Машинное обучение: нейронные сети и глубокое обучение (введение)

Эмели Драль
MADE academy, Москва 2020

Алгоритмы машинного обучения

1. Обучение с учителем: линейные модели
2. Обучение с учителем: ансамбли моделей
3. **Обучение с учителем: нейросетевые модели**
4. Обучение без учителя: обзор методов
5. Обучение с подкреплением
6. (optional) Рекомендательные системы

Нейронные сети и глубокое обучение

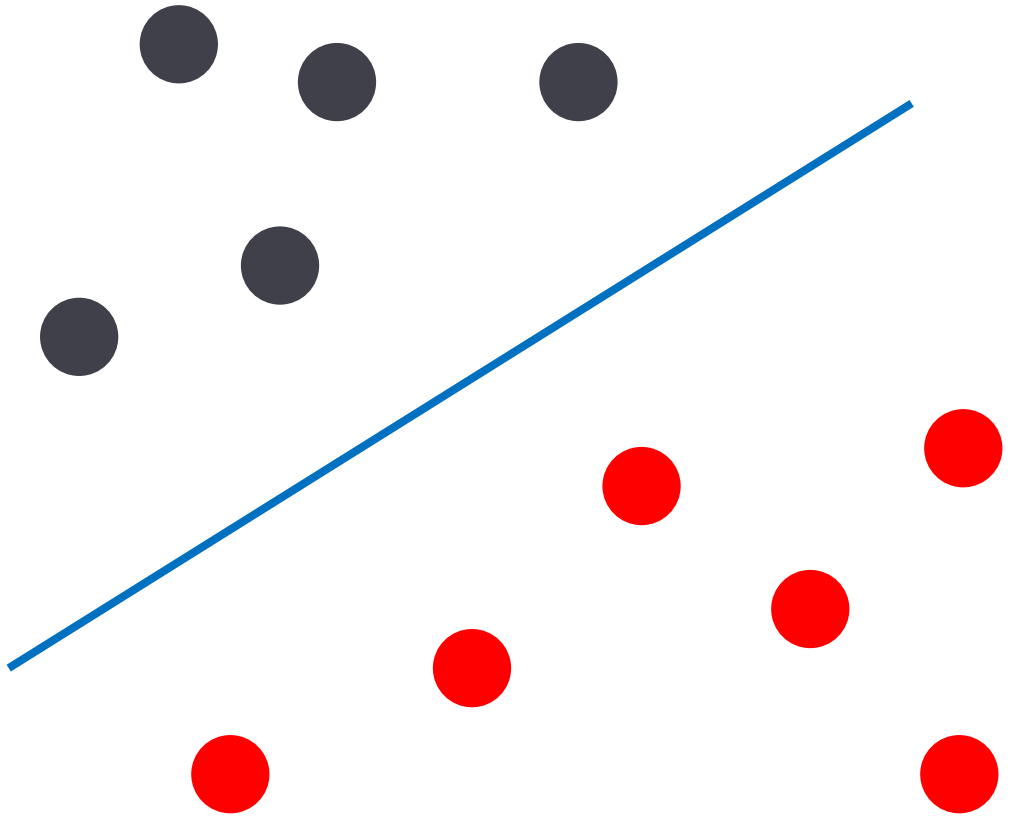
1. Связь с линейными моделями
2. Модель нейронной сети
3. Обучение модели
4. Оптимизация модели
5. Популярные приложения
6. Практические рекомендации

Связь с линейными моделями

Связь с линейными моделями

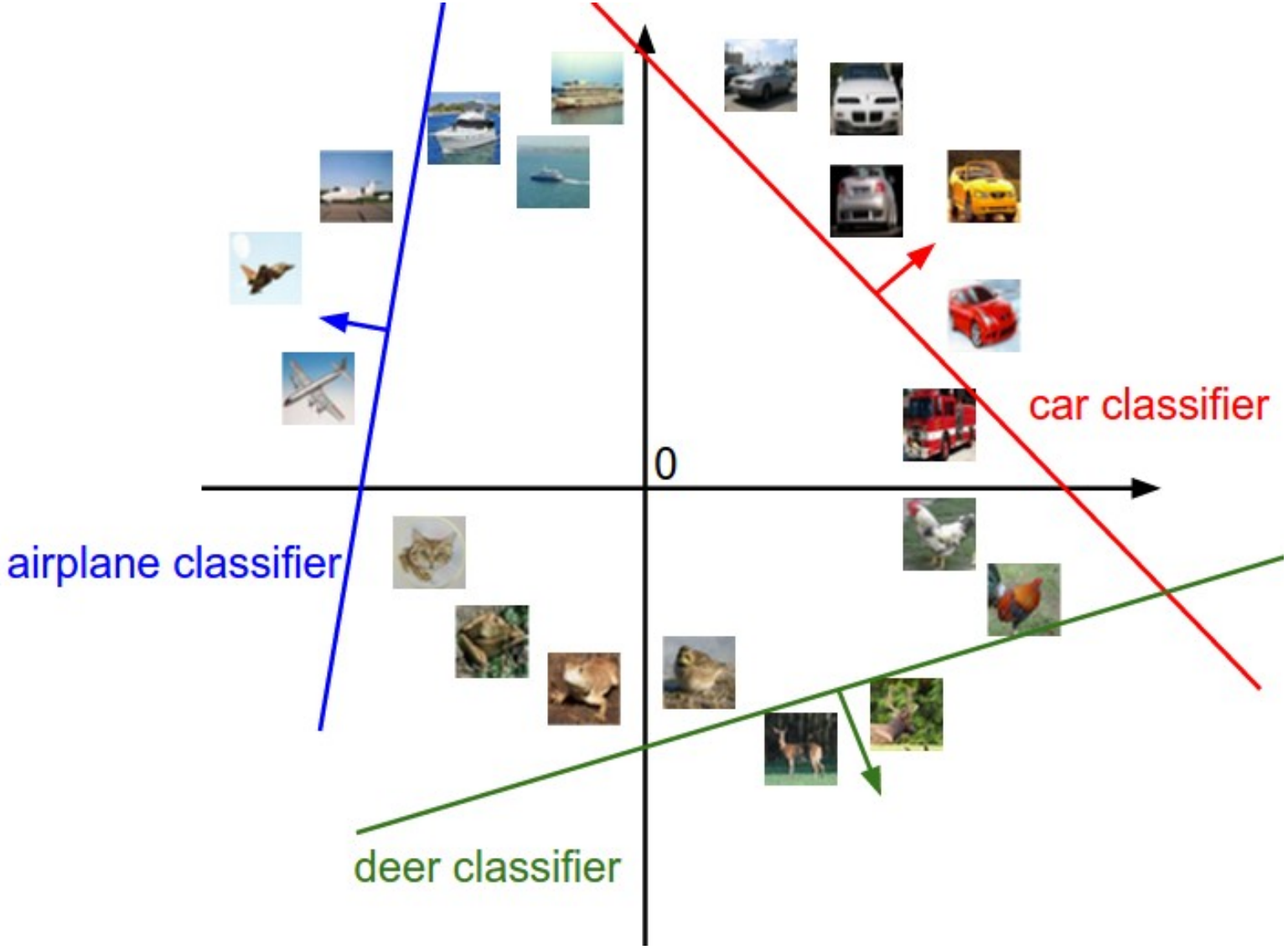
Линейная классификация

$$a(x) = sign(\langle w, x \rangle + w_0)$$



Связь с линейными моделями

Линейная классификация



Credits to http://cs231n.stanford.edu/slides/

Связь с линейными моделями

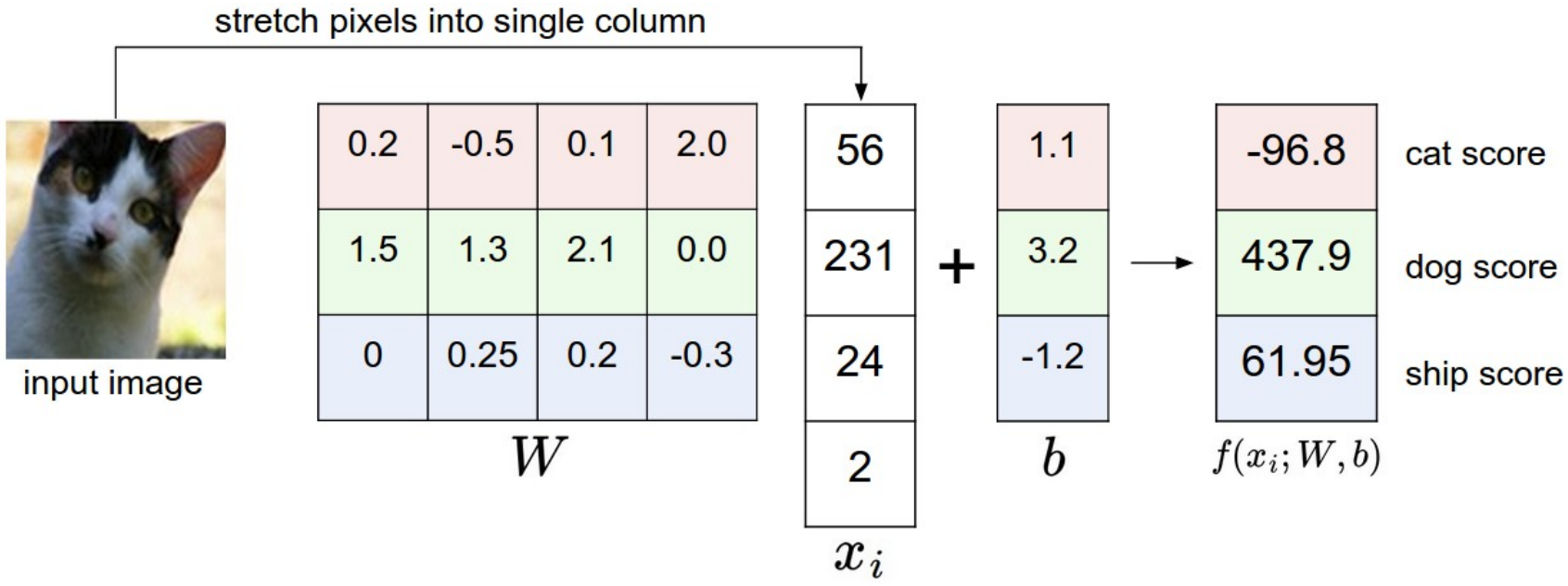
Как компьютер видит изображение?

```
08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48
```

Credits to http://cs231n.stanford.edu/slides/

Связь с линейными моделями

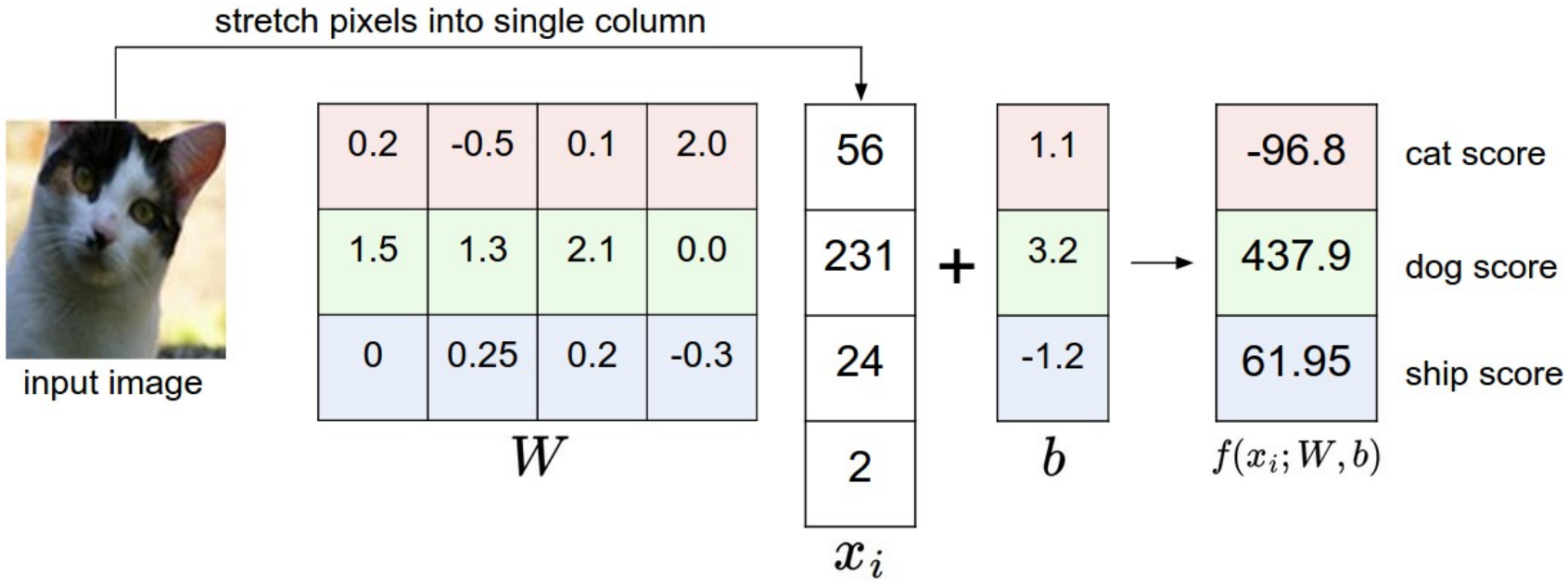
Применим линейную модель для классификации изображения



Credits to http://cs231n.stanford.edu/slides/

Связь с линейными моделями

Применим линейную модель для классификации изображения



Веса линейных моделей можно интерпретировать.

Что произойдет если мы возьмем вектор весов и применим операцию resize для получения матрицы с размерами исходного изображения?

Credits to http://cs231n.stanford.edu/slides/

Связь с линейными моделями

Шаблоны

Credits to http://cs231n.stanford.edu/slides/

Связь с линейными моделями

В чем проблема такого подхода?

Давайте подумаем, насколько данные подход можно обобщить.

Потенциальные проблемы: фон

Связь с линейными моделями

Credits to https://koteiki.com/wp-content/uploads/2018/03/img_5a27db2cac644.png
https://koteiki.com/wp-content/uploads/2018/03/img_5a27d9bfb2f71.png

Связь с линейными моделями

Потенциальные проблемы: смещения, повороты

Credits to https://i.pinimg.com/originals/a0/81/8f/a0818fe4db0c3ab36d3fb7aad7b8b068.jpg
https://express-novosti.ru/images/images/5d89eb8da10a36.15971662.jpg

Связь с линейными моделями

Потенциальные проблемы: много объектов на одном изображении

Credits to https://i.pinimg.com/originals/9f/4e/97/9f4e97815d40efe8ebfceedfb2dfa281.jpg

Связь с линейными моделями

Потенциальные проблемы: объект частично скрыт

Credits to http://tutfon.ru/wallpapers/krasivye-kartinki-vesny/ryzhij-kot-spryatalsya-v-zelenoj-trave-12623.html
https://cs8.pikabu.ru/post_img/big/2018/03/03/11/152010294016702229.jpg

Задачи машинного обучения

Идея: усложним модель?

Задачи машинного обучения

Идея: усложним модель?

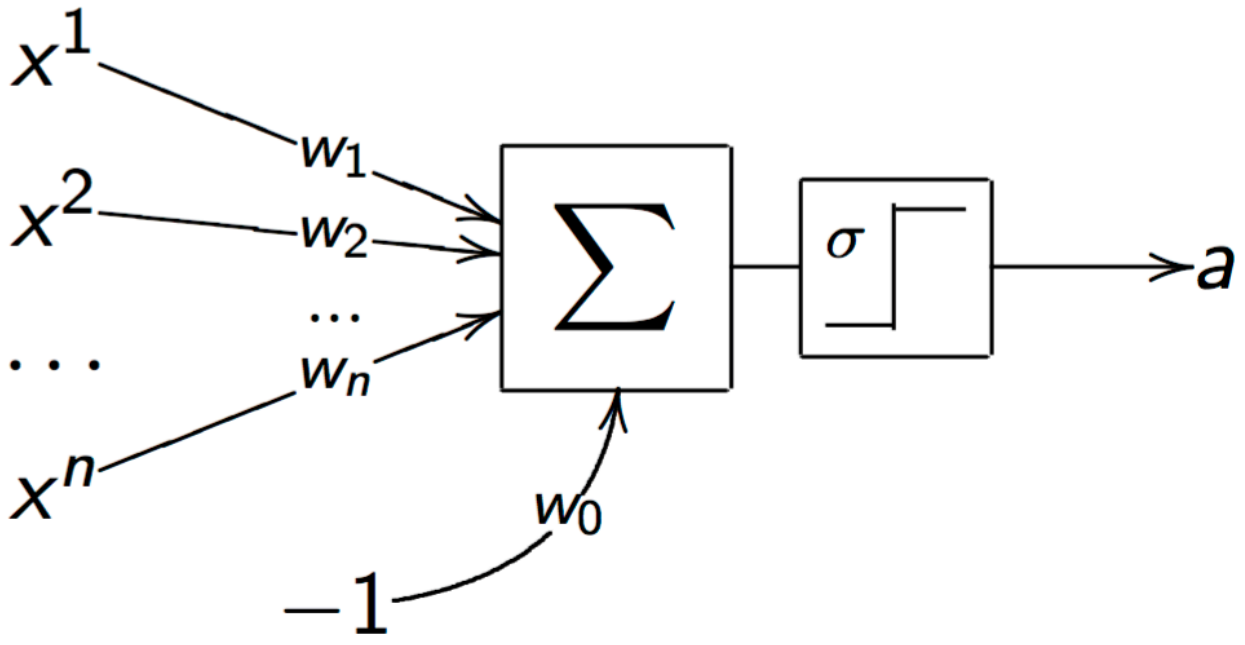
Давайте построим суперпозицию линейных моделей:

- Будем строить линейную модель над линейными моделями
- Это можно делать много раз (не только 2!)
- Надо будет добавить нелинейность, ведь суперпозиция линейных моделей оставляет нас в пространстве линейных моделей

Модель нейронной сети

Модель нейронной сети

Линейная модель нейрона



Чтобы получить линейный классификатор, достаточно взять $\sigma = sign(z)$

Модель нейронной сети

Усложняем модель

1. Строим линейную модель над линейной моделью

$a(x) = W2 \cdot (W1 \cdot x + b1) + b2 = W \cdot x + b$ – снова линейная модель

Модель нейронной сети

Усложняем модель

1. Строим линейную модель над линейной моделью

$a(x) = W2 \cdot (W1 \cdot x + b1) + b2 = W \cdot x + b$ – снова линейная модель

2. Добавляем нелинейность

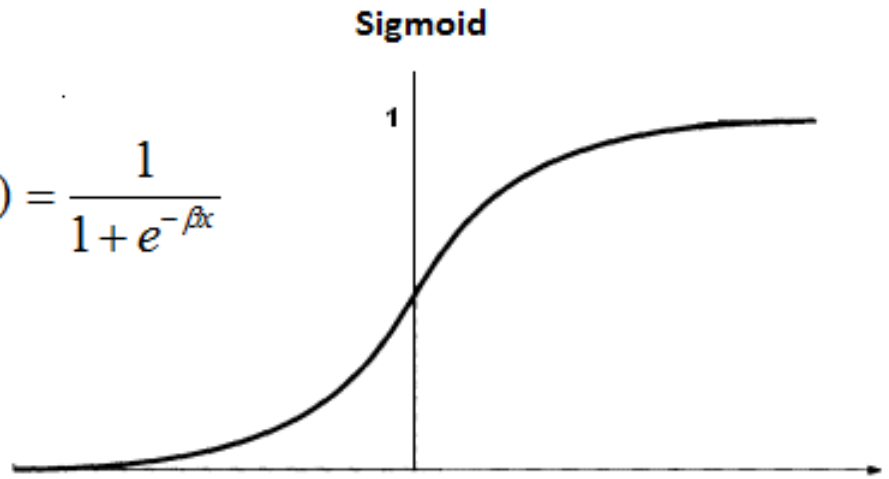
$a(x) = W2 \cdot \sigma(W1 \cdot x + b1) + b2 \mathrel{!}= W \cdot x + b$ – это уже не линейная модель, значит у нас получилось усложнить

Модель нейронной сети

Усложняем модель

Как ввести нелинейность?

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



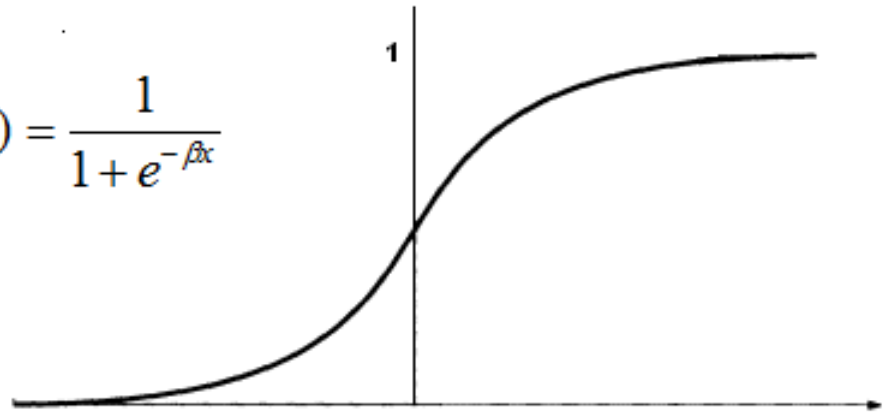
Модель нейронной сети

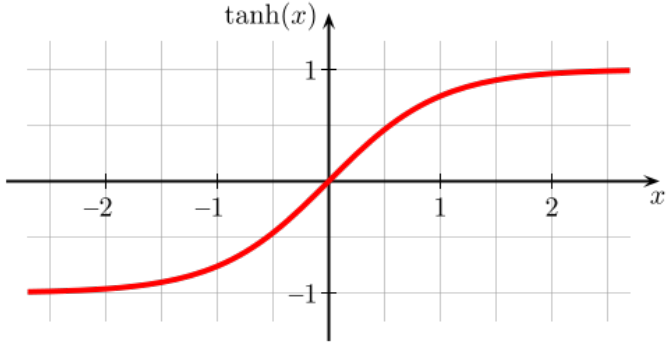
Усложняем модель

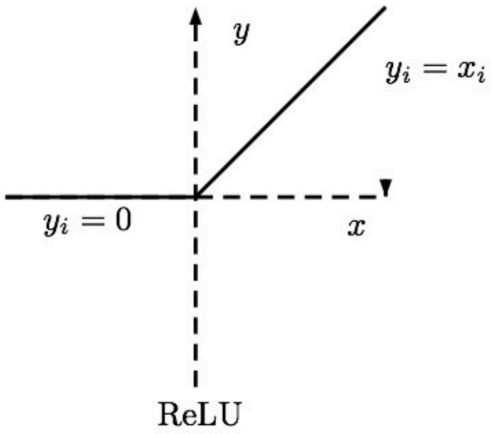
Как ввести нелинейность?

Sigmoid

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

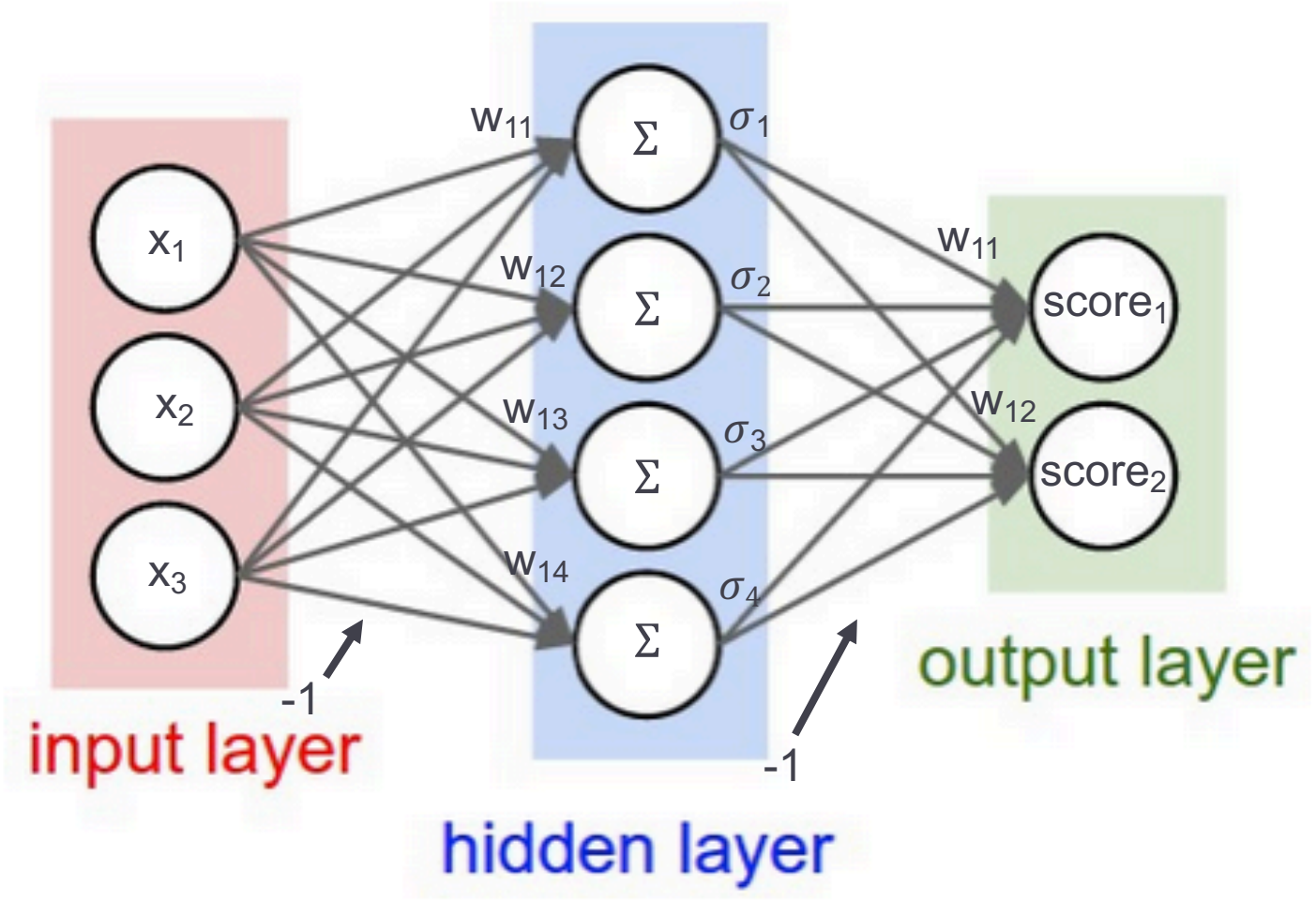






Модель нейронной сети

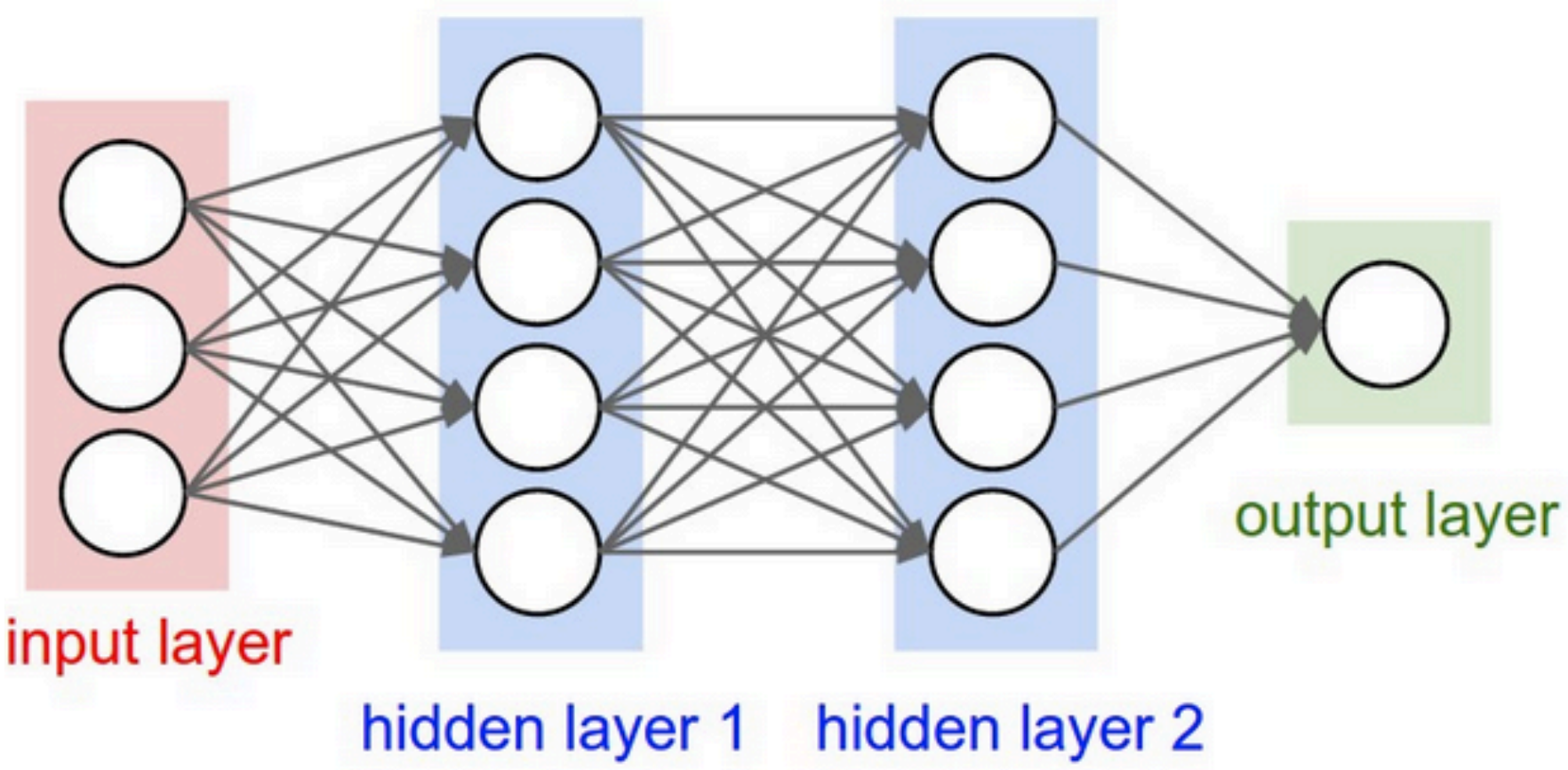
Собираем полносвязную нейронную сеть



Credits to http://cs231n.stanford.edu/slides/

Модель нейронной сети

Собираем полносвязную нейронную сеть



Credits to http://cs231n.stanford.edu/slides/

Обучение модели

Обучение модели

Задача оптимизации для линейно модели

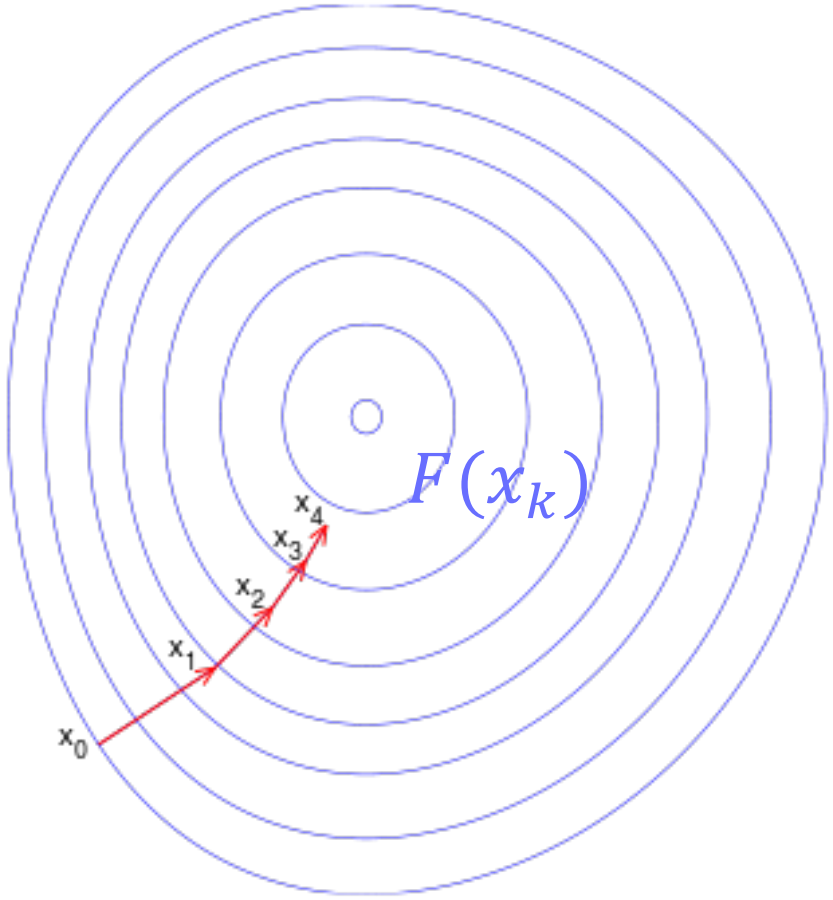
$$a(x) = \langle w, x \rangle + w_0$$

$$Q = \sum_{i=1}^{l} L\big(y_i, a(x_i)\big) \to \min_w$$

Обучение модели

Решение задачи оптимизации

Численное решение возможно с помощью градиентного спуска (GD, Gradient Decent)



$$x_{k+1} = x_k - \gamma_k \nabla F(x_k)$$

Обучение модели

Стохастический градиент (SGD)

$$w_{k+1} = w_k - \gamma_k \sum_{i=1}^{l} y_i x_i L'(M_i)$$

$$w_{k+1} = w_k - \gamma_k y_i x_i L'(M_i)$$

x_i – случайный элемент обучающей выборки

Обучение модели

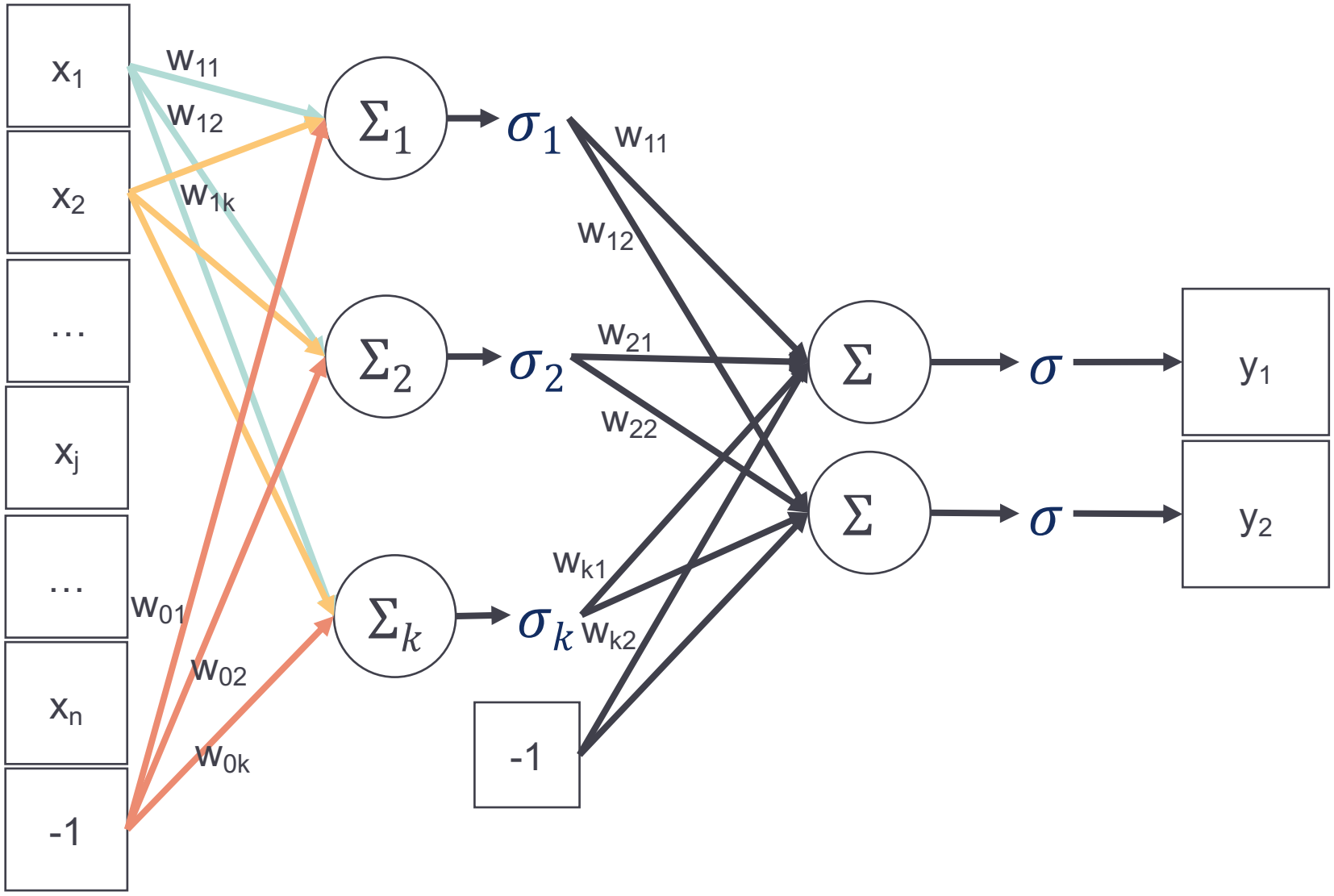
Применим SGD для нейронной сети

$$Q = \sum_{i=1}^{l} L\big(y_i, a(x_i)\big) \to \min_w$$

Проблема: в нейронной сети гораздо больше параметров *w*. Это приводит к проблемам при расчете градиента функции потерь L’(*w*)

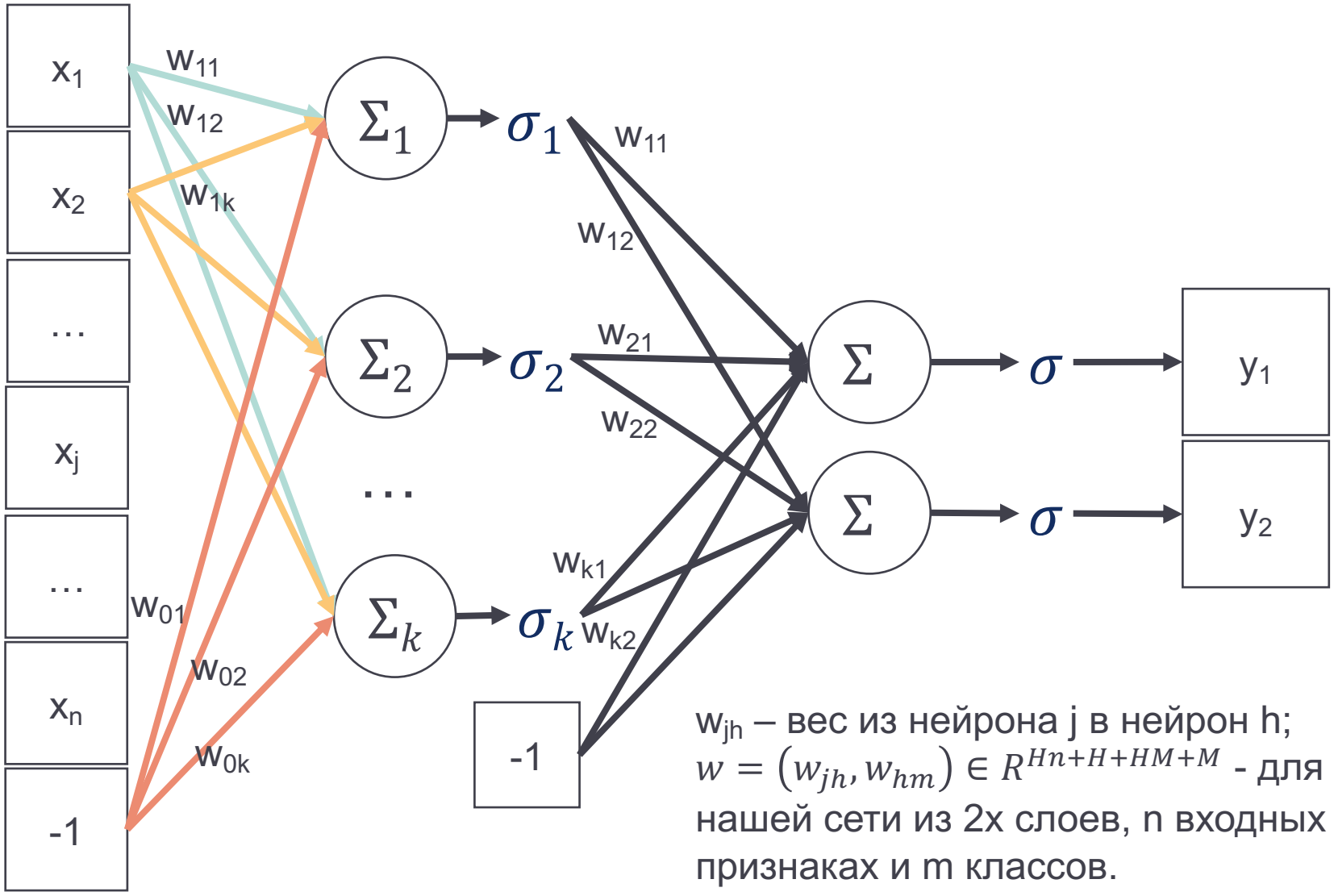
Обучение модели

Нейронная сеть



Обучение модели

Нейронная сеть



w_{jh} – вес из нейрона j в нейрон h;
$w = (w_{jh}, w_{hm}) \in R^{Hn+H+HM+M}$ - для нашей сети из 2х слоев, n входных признаках и m классов.

Обучение модели

Back Propagation

Продолжим анализировать пример с 2х-слойной нейронной сетью.

Запишем прогноз нейрона *m* на последнем слое:

$a^m(x_i) = \sigma_m(\sum_{h=0}^{H} w_{hm} u^h(x_i))$, где

$u^h(x_i) = \sigma_h(\sum_{j=0}^{J} w_{jh} f_j(x_i))$ - нейроны с промежуточного слоя.

Обучение модели

Back Propagation

Рассмотрим квадратичную функцию потерь

$$L_i(w) = \frac{1}{2}\sum_{m=1}^{M}(a^m(x_i) - y_i^m)^2$$

На каждом объекте мы получаем прогноз в виде вектора размера M. Соответственно, потери мы считаем также по векторам.

Обучение модели

Back Propagation

Чтобы оптимизировать веса градиентными методами, нужно научиться считать градиент потерь по всем весам.

Для этого далее потребуется оценить $\frac{\partial L_i(w)}{\partial a^m}$ и $\frac{\partial L_i(w)}{\partial u^h}$, давайте с них начнем:

$$\frac{\partial L_i(w)}{\partial a^m} = a^m(x_i) - y_i^m, \text{обозначим } \varepsilon_i^m$$

$$\frac{\partial L_i(w)}{\partial u^h} = \sum_{m=1}^{M} \frac{\partial L_i(w)}{\partial a^m} \sigma'_m w_{hm} = \sum_{m=1}^{M} \varepsilon_i^m \sigma'_m w_{hm}, \text{обозначим } \varepsilon_i^h$$

Назовём ε_i^h ошибкой на скрытом слое *h*

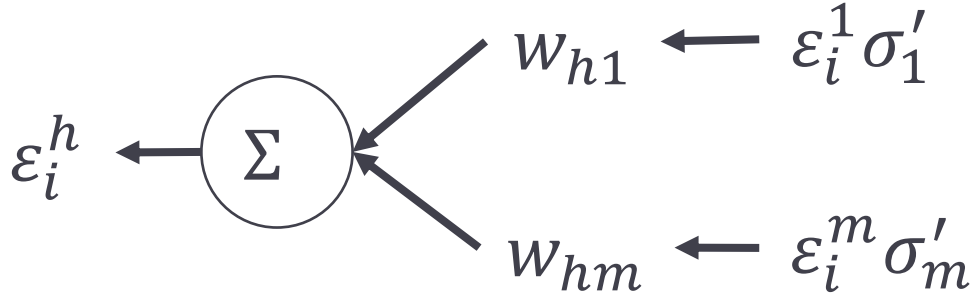
Обучение модели

Back Propagation

$$\frac{\partial L_i(w)}{\partial u^h} = \sum_{m=1}^{M} \varepsilon_i^m \sigma'_m w_{hm} = \varepsilon_i^h$$

Получается, что ε_i^h вычисляется по ε_i^m если запустить сеть задом наперёд (backward pass)

Рассмотрим нейрон со скрытого слоя h:



Получилось, что мы пропустили ошибку обратным ходом через сеть. Отсюда back propagation.

Обучение модели

Back Propagation

А получив частные производные L_i по a^m и по u^h легко найти производные и по весам.

На скрытом слое (по всем *m* и *h*):

$$\frac{\partial L_i(w)}{\partial w_{hm}} = \frac{\partial L_i(w)}{\partial a^m}\frac{\partial a^m}{\partial w_{hm}} = \varepsilon_i^m \sigma_m' u^h(x_i)$$

На входном слое (по всем *h* и *j*) :

$$\frac{\partial L_i(w)}{\partial w_{1h}} = \frac{\partial L_i(w)}{\partial u^h}\frac{\partial u^h}{\partial w_{jh}} = \varepsilon_i^h \sigma_h' f_j\,(x_i)$$

Обучение модели

Оптимизация модели методом Back Propagation

1. Инициализируем исходные веса
2. Выбираем объект или объекты для того, чтобы сделать шаг оптимизации.
3. На каждом шаге:
   - делаем прямой проход (forward pass), чтобы посчитать значения на всех скрытых слоях, прогноз и ошибки на выходном слое
   - делаем обратный проход (backward pass), чтобы рассчитать ошибки на срытых слоях и оценить частные производные
   - зная производные, делаем шаг - обновляем веса
   - считаем потери
4. Повторяем шаг 3, пока потери не стабилизировались

Обучение модели

Оптимизация модели методом Back Propagation

1. Инициализируем исходные веса w
2. Выбираем объект x_i или объекты для того, чтобы сделать шаг оптимизации.
3. На каждом шаге:
   - делаем прямой проход (forward pass), чтобы посчитать значения на всех скрытых слоях $u^h(x_i) = \sigma_h(\sum_{j=0}^{J} w_{jh} f_j(x_i))$, прогноз $a^m(x_i) = \sigma_m(\sum_{h=0}^{H} w_{hm} u^h(x_i))$ и ошибки на выходном слое $\varepsilon_i^m = \frac{\partial L_i(w)}{\partial a^m}$
   - делаем обратный проход (backward pass), чтобы рассчитать ошибки на срытых слоях и оценить частные производные $\varepsilon_i^h = \sum_{m=1}^{M} \varepsilon_i^m \sigma'_m w_{hm}$
   - зная производные, делаем шаг - обновляем веса $w_{jh} = w_{jh} - \eta \varepsilon_i^h \sigma'_h f_j(x_i)$ и $w_{hm} = w_{hm} - \eta \varepsilon_i^m \sigma'_m u^h(x_i)$
   - считаем потери $Q = (1 - \lambda)Q + \lambda L_i$
4. Повторяем шаг 3, пока потери не стабилизировались

Обучение модели

Back Propagation

Достоинства

- Быстрое вычисление градиента
- Потоковое обучение, распределенное обучение
- Простое обобщение для большого числа слоёв, различных архитектур, функций потерь и функций активации

Ограничения

- Скорость сходимости
- Локальные экстремумы
- Переобучение
- Паралич сети (горизонтальные асимптоты функций активации)

Обучение модели

Подробнее про Back Propagation

- Реализация на Python: https://machinelearningmastery.com/implement-backpropagation-algorithm-scratch-python/
- Математическое изложение, см. стр. 13 в лекциях К.В. Воронцова по нейронным сетям: http://www.ccas.ru/voron/download/NeuralNets.pdf
- Популярное и подробное изложение (english), см. стр. 200-217 в Deep Learning Book: https://www.deeplearningbook.org/contents/mlp.html

Оптимизация модели

Оптимизация модели

Оптимизация сходимости

Идея: можем ли мы улучшить способ обновления весов методом градиентного спуска, чтобы ускорить сходимость?

Оптимизация модели

Оптимизация сходимости

Метод накопления импульса:

- Momentun – экспоненциальное скользящее среднее градиента по последним итерациям
- NAG (Nesterov's accelerated gradient) – стохастический градиент с импульсом Нестерова

Адаптивный шаг:

- RMSProp (running mean square) – адаптация скорости изменения весов скользящим средними по послежним итерациям
- AdaDelta (adaptive learning rate) – двойная нормировка приращений весов

А также другие методы и их комбинации!

Оптимизация модели

Пактеная нормализация данных (batch normalization)

$B = \{x_i\}$ – mini-batch или пакет данных

Оптимизация градиентов $L_i(w)$ по пакету ускоряет сходимость

Оптимизация модели

Функции активации

Такие функции активации как сигмоид или гиперболический тангенс могу привести к затуханию градиентов

Функция положительной срезки (rectified linear unit):

- ReLU(y) = max{0; y}
- PReLU(y) = max{0; y} + αmin{0; y} (parametric ReLU)

Оптимизация модели

Начальное приближение весов

Существуют различные эвристики для начального приближения весов:

- Выбор весов случайно из распределений с нулевым средним, например, нормального или равномерного
- Инициализация весами предобученной модели

Интересная идея для начального приближения - обучение нейронов как независимых линейных моделей:

- по подмножествам признаков
- по подмножествам объектов
- из разных начальных приближений

В этом случае обучение будет послойным. Далее модель обучается методом back propagation.

Регуляризация модели

Регуляризация модели

Идея: можем ли мы сделать нейронную сеть более устойчивой?

Регуляризация модели

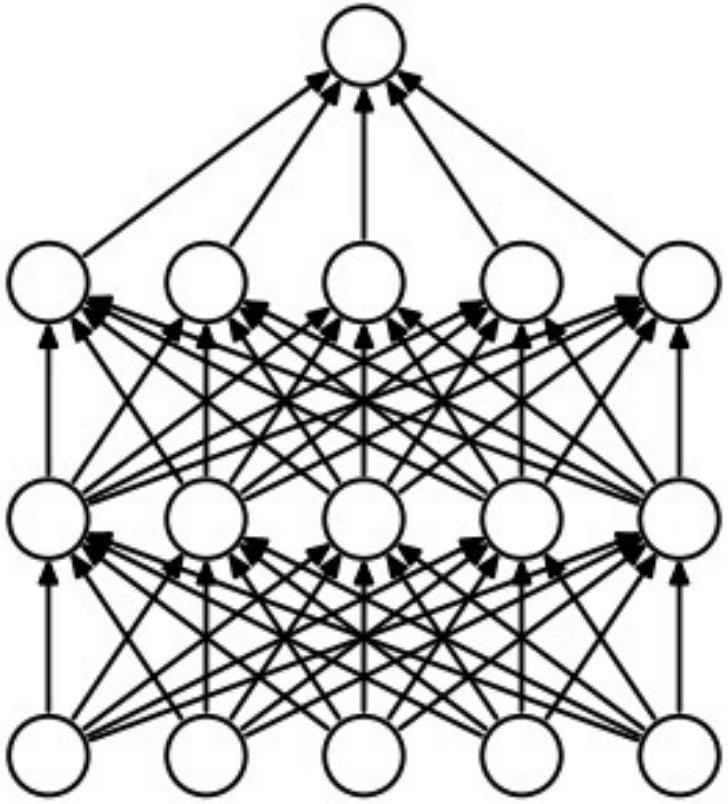
Dropout

Идея: можем ли мы сделать нейронную сеть более устойчивой?

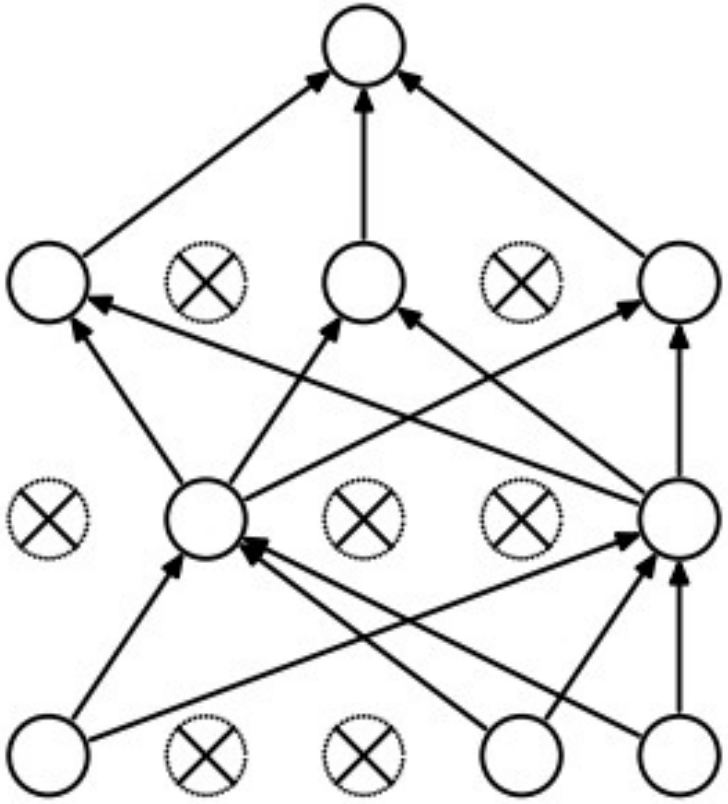
Более устойчивая сеть – такая сеть, которая при отключении части нейронов работает примерно также как исходная.

Регуляризация модели

Dropout



(a) Standard Neural Net

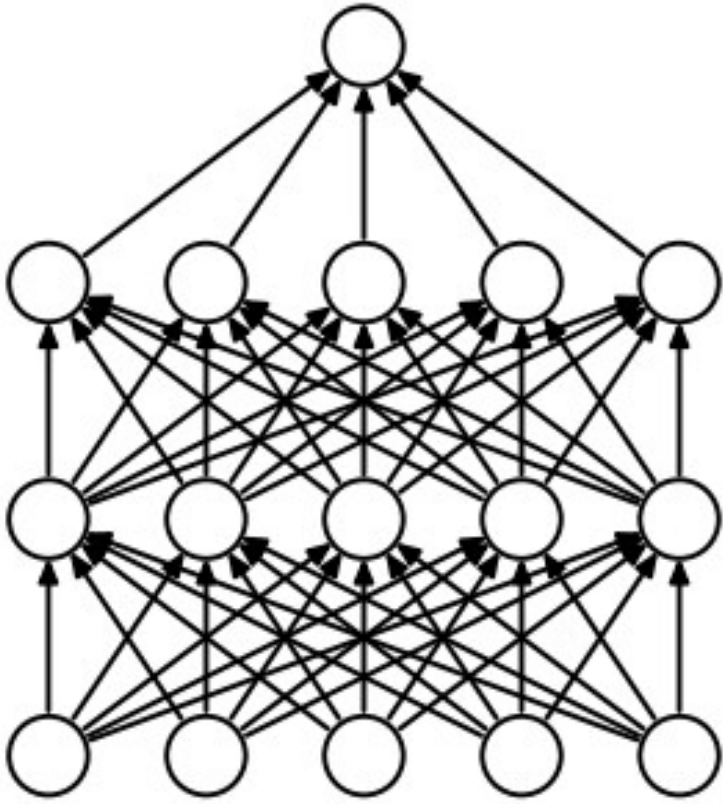


(b) After applying dropout.

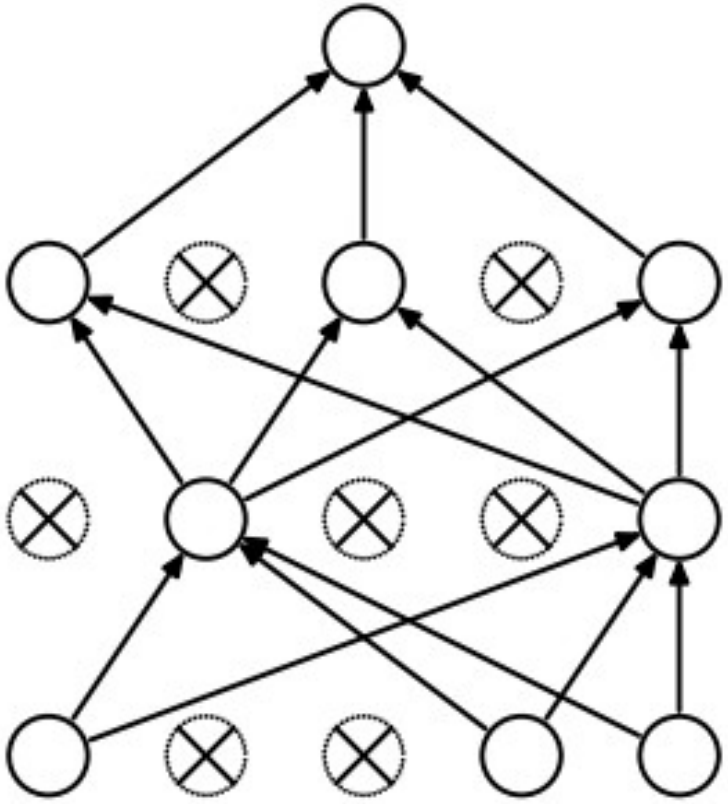
Srivastava, Nitish, et al. ”Dropout: a simple way to prevent neural networks from overfitting”, JMLR 2014

Регуляризация модели

Dropout



(a) Standard Neural Net



(b) After applying dropout.

С вероятность p_l мы отключаем h-ый нейрон слоя l. Это означает, что на вход слоя l+1 из нейрона h предыдущего слоя ничего не придет (придет 0).

Srivastava, Nitish, et al. ”Dropout: a simple way to prevent neural networks from overfitting”, JMLR 2014

Регуляризация модели

Dropout

- Dropout позволяет сделать нейронную сеть сверх устойчивой за счет того, что разные подмножества нейросети работают примерно также, как вся нейросеть
- Этот эффект достигается за счет отключения нейронов во время обучения
- Получается, что мы как будто учим разные подмножества сети приближать целевую функцию независимо, вместо того, чтобы они компенсировали ошибки друг друга

Регуляризация модели

Dropout + *l2* регуляризация

- Помните, мы делали l2 регуляризацию для линейных моделей? А именно минимизировали не только ошибки, но и квадраты весов

$$V(w) = \|w\|_{l2}^2 = \sum_{n=1}^{d} w_n^2$$

Регуляризация модели

Dropout + *l2* регуляризация

- Помните, мы делали l2 регуляризацию для линейных моделей? А именно минимизировали не только ошибки, но и квадраты весов

$$V(w) = \|w\|_{l2}^2 = \sum_{n=1}^{d} w_n^2$$

- Этот подход можно комбинировать с Dropout. Изменится формула для расчета градиента потерь при обучении нейросети, там появится дополнительное слагаемое от регуляризации.

Популярные приложения

Популярные приложения

Популярные приложения

- Computer vision: классификация изображений, детекция объектов, генерация тестовых описаний изображений, обработка и стилизация и пр.
- Natural Language Processing: распознавание речи, оценка тональности текста, поддержание диалога и пр.
- Timeseries analysis: прогнозирование спроса, прогнозирование популяции животных и пр.

А также многое другое: обработка звука; обработка видео; генерация изображений, музыки, текстов и пр.

Популярные приложения

Популярные приложения

- Computer vision
- Natural Language Processing
- Timeseries analysis
- и др.

Для работы с каждым из типов задач используются специализированные архитектуры сетей и оптимизации процесса обучения.

Популярные приложения

Популярные приложения

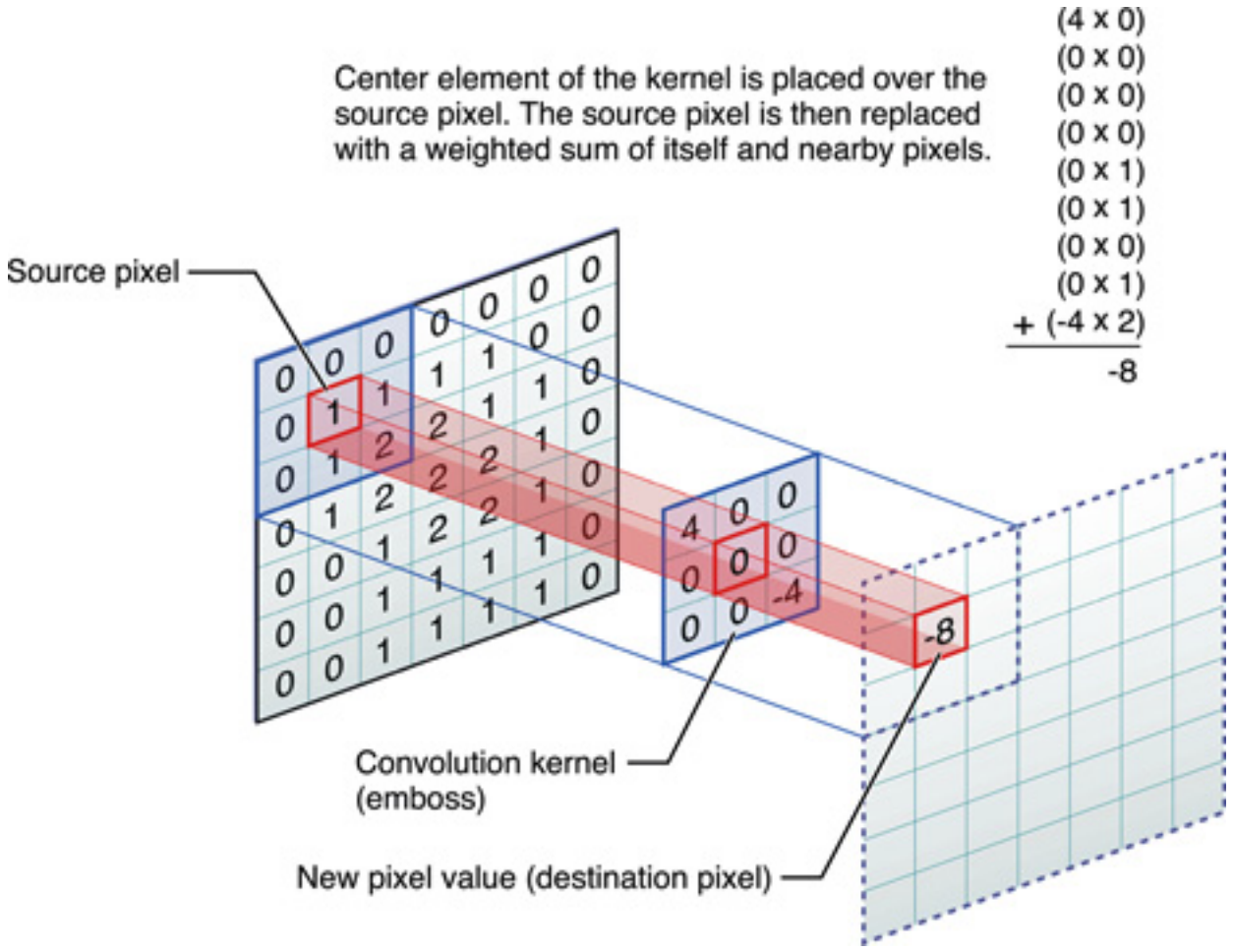
Для работы с каждым из типов задач используются специализированные архитектуры сетей и оптимизации процесса обучения.

Мы коротко обсудим:

- Сверточные сети
- Автоенкодеры
- Рекуррентные сети

Свёрточные сети

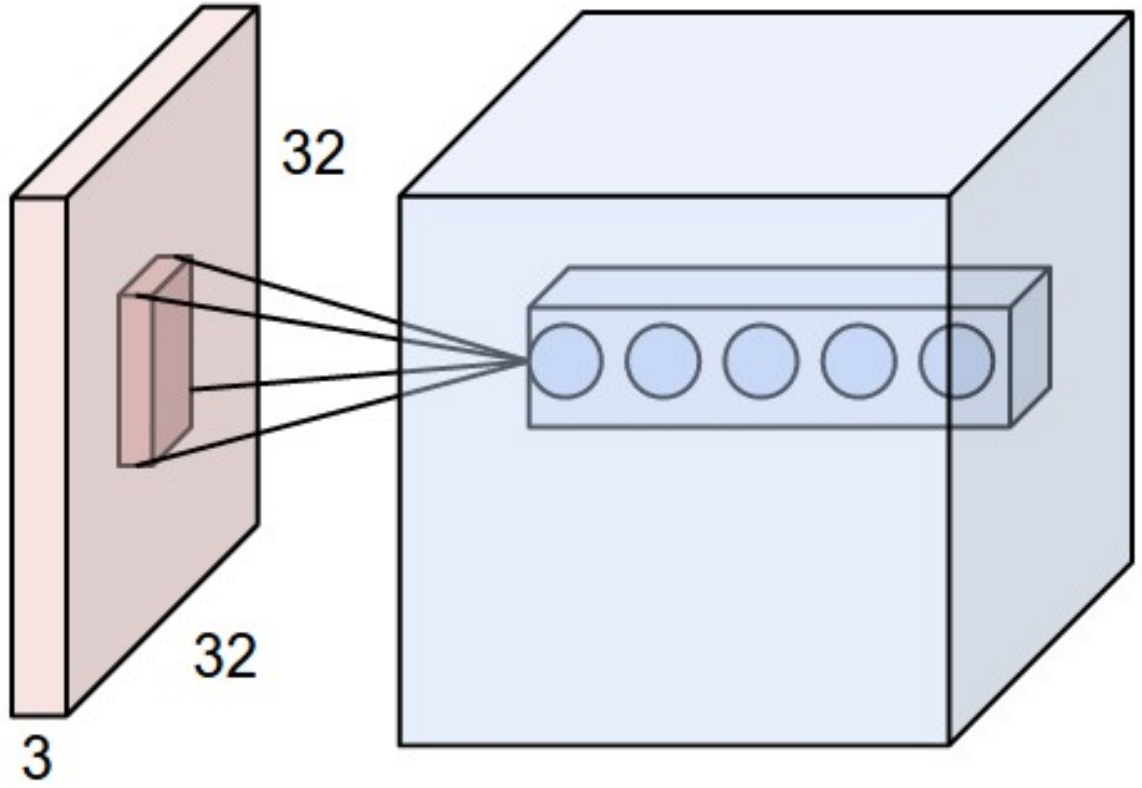
Convolution (свёртка)



Credits to http://cs231n.stanford.edu/slides/

Свёрточные сети

Convolution (свёртка)



Credits to http://cs231n.stanford.edu/slides/

Свёрточные сети

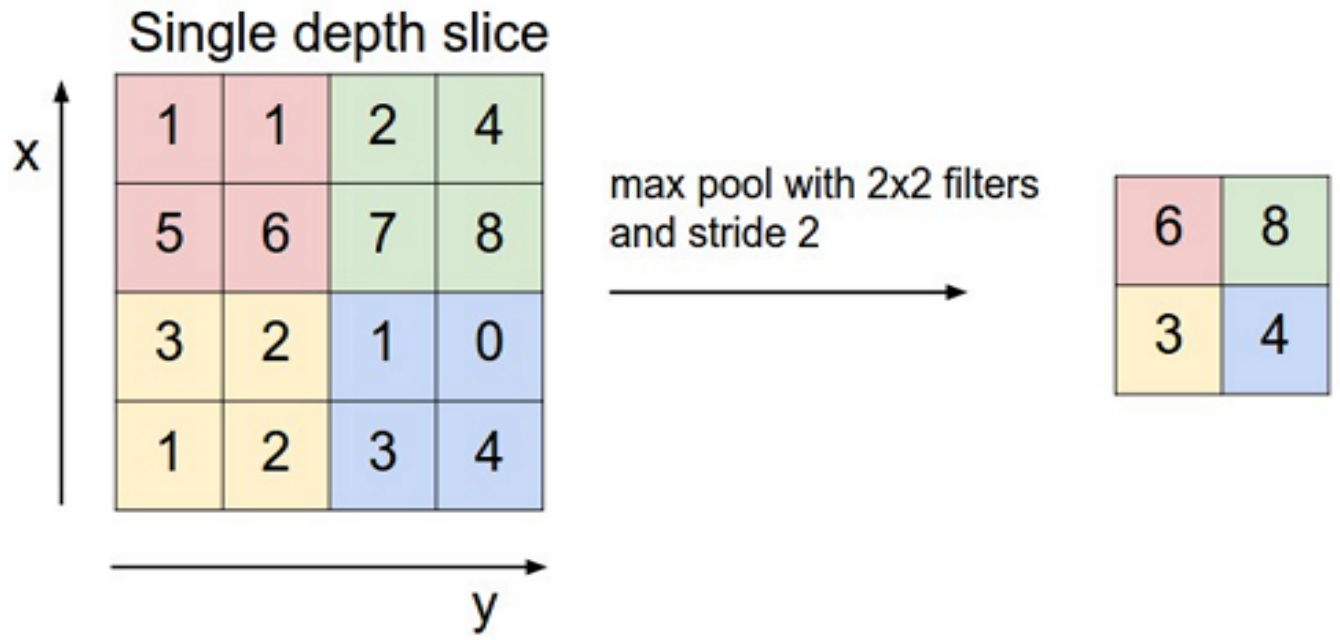
Pooling

Проблемы использования большего количества сверток:

- Размер изображения после свертки такой же или почти такой же, как был изначально
- Размерность не уменьшается, а еще фильтров будет много – значит только увеличивается
- Получается очень много параметров – есть риск не получить никакой обобщающей способности у алгоритма

Свёрточные сети

Pooling



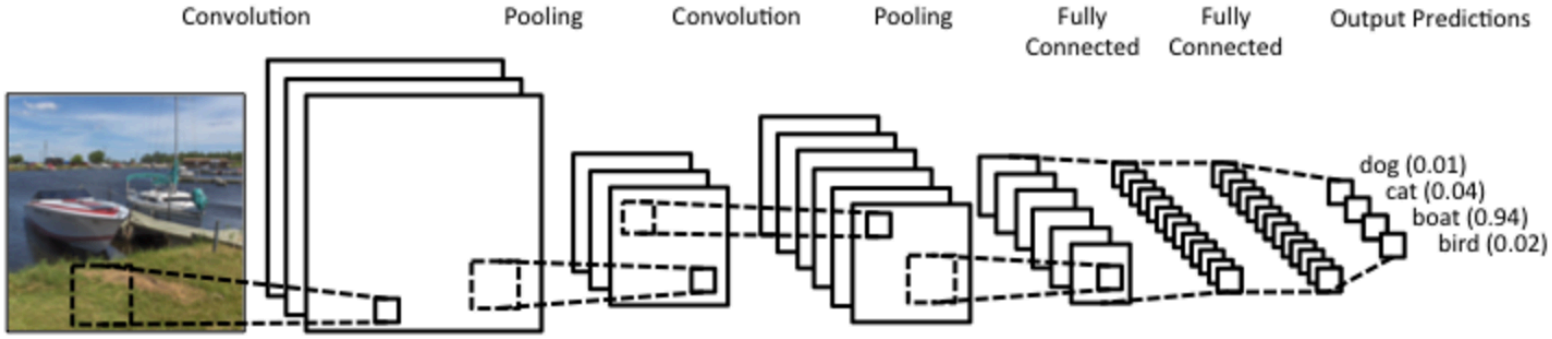
На практике используют

- Average Pooling
- Max Pooling

Credits to http://cs231n.stanford.edu/slides/

Свёрточные сети

Сверточная нейронная сеть



Credits to http://cs231n.stanford.edu/slides/

Свёрточные сети

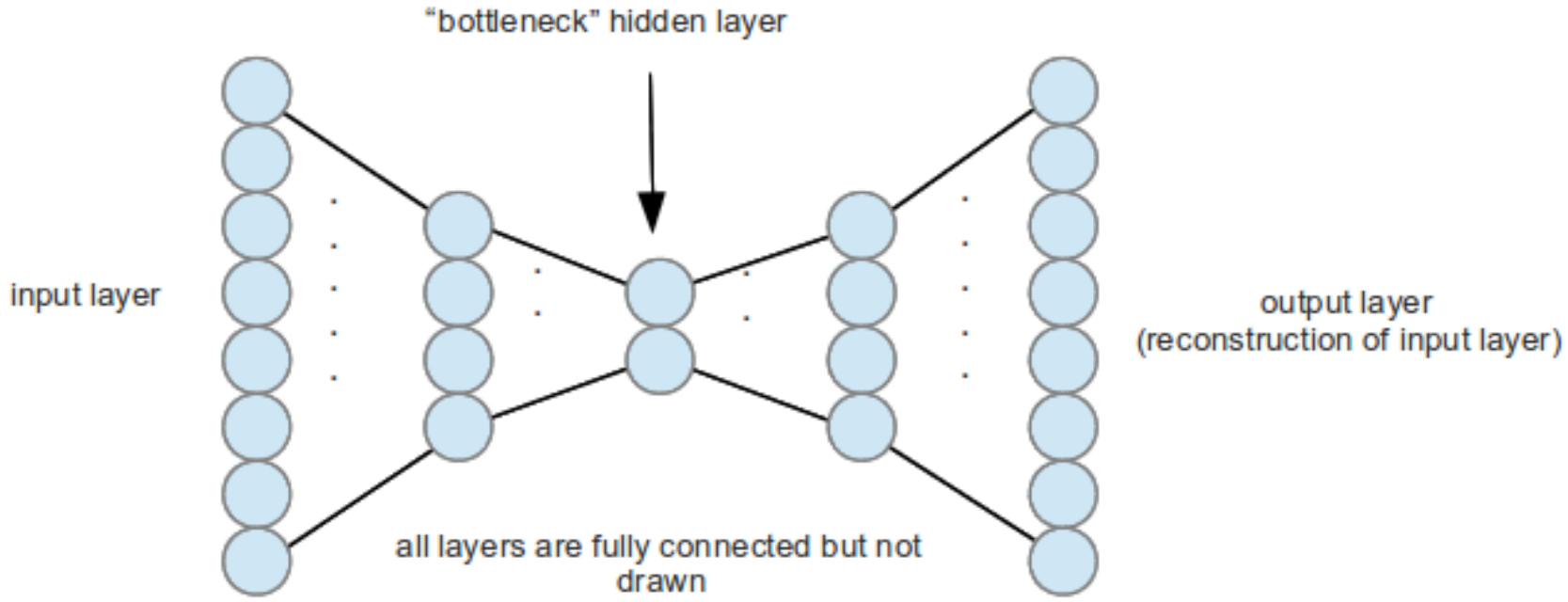
Пример: сеть GoogLeNet

Convolution
Pooling
Softmax
Other

Автоэнкодеры

Автоэнкодеры

1. Построим первые слои нейросети так, чтобы в них было все меньше и меньше нейронов
2. Далее зеркально отразим слои

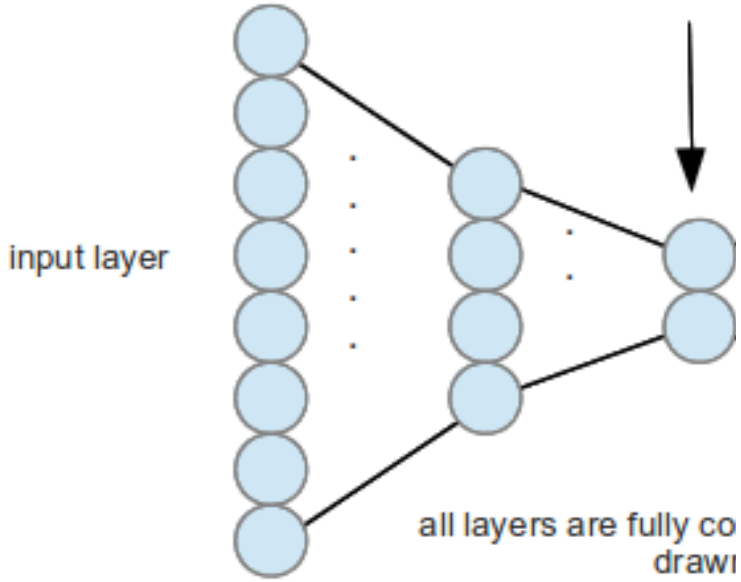


3. Обучим нейросеть восстанавливать на выходе то же, что было на входе.

Автоэнкодеры

Автоэнкодеры

4. «Срежем» всё после самой «узкой» части и будем использовать сеть для снижения размерности



Автоэнкодеры

Автоэнкодеры

Виды автоенкодеров:

- Sparse autoencoders
- Denoising autoencoders
- Convolutional autoencoders

Автоэнкодеры

Автоэнкодеры

Как использовать полученные признаки?

- Обучать другие алгоритмы на них
- Обучать модель на них и исходных признаках
- Использовать автоэнкодеры для предобучения нейросетей, чтобы потом обучение сходилось к лучшему решению

Рекуррентные нейронные сети

Рекуррентные нейронные сети

При традиционном подходе к обучению нейронных сетей предполагается, что все входы и выходы независимы.

Данное предположение работает не всегда:

- NLP: мы хотим предсказать следующее слово предложения по k предыдущим
- Timeseries: мы хоти продолжить временной ряд

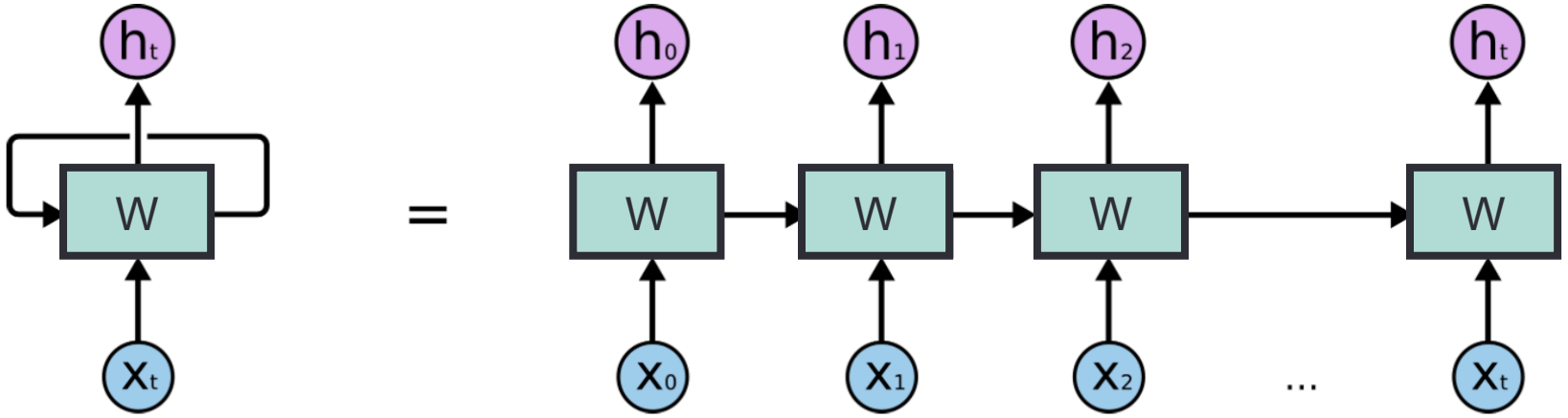
Существует много задач, где важно учитывать предыдущие входы. В терминах обучения сети: учитывать предыдущие состояния сети для вычисления текущего состояния.

Рекуррентные нейронные сети

Рекуррентные нейронные сети

- Нейронная сеть преобразует слой t в слой t +1 по правилу $h_{t+1} = \sigma_t(W_t h_t + b_t)$
- Получается, что нейросеть - это суперпозиция преобразований вида $\mathrm{a(x)} = \sigma_t(W_t(\sigma_{t-1}(W_{t-1} \ldots + b_{t-1})) + b_t)$

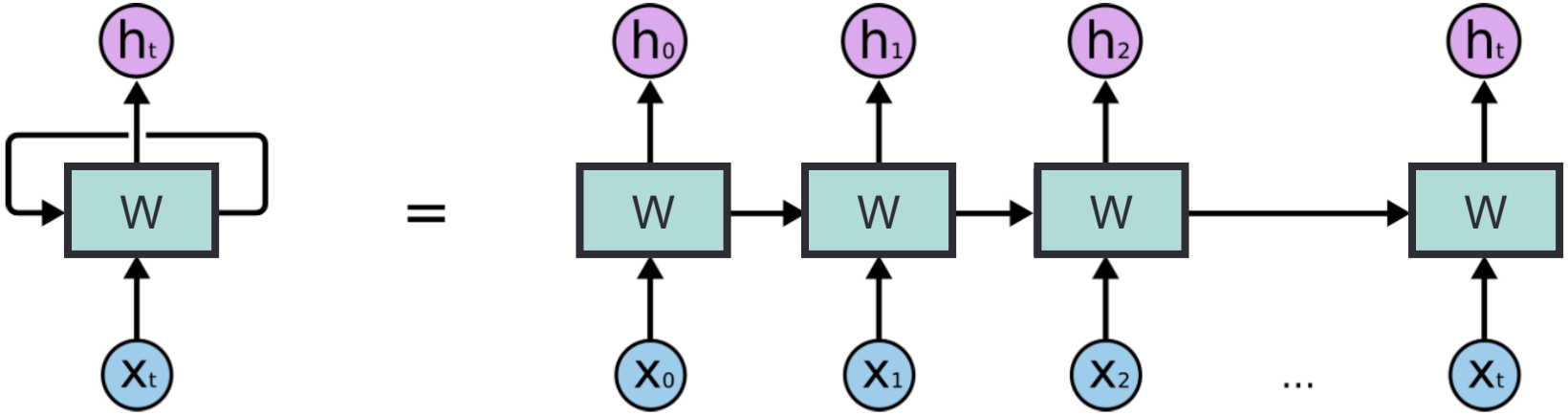
Идея: на каждом шаге обучения t будем обновлять посчитанные на предыдущем шаге веса W_t и b_t скрытого нейронного слоя



Рекуррентные нейронные сети

Рекуррентные нейронные сети

Идея: на каждом шаге обучения t будем обновлять посчитанные на предыдущем шаге веса W_t и b_t скрытого нейронного слоя



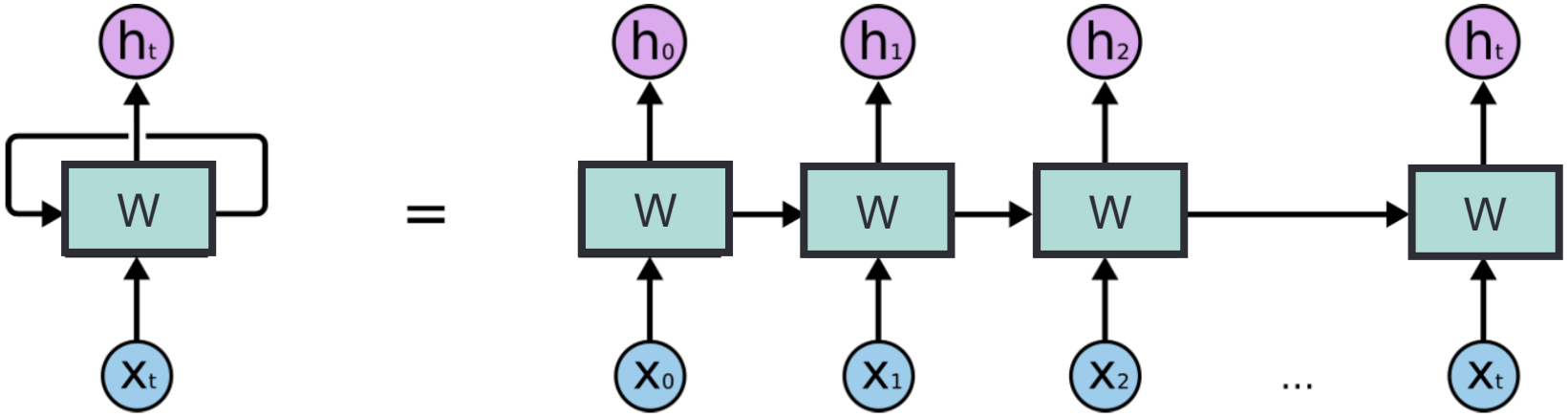
Тогда сеть будет задана следующим образом:

$h_{t+1} = \sigma_t(W_{hh}h_t + W_{hx}x_t + b_h)$, где

- Матрица W_{hh} определяет вклад вектора *h* с предыдущего шага в новый;
- Матрица W_{hx} определяет вклад нового объекта *x*.

Рекуррентные нейронные сети

Рекуррентные нейронные сети



Тогда сеть будет задана следующим образом:

$h_{t+1} = \sigma_t(W_{hh}h_t + W_{hx}x_t + b_h)$, где

- Матрица W_{hh} определяет вклад вектора *h* с предыдущего шага в новый;
- Матрица W_{hx} определяет вклад нового объекта *x*.

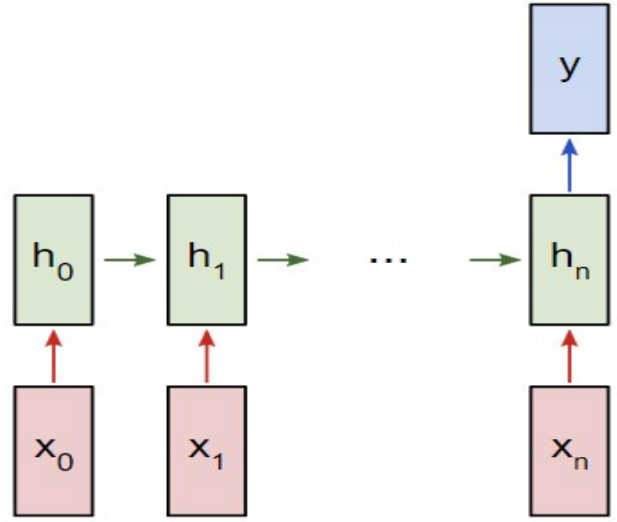
Осталось заметить, что на последнем слое нам нужно получить вектор ответов y, поэтому нужно добавить слой для вычисления выходного значения: $a(x) = \sigma_y(W_{hy}h_n + b_y)$

Рекуррентные нейронные сети

Рекуррентные нейронные сети

Таким образом простая рекуррентная сеть задается:

$$h_{t+1} = \sigma_t(W_{hh}h_t + W_{hx}x_t + b_h) \text{ и } a(x) = \sigma_y\left(W_{hy}h_n + b_y\right)$$



Здесь:

- Матрица W_{hh} определяет вклад вектора *h* с предыдущего шага в новый;
- Матрица W_{hx} определяет вклад нового объекта *x;*
- Матрица W_{hy} и b_y позволяют вычислить выходное значение.

Credits to https://python-scripts.com/wp-content/uploads/2019/10/rnn-004.jpg

Рекуррентные нейронные сети

Рекуррентные нейронные сети

Простые нейронные сети можно усложнить для получения решений с лучшими свойствами (в частности более длительной памятью):

- LSTM (Long short-term memory)
- GRU (Gated Recurrent Unit)

Модель нейронной сети

Обучение нейронных сетей

- Модель персептрона и нейронная сеть известны давно
- Применение подхода back propagation позволило считать градиенты быстрее, но всё равно не достаточно быстро для обучения сетей с большим количеством скрытых слоев
- Современные вычислительные мощности и матричная модель вычислений позволили обучать нейросети с большим количеством слоев достаточно быстро
- Появились достаточно объемные наборы данных, для обучения моделей с большим количеством параметров: Imagenet, CIFAR-100
- Сегодня появляются новые архитектуры нейросетей, позволяющие более качественно решать соответствующие прикладные задачи

Практические рекомендации

Как обучить нейронную сеть

Как обучить нейронную сеть

Библиотеки для обучения моделей:

- **Tensorflow**
- **Keras** (входит в Tensorflow, начиная с версии 2)
- **PyTorch**
- Theano
- Lasagne
- Nolearn
- Caffe

Как обучить нейронную сеть

Как обучить нейронную сеть

Для обучения модели потребуется:

1. Установить одну из библиотек на выбор
2. Найти любой пример с нейросетью на Keras/PyTorch/Tensorflow и воспроизвести у себя
3. Пробовать использовать разные архитектуры нейросетей (полносвязные, сверточные, рекуррентные, автоэнкодеры)

Как обучить нейронную сеть

Как обучить нейронную сеть

Будет полезно подробнее изучить Deep Learning.

- Стенфордский курс по computer vision: http://cs231n.stanford.edu/
- Курс по NLP от deeplearning.ai https://www.deeplearning.ai/natural-language-processing-specialization/
- Обучение и деплой моделей: https://course.fullstackdeeplearning.com/

Курсов много, ищите курс, который вам понравится.

Машинное обучение: нейронные сети и глубокое обучение (введение)

Спасибо!

Эмели Драль