт. Персептроны. – М.: Мир, 1971.
- Обучение обратным распространением ошибки
 - *Werbos P. J.*, Beyond regression: New tools for prediction and analysis in the behavioral sciences. Ph.D. thesis, Harvard University, Cambridge, MA, 1974.
 - *Rumelhart D.E., Hinton G.E., Williams R.J.*, Learning Internal Representations by Error Propagation. In: Parallel Distributed Processing, vol. 1, pp. 318—362. Cambridge, MA, MIT Press. 1986.
- Обучение обратным распространением ошибки neural networks (autoencoder) –
 - середина 2000-х годов



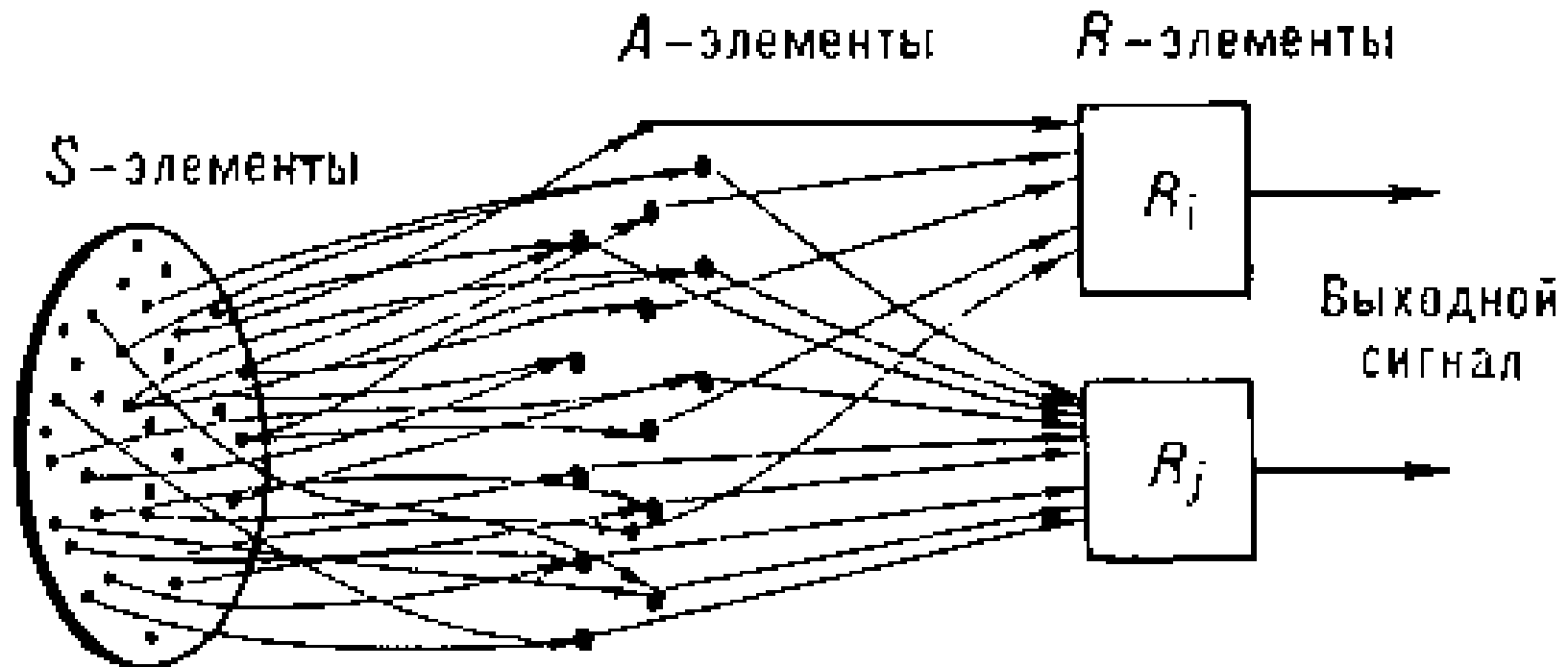
История персептронов в СССР

- А.И.Галушкин. Синтез многослойных систем распознавания образов. – «Энергия», Москва, 1974. (предложил градиентный метод обучения)
- Другие советские исследователи НС:

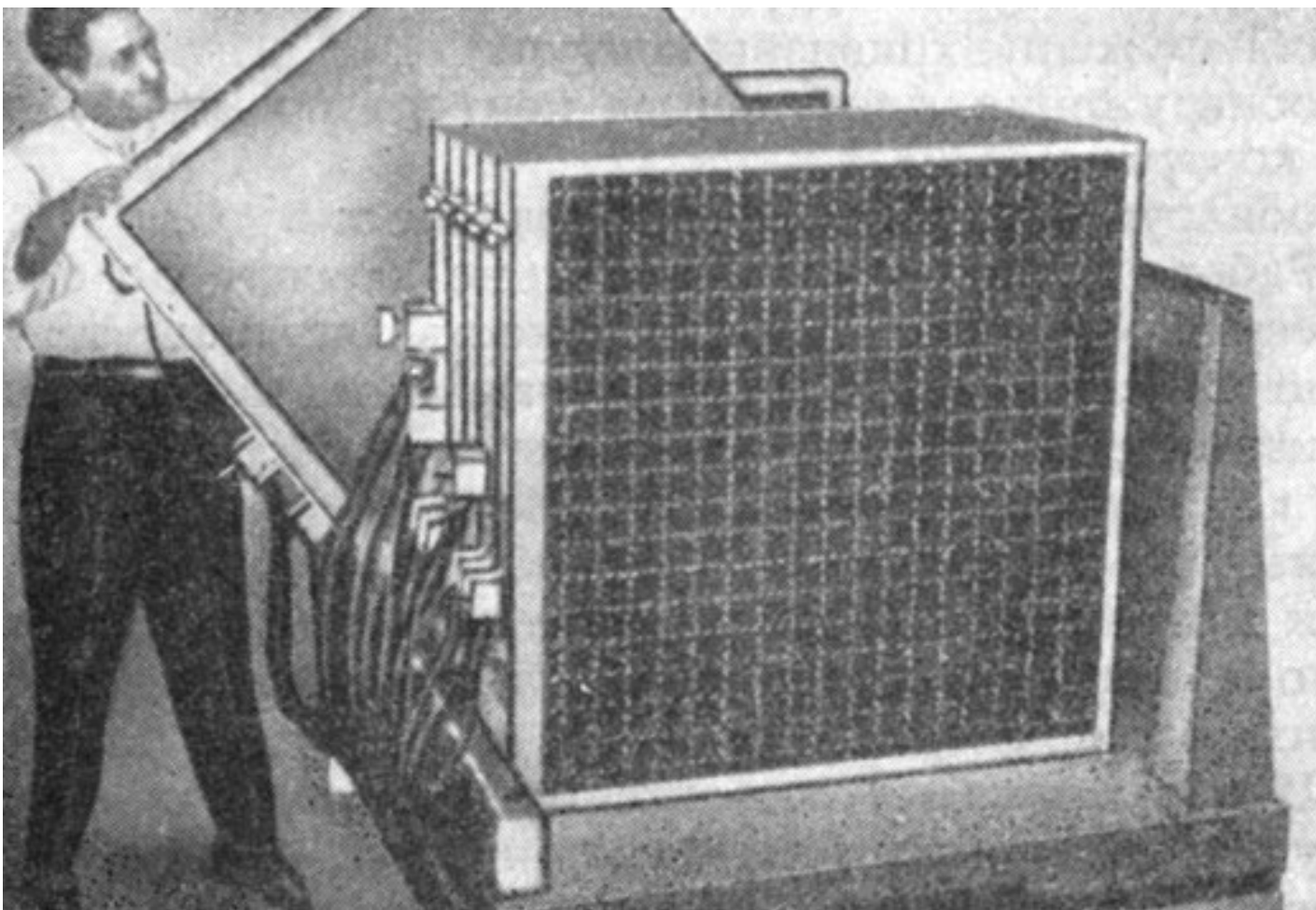
А.Г. Ивахненко, М. М. Бонгард, М. Л. Цетлин, М.А. Айзерман, Э.М. Браверман, Л.И. Розоноэр, В.И. Варшавский, В.С. Гурфинкель, В.Н. Вапник, А.Я. Червоненкис, А. И. Галушкин,...



Персептрон Розенблатта

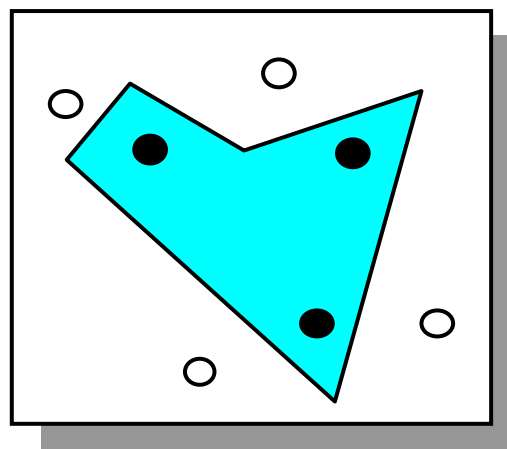
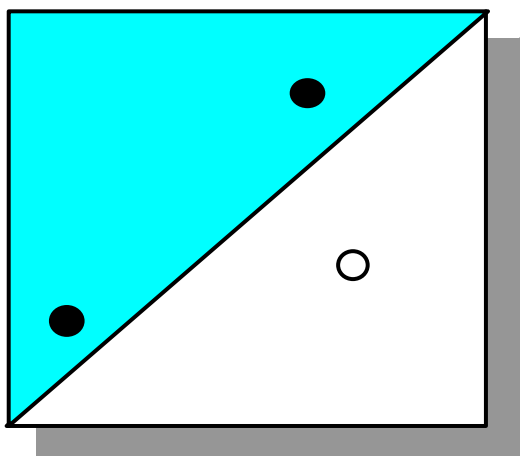
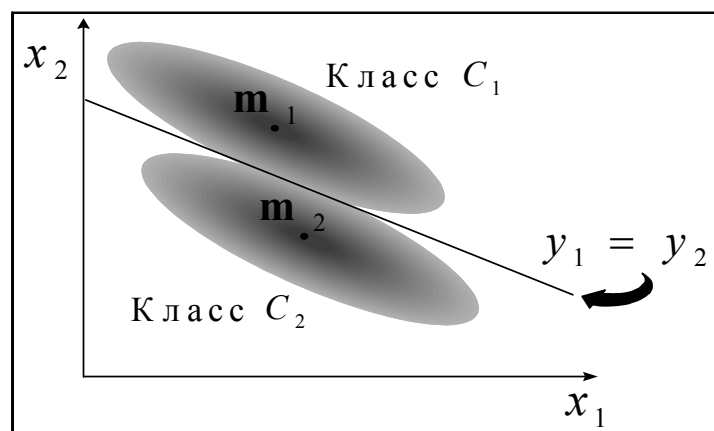


Электронная модель глаза лягушки (1960-е годы)



Гаврилов А.В.
НГТУ, кафедра ВТ

Классификация



Классификация

- Сеть с одним скрытым слоем, содержащим H нейронов со *ступенчатой* функцией активации, способна осуществить произвольную классификацию Hd точек d -мерного пространства (т.е. классифицировать Hd примеров).
- Одного скрытого слоя нейронов с *сигмоидной* функцией активации достаточно для аппроксимации любой границы между классами со сколь угодно высокой точностью.

Регрессия (Аппроксимация)

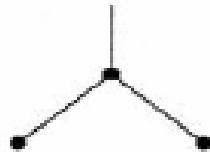
- **Одного скрытого слоя нейронов с *сигмоидной* функцией активации достаточно для аппроксимации любой функции со сколь угодно высокой точностью. (Более того, такая сеть может одновременно аппроксимировать и саму функцию и ее производные.)**

Точность аппроксимации возрастает с числом нейронов скрытого слоя.

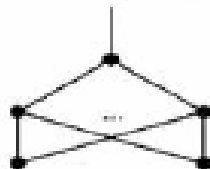
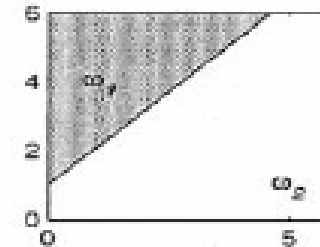
При H нейронах ошибка оценивается как $O(1/H)$.

Связь структуры персептрона с его классифицирующей способностью

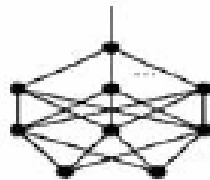
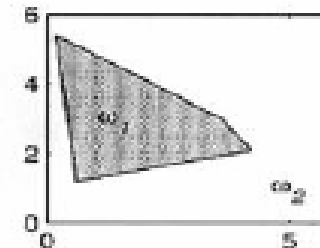
конфигурация НС



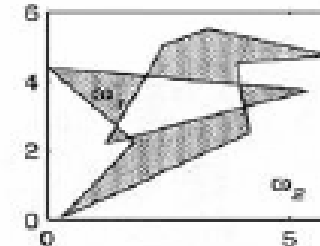
полупространства,
ограниченные
гиперплоскостями



выпуклые области
(открытые или закрытые)



произвольные области



Алгоритм обучения нейрона Маккалока-Питтса в персептроне Розенблатта (бинарного)

- Случайно выбираются веса w_{ij}
- На входы подается обучающий вектор x и рассчитывается выходной сигнал y_i с использованием пороговой функции
- Если ожидаемое значение d_i совпадает с y_i , то веса не изменяются
- Если $y_i=0$ и $d_i=1$, то $w_{ij}(t+1)=w_{ij}(t)+x_j$
- Если $y_i=1$ и $d_i=0$, то $w_{ij}(t+1)=w_{ij}(t)-x_j$
- Повторяются шаги, начиная со второго, для новых примеров

Правило обучения Видроу-Хоффа (для персептрона с аналоговыми выходами)

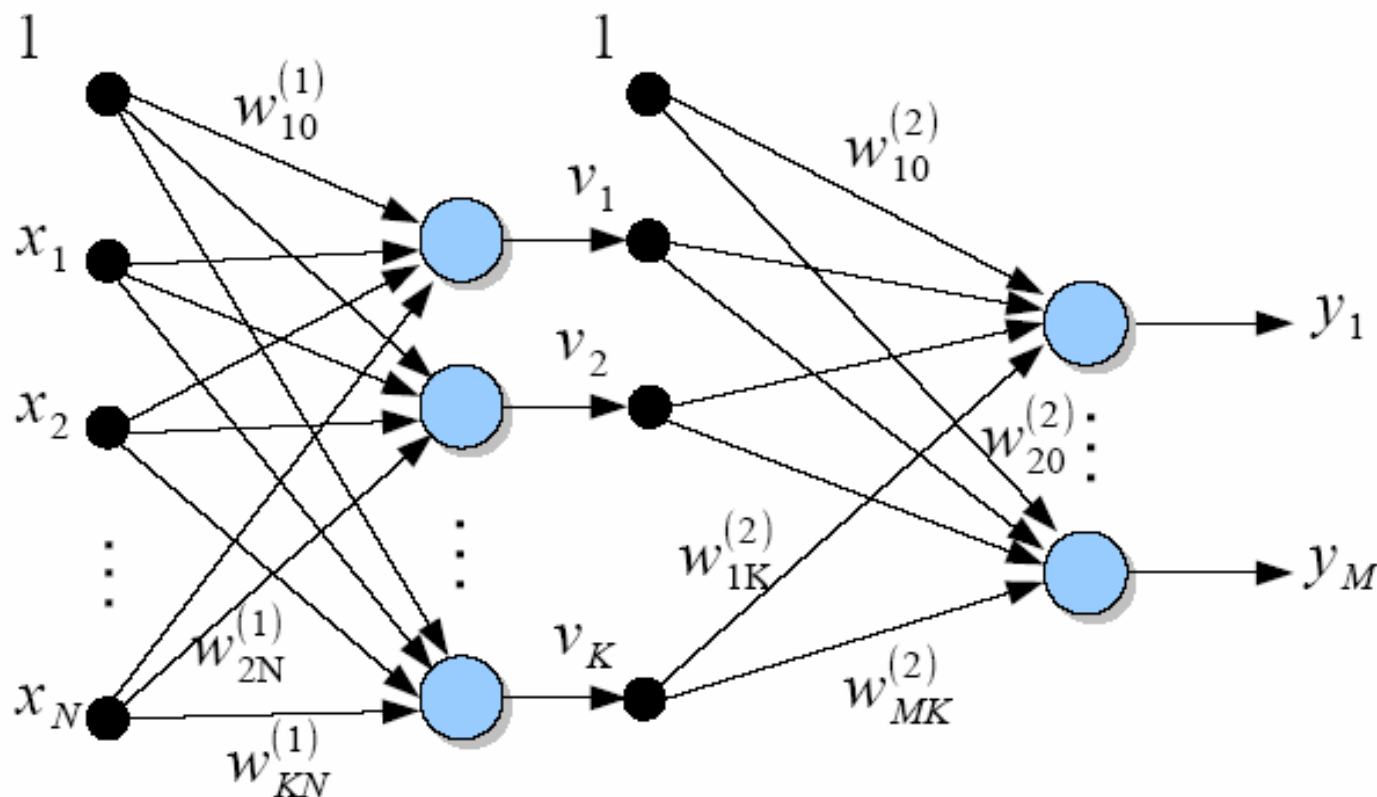
$$w_{ij}(t+1) = w_{ij}(t) + x_j(d_i - y_i)$$

Задача минимизации целевой функции

$$E = \sum_{k=1}^p (y^{(k)}_i - d^{(k)}_i)^2$$

где p – количество примеров

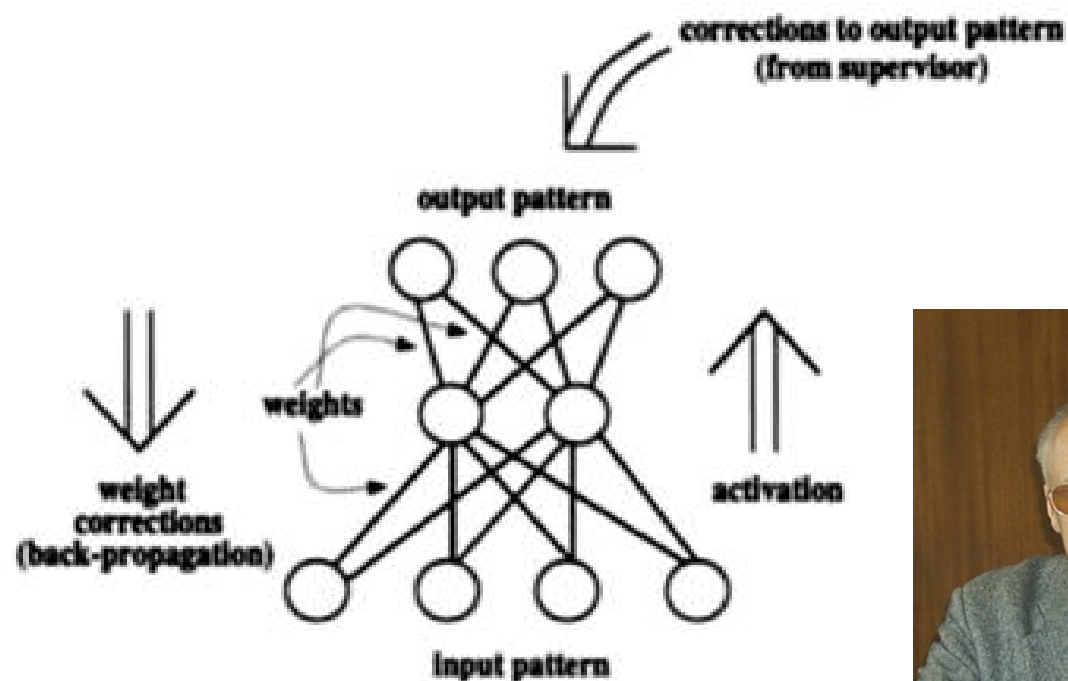
Многослойный персептрон (сеть прямого распространения)



Два режима работы:

- 1) Режим обучения
- 2) Режим использования обученной сети

Многослойный персептрон (2)



Paul Werbos



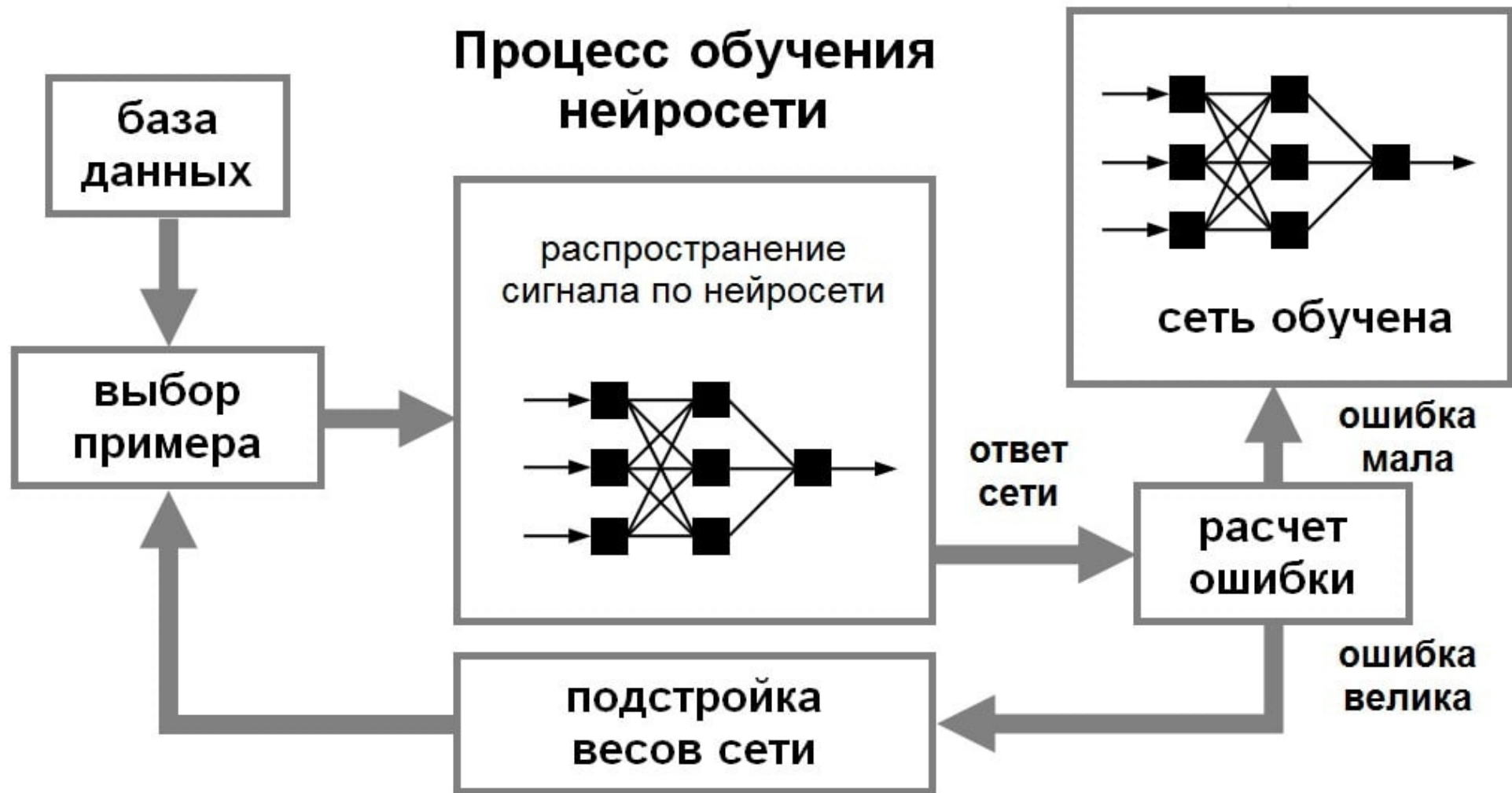
А.И.Галушкин

Гаврилов А.В.
НГТУ, кафедра ВТ



Д.Е.Румелхарт

Обучение НС



Обучение многослойного персептрона

- Цикл обучения (на множестве примеров)
 - Предъявление сети очередного примера (входного и выходного векторов)
 - Вычисление фактического выходного вектора (прямое распространение сигналов по слоям)

$$y_k = f\left(\sum_{i=0}^K w_{ki}^{(2)} v_i\right) = f\left(\sum_{i=0}^K w_{ki}^{(2)} \left(\sum_{j=0}^N w_{ij}^{(1)} x_j\right)\right)$$

- Изменение весов связей с целью минимизации ошибки на выходе (обратное распространение ошибки по слоям)
- Изменение весов – итерационный процесс. Критерий останова – исчерпание лимита количества итераций или достижение заданной величины ошибки E

Обучение многослойного персептрона (2)

- В основе алгоритма обучения обратным распространением ошибки лежит минимизация ошибки методом градиентного спуска

$$E = \frac{1}{2} \sum_{k=1}^P \|\vec{y}^{(k)} - \vec{d}^{(k)}\|^2 = \frac{1}{2} \sum_{k=1}^P \sum_{i=1}^M (y_i^{(k)} - d_i^{(k)})^2$$

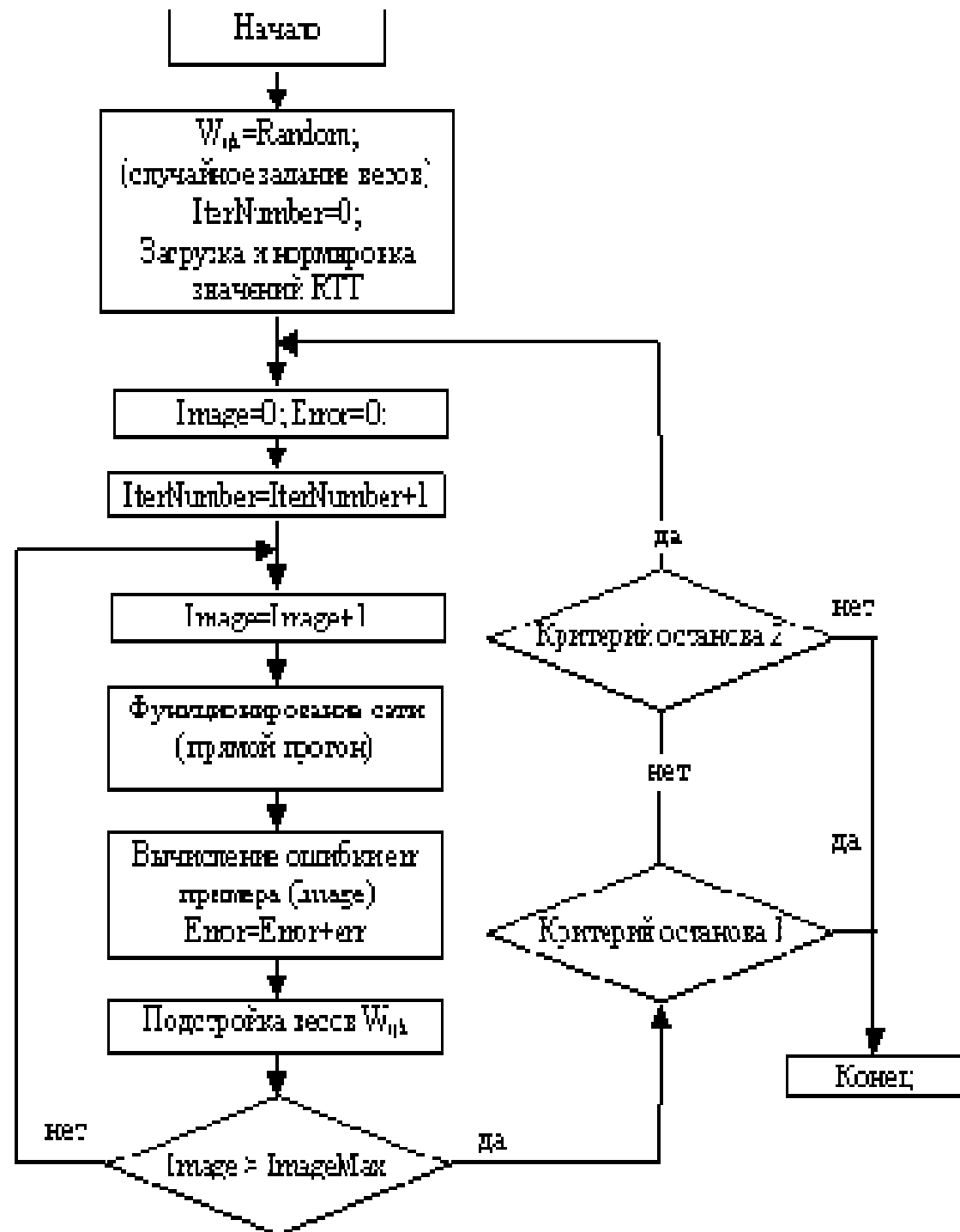
Нужно двигаться в сторону, обратную *градиенту*. Градиент — направление, в котором достигается наибольший прирост значений:

$$\nabla E(w_0, \dots, w_n) = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n} \right].$$

Значит, веса нужно подправлять так:

$$w_i \leftarrow w_i - \eta \frac{\partial E}{\partial w_i}.$$

Обучение многослойного персептрона (3)



Обучение методом обратного распространения
ошибки (Error Back Propagation) для
многослойных персептронов с сигмоидными
функциями активации (с диапазоном от 0 до 1)

$$w_{i,j}(t+1) = w_{i,j}(t) + r g_j x_i'$$

Для выходного слоя $g_j = y_j(1 - y_j)(d_j - y_j)$

Для других слоев $g_j = x_j'(1 - x_j') \sum_k g_k w_{jk}$

Модифицированный (инерционный) алгоритм

$$w_{i,j}(t+1) = w_{i,j}(t) + r g_j x_j' + a(w_{i,j}(t) - w_{i,j}(t-1))$$

Виды градиентного спуска (1)

- Градиентный спуск (Gradient Descent), он же Пакетный градиентный спуск (Batch Gradient Descent) На каждой итерации алгоритм вычисляет градиент функции потерь, используя ВЕСЬ набор данных (все обучающие примеры).
- Плюсы:
 - **Вычислительная эффективность:** Для больших датасетов расчет градиента по всем данным сразу может быть highly optimized через векторные операции (например, с использованием GPU).
 - **Стабильность:** Направление спуска является наиболее точным и верным для всего набора данных. Он уверенно движется прямо к локальному минимуму.
 - **Детерминированность:** При одинаковых начальных условиях всегда приведет к одному и тому же результату.
- Минусы:
 - **Медленные шаги:** Одна итерация требует полного прохода по всем данным. Для огромных датасетов (миллионы примеров) один шаг становится невыносимо медленным.
 - **Риск застревания в локальных минимумах:** Из-за своей стабильности метод с большей вероятностью застрянет в первом попавшемся локальном минимуме, не имея "шума" для выхода из него.
 - **Требует всей выборки в памяти:** Для расчета градиента нужно одновременно иметь доступ ко всем данным, что проблематично для очень больших наборов.

Виды градиентного спуска (2)

- Стохастический градиентный спуск (Stochastic Gradient Descent, SGD). На каждой итерации алгоритм вычисляет градиент функции потерь, используя всего ОДИН случайно выбранный пример из обучающего набора.
- Плюсы:
 - **Скорость:** Один шаг делается очень быстро, так как требуется обработать лишь один пример. Это позволяет делать много шагов за то время, за которое обычный GD сделает один
 - **Возможность избежать локальных минимумов:** Поскольку градиент вычисляется для одного, часто "шумного" примера, направление спуска становится очень нестабильным. Этот "шум" помогает выпрыгнуть из локальных минимумов и найти в итоге более хороший (глобальный) минимум.
 - **Возможность онлайн-обучения:** Модель может обучаться на лету, по мере поступления новых данных.
- Минусы:
 - **Нестабильность:** Путь к минимуму очень извилистый и "шумный". Потери могут сильно колебаться от итерации к итерации.
 - **Низкая вычислительная эффективность:** Векторные операции теряют свою эффективность при работе с одним примером.
 - **Никогда не сходится точно:** Из-за шума метод будет колебаться вокруг минимума, но не достигнет его точно. Для решения этой проблемы часто используют затухающую скорость обучения (learning rate decay).

Виды градиентного спуска (3)

- Мини-пакетный градиентный спуск (Mini-batch Gradient Descent). На практике почти всегда используется именно эта гибридная версия, которую чаще всего и называют Stochastic Gradient Descent (SGD). На каждой итерации алгоритм вычисляет градиент функции потерь, используя небольшую случайную подвыборку (mini-batch) данных. Типичные размеры пачек (batch size) — 32, 64, 128, 256.

Виды градиентного спуска (4)

- Плюсы:
 - **Вычислительная эффективность:** Использование мини-пакетов позволяет полностью использовать векторные операции на GPU.
 - **Относительная стабильность:** Направление спуска более стабильно, чем в чистом SGD, так как усредняется по нескольким примерам. Это ведет к более сглаженной и устойчивой сходимости.
 - **Скорость:** Один шаг делается гораздо быстрее, чем в полном GD, так как пачка намного меньше всего датасета.
 - **Возможность избегать локальных минимумов:** Небольшая доля шума от случайного выбора пачки все еще помогает избегать локальных минимумов.

О выходном слое MLP

- Функция активации в выходном слое зависит от типа решаемой задачи. Например,
 - Задача регрессии может иметь единственный выходной нейрон, и этот нейрон может не иметь функции активации (или линейная функция активации).
 - Задача Двоичной классификации (Binary Classification) может иметь один выходной нейрон и использовать сигмоиду для вывода значения от 0 до 1, чтобы представить вероятный класс. С помощью порога, по умолчанию равного 0,5, эти вероятности превращаются в четкий класс.
 - Задача Мультиклассовой классификации (Multi-Class Classification) может иметь несколько нейронов в выходном слое, по одному для каждого класса (например, три нейрона для трех видов цветов радужной оболочки глаза). В этом случае Функция активации Softmax может использоваться для вывода вероятности предсказания сетью каждого из значений класса. Выбор результата с наибольшей вероятностью может быть использован для получения четкого значения класса.

Обобщение

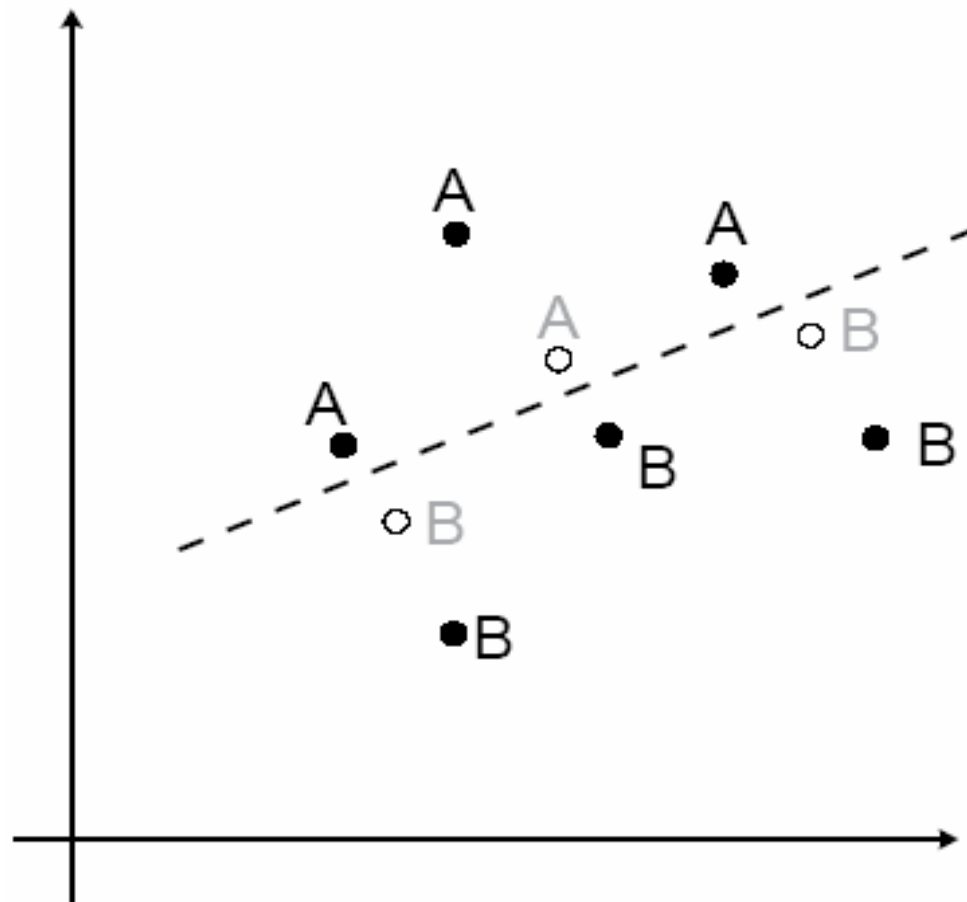
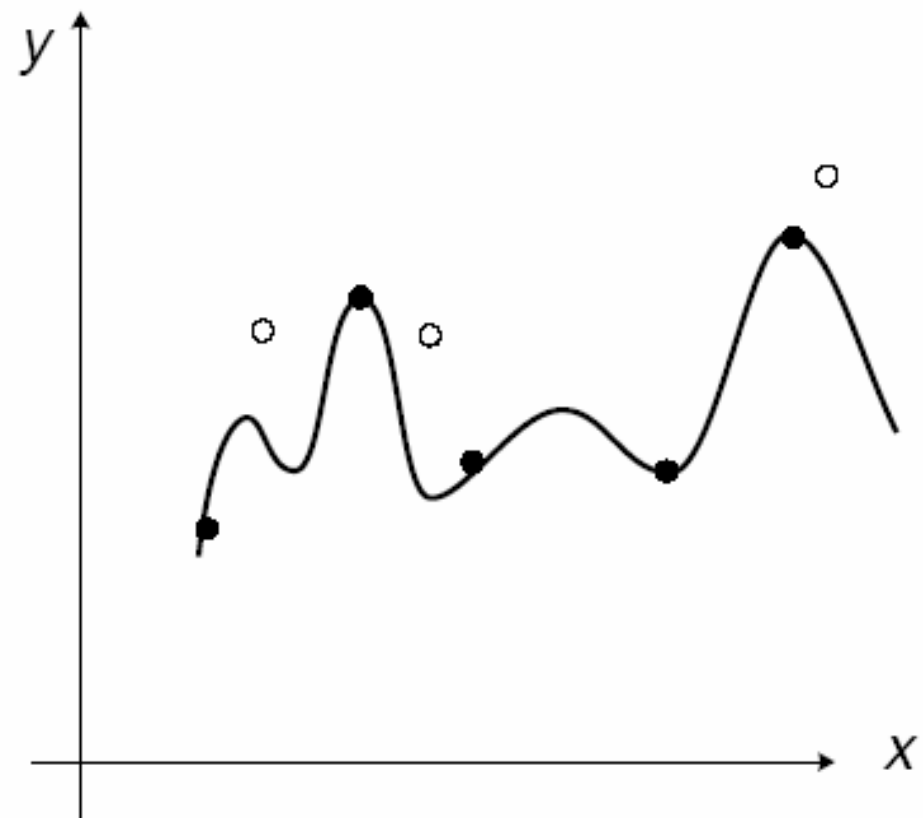
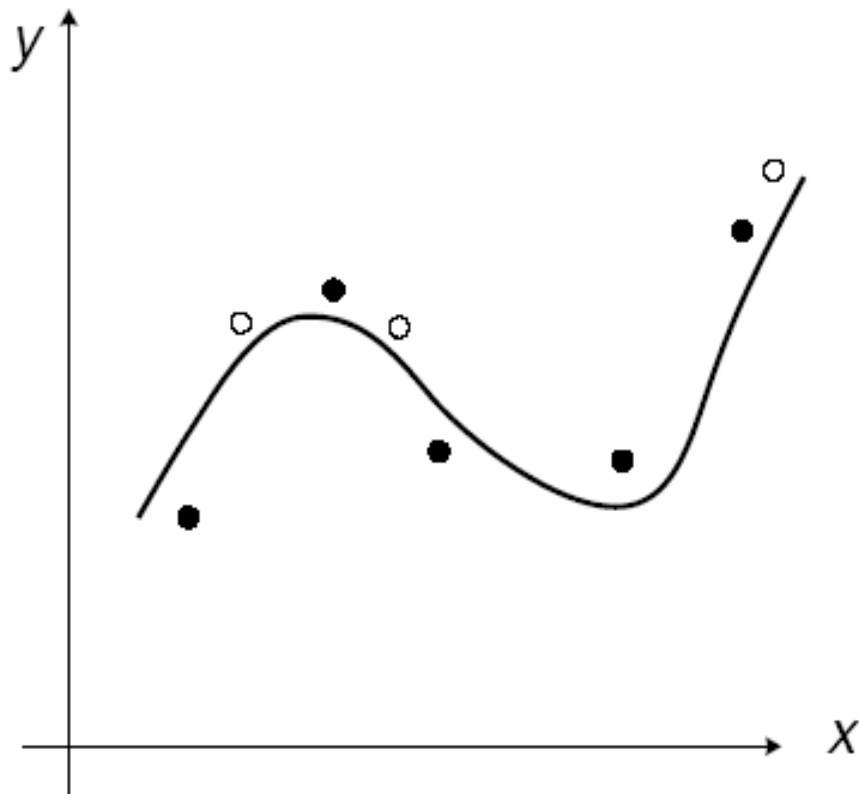


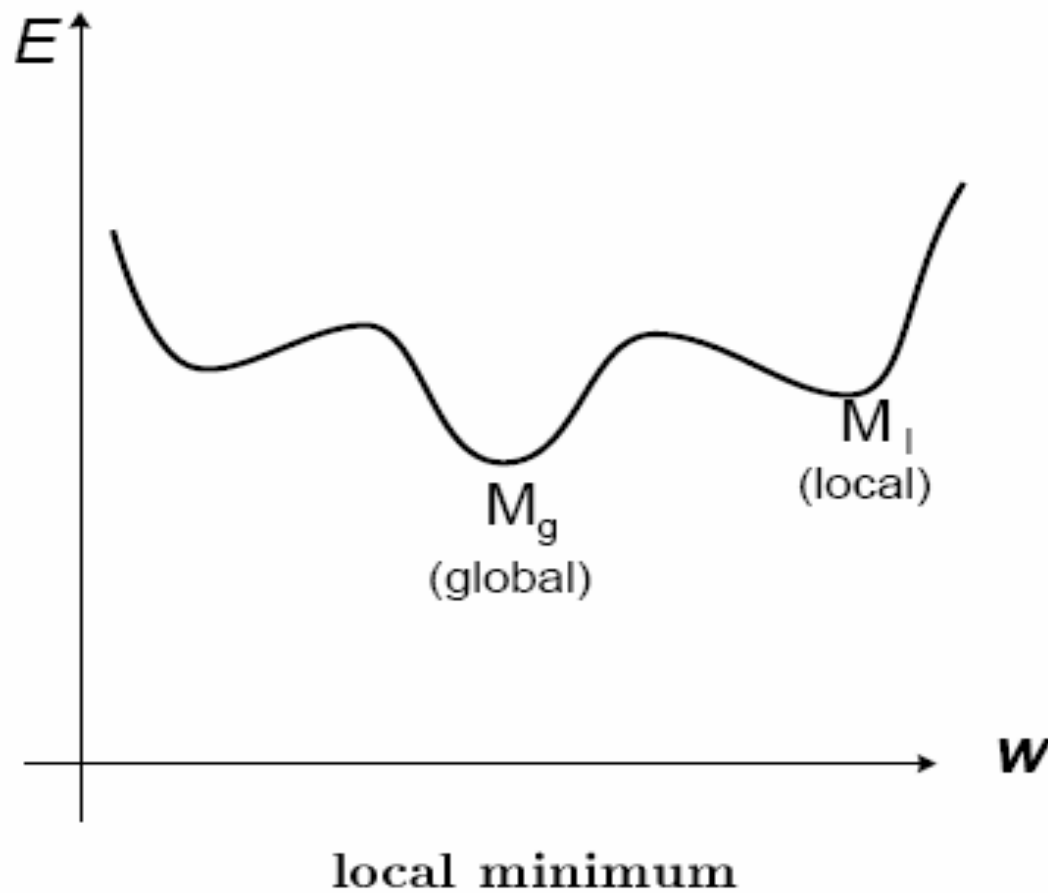
diagram sparse training - two classes

Переобученность (Overfitting)



overfitting in x-y space

Локальный минимум



Достоинства и недостатки многослойного персептрона с обучением обратным распространением ошибки

- Достоинства:
 - Гарантирована возможность решения задачи
- Недостатки:
 - Возможна сходимость к локальному минимуму функции ошибки
 - Низкая скорость обучения
 - Возможность переобученности
 - Невозможность переобучения (дообучения)
 - Необходимо выбирать структуру сети под конкретную задачу (обычно это проблема)
 - Эффект исчезающего градиента в глубоких (первых) слоях

Многослойный персептрон в с использованием библиотеки Scikit-learn

Классификация :

```
clf = MLPClassifier(solver='lbfgs', activation='logistic',  
    hidden_layer_sizes=neur_layer, random_state=1)  
clf.fit(X_train, Y_train)  
res = clf.predict(X_test)
```

Регрессия:

```
reg = MLPRegressor(hidden_layer_sizes=(64,64,64),  
    activation='relu', random_state=1,  
    max_iter=2000).fit(X_train_scaled, y_train)
```

Многослойный персептрон в с использованием библиотеки Keras

```
model = models.Sequential()  
model.add(layers.Dense(16, activation='relu', input_shape=(2,)))  
model.add(layers.Dense(16, activation='relu'))  
model.add(layers.Dense(1, activation='sigmoid'))  
model.summary()
```

2 полносвязных скрытых слоя с 16 нейронами, 1 выходной
слой с 1 нейроном

Запуск модели

```
H = model.fit(  
    train_data,  
    train_label,  
    epochs=20,  
    validation_data=(test_data, test_label)  
)
```

Различные подходы к обучению НС

- Обучение основанное на коррекции ошибки
 - Используется ошибка ($d-y$) между желаемым и реальным выходом для изменения весов связей с целью уменьшения ошибки
- Вероятностное обучение (машина Больцмана)
 - Адаптация весов видимых нейронов так, чтобы состояния видимых нейронов соответствовали желаемому распределению вероятностей
- Правило Хебба – усиливаются связи между нейронами с похожими (или одинаковыми) состояниями и наоборот

$$w_{ij}(t + 1) = w_{ij}(t) + \eta y_j(t) x_i(t)$$

- Соревновательное обучение
 - Только один нейрон (победитель) в выходном слое активизируется (*winner-take-all*, «победитель получает все»)
- Конструктивное обучение
 - В растущих НС создаются новые нейроны для новых кластеров
- Обучение с использованием генетических алгоритмов (эволюционное)
 - Изменение весов связей
 - Изменение структуры сети
- Глубокое обучение (autoencode - автокодировщик)
 - Обучение отдельно каждого слоя

АЛГОРИТМЫ ОБУЧЕНИЯ НС

Парадигма	Правило обучения	Архитектура	Алгоритм обучения	Решаемая задача
С учителем	Коррекция ошибки	Однослойный и многослойный перцептрон	Алгоритмы обучения перцептрона Обратное распространение Adaline и Madaline	Классификация Распознавание образов Аппроксимация функций Прогнозирование Управление
	Больцмана	Рекуррентная	Алгоритм обучения Больцмана	Классификация Распознавание образов
	Хебба	Многослойная прямого распространения	Линейный дискриминантный анализ	Анализ данных Классификация Распознавание образов
	Соревнование	Соревнование	Векторное квантование	Категоризация в середине класса Сжатие данных
		Сеть ART	ARTMap	Классификация Распознавание образов

Парадигма	Правило обучения	Архитектура	Алгоритм обучения	Решаемая задача
Без учителя	Коррекция ошибки	Многослойная прямого распространения	Проекция Саммона	Категоризация в середине класса Анализ данных
	Хебба	прямого распространения или соревнование	Анализ главных компонентов	Анализ данных Сжатие данных
		Сеть Хопфилда	Обучение ассоциативной памяти	Ассоциативная память
	Соревнование	Соревнование	Векторное квантование	Категоризация Сжатие данных
		SOM Кохонена	SOM Кохонена	Категоризация Анализ данных
		Сеть ART	ART1, ART2	Категоризация
	Соревнование	Сеть RBF	Алгоритм обучения RBF	Классификация Распознавание образов Аппроксимация функций Прогнозирование Управление
Смешанная	Коррекция ошибки и соревнование	Сеть RBF	Алгоритм обучения RBF	Классификация Распознавание образов Аппроксимация функций Прогнозирование Управление