

Практический анализ данных и машинное обучение: искусственные нейронные сети

Ульянкин Филипп

9 октября 2018 г.

Введение в Нейросети

Я

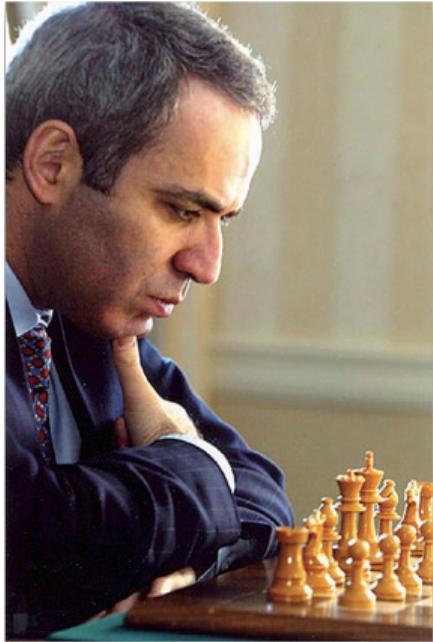
Agenda

- Немного истории
- Всё, что вы хотели знать о градиентном спуске, но боялись спросить
- Перцепtron
- Алгоритм обратного распространения ошибки, учим нейросеть руками
- Нейросети — конструктор Lego

Немного истории







Garry Kasparov



Garry Kasparov



Deepblue

the new york times bestseller

шум сигнал и шум
сигнал и шум сигнал
и шум сигнал и шу
почему одни сигна
прогнозы шум сиг
сбываются, сигна
а другие - нет шу
сигнал и шум сигнал
нейт сильвер шум

Сигнал – это правда, а шум – это то, что
отвлекает нас от правды.

нейт сильвер

сигнал и шум



- Шашки полностью решены в 2007 году
- Шахматы: впервые компьютер побеждает человека в феврале 1996 года
- Шахматы: в ноябре 2005 человек в последний раз побеждает у компьютера
- Го: последний оплот человечества до апреля 2016 года

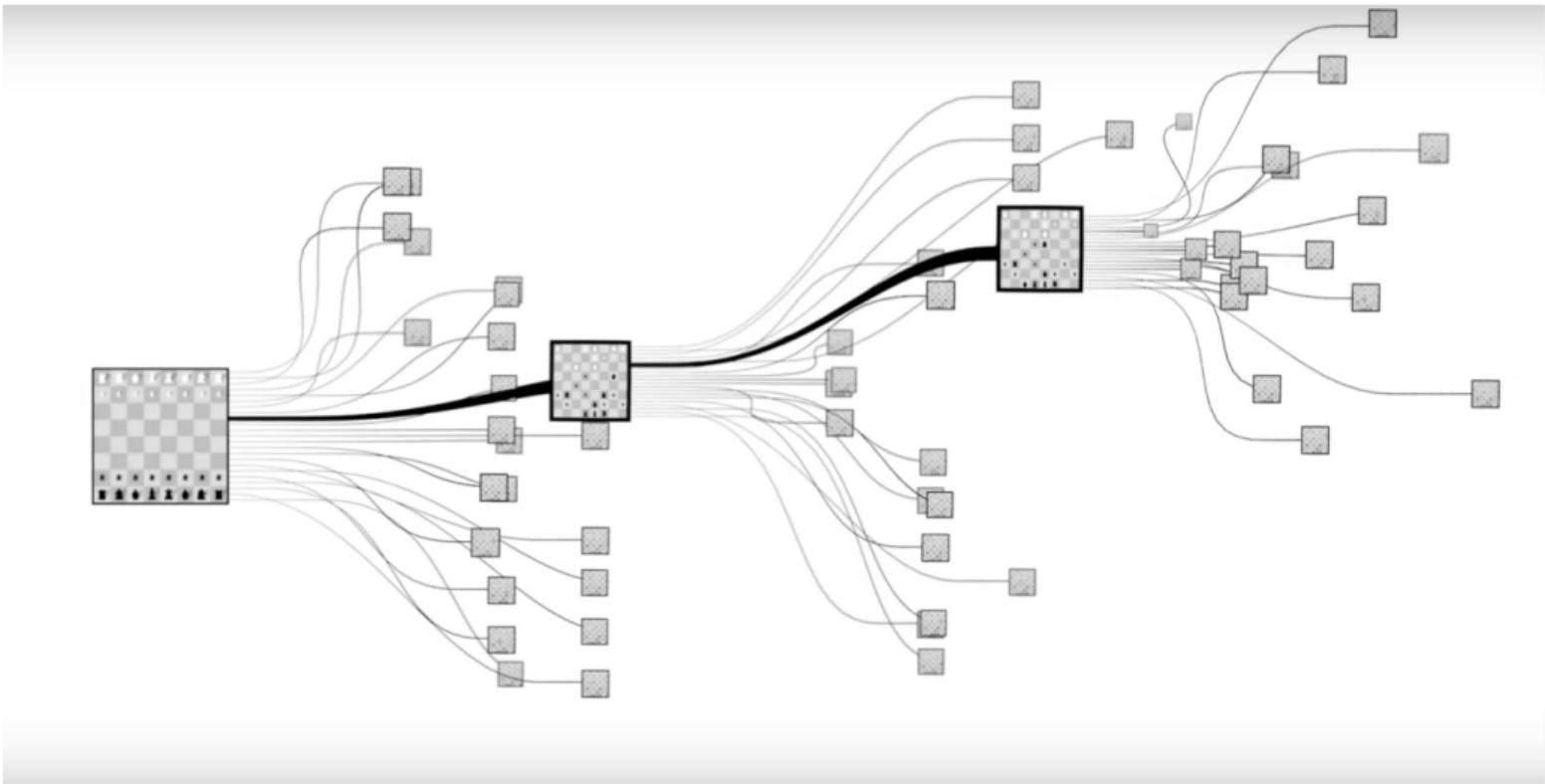
GO пал со счётом 4 : 1

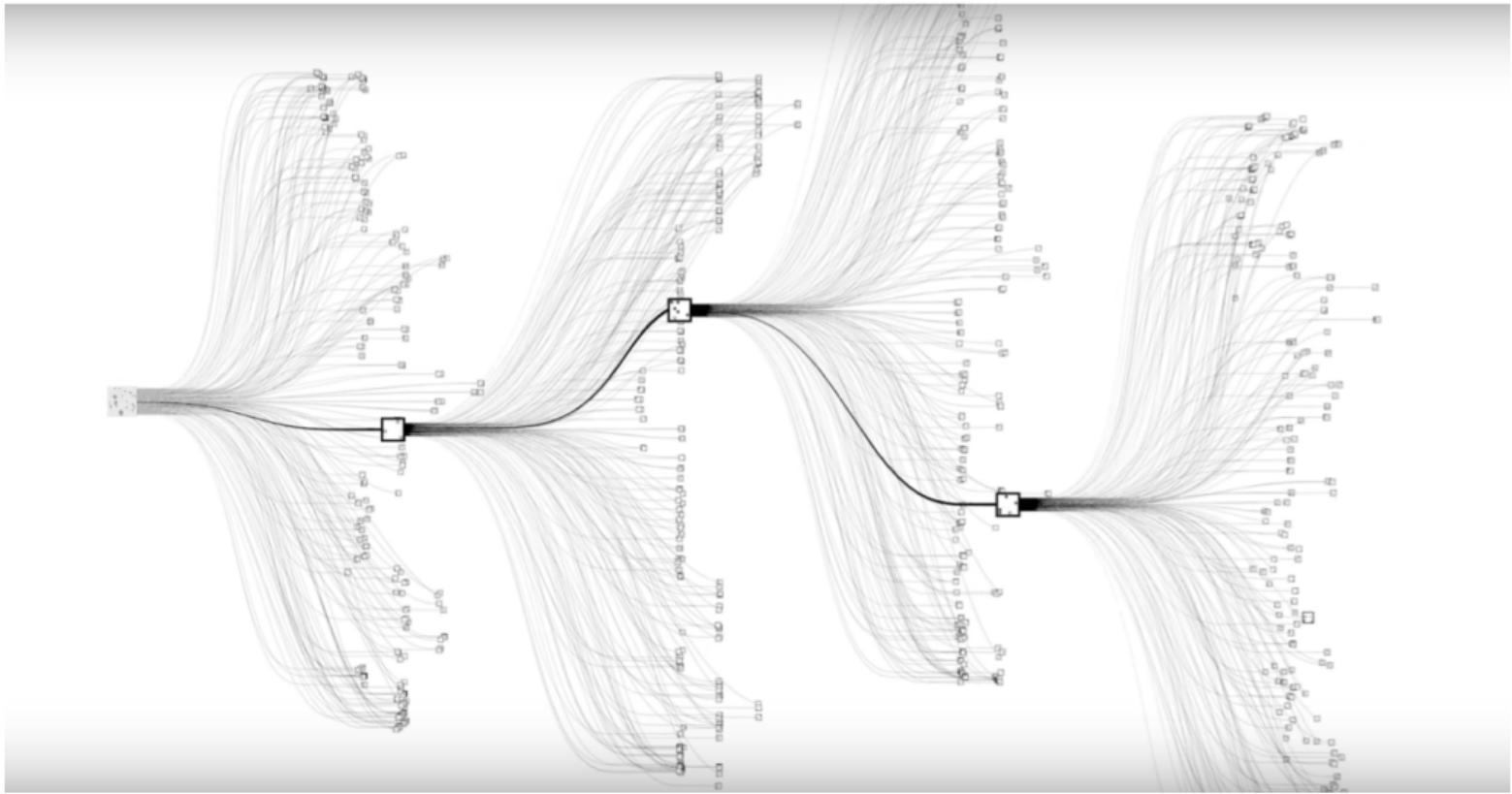


AlphaGo by Google



Lee Sedol





О том как всё начиналось

- Середина 1950-х — перцепtron Розенблата

О том как всё начиналось

- Середина 1950-х — перцепtron Розенблата
- 1956 год — Дартмутский семинар, безграничный оптимизм

О том как всё начиналось

- Середина 1950-х — перцепtron Розенблата
- 1956 год — Дартмутский семинар, безграничный оптимизм
- Конец 1960-х — провал крупного проекта по машинному переводу, первая зима

О том как всё начиналось

- Середина 1950-х — перцепtron Розенблата
- 1956 год — Дартмутский семинар, безграничный оптимизм
- Конец 1960-х — провал крупного проекта по машинному переводу, первая зима
- Начало 1980-х — разработан backpropagation, появление огромного числа архитектур, ренессанс нейросетей

О том как всё начиналось

- Середина 1950-х — перцепtron Розенблата
- 1956 год — Дартмутский семинар, безграничный оптимизм
- Конец 1960-х — провал крупного проекта по машинному переводу, первая зима
- Начало 1980-х — разработан backpropagation, появление огромного числа архитектур, ренессанс нейросетей
- Начало 1990-х — завышенные ожидания снова не оправданы, вторая зима

О том как всё начиналось

- Середина 2000-х – третий и единственный успешный ренессанс нейронных сеток, люди научились обучать глубокие нейросети

О том как всё начиналось

- Середина 2000-х – третий и единственный успешный ренессанс нейронных сеток, люди научились обучать глубокие нейросети
- Начиная с 2014 года несколько архитектур друг за другом рвут ImageNet

О том как всё начиналось

- Середина 2000-х – третий и единственный успешный ренессанс нейронных сеток, люди научились обучать глубокие нейросети
- Начиная с 2014 года несколько архитектур друг за другом рвут ImageNet
- Человечество копит в течение 2000-х годов огромное количество данных, а мощности компьютеров выходят на новые рубежи

Линейные модели

Как обучать?

- «Тренировка» — поиск оптимальных W и b
- «Оптимальных» — минимизирующих какой-то функционал
- Какими бывают функционалы: MSE, MAE, Logloss
- Как оптимизировать: градиентный спуск!

Градиентный спуск

Градиентный спуск (GD)

Проблема оптимизации:

$$L(w) = \sum_{i=1}^n L(w, x_i, y_i) \rightarrow \min_w$$

Инициализация w_0

while True:

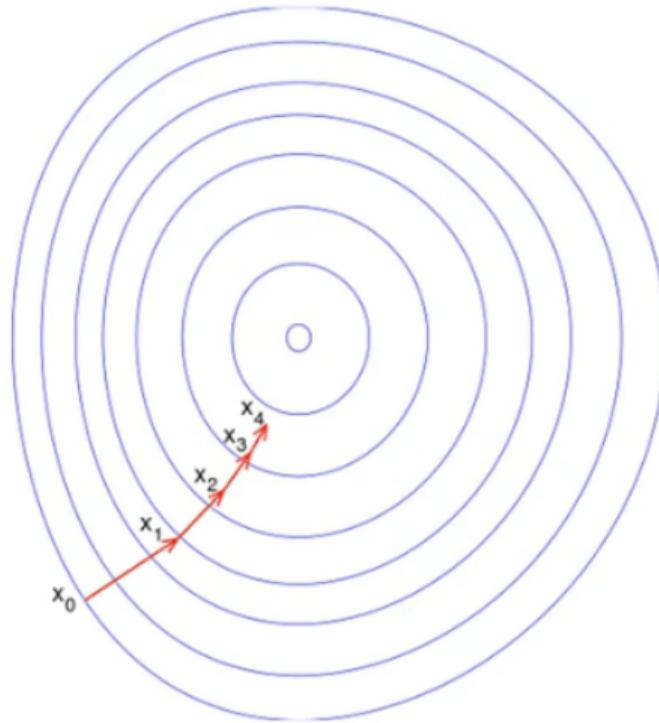
$$g_t = \frac{1}{n} \sum_{i=1}^n \nabla L(w, x_i, y_i)$$

$$w_t = w_{t-1} - \eta_t \cdot g_t$$

if $\|w_t - w_{t-1}\| < \varepsilon$:

break

Градиентный спуск



Стochastic gradient descent (SGD)

Проблема оптимизации:

$$L(w) = \sum_{i=1}^n L(w, x_i, y_i) \rightarrow \min_w$$

Инициализация w_0

while True:

рандомно выбрали i

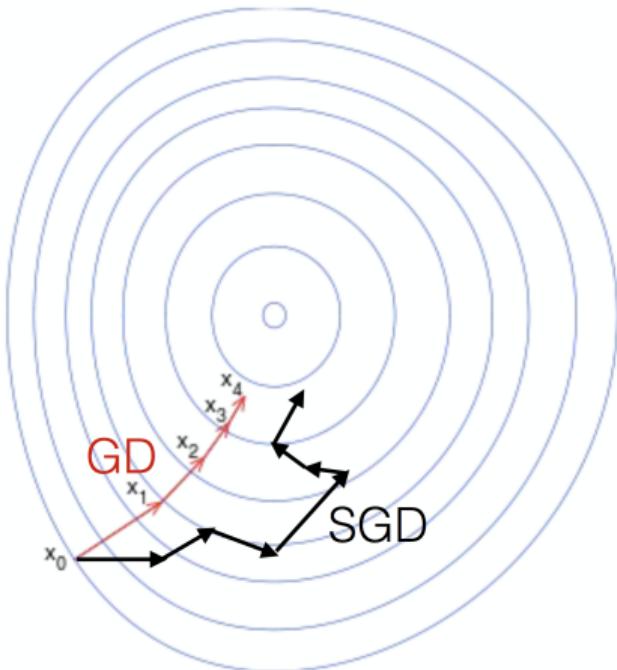
$$g_t = \nabla L(w_{t-1}, x_i, y_i)$$

$$w_t = w_{t-1} - \eta_t \cdot g_t$$

if $\|w_t - w_{t-1}\| < \varepsilon$:

break

Стохастический градиентный спуск (SGD)



- И для GD и для SGD нет гарантий глобального минимума, сходимости
- SGD быстрее, на каждой итерации используется только одно наблюдение
- Для SGD спуск очень зашумлён
- GD: $O(n)$, SGD: $O(1)$
- Скорость обучения надо подбирать аккуратно, если она будет большой, мы можем скакать вокруг минимума, если маленькой - вечно ползти к нему

Mini-bath SGD

Проблема оптимизации:

$$L(w) = \sum_{i=1}^n L(w, x_i, y_i) \rightarrow \min_w$$

Инициализация w_0

while True:

рандомно выбрали $m < n$ индексов

$$g_t = \frac{1}{m} \sum_{i=1}^m \nabla L(w, x_i, y_i)$$

$$w_t = w_{t-1} - \eta_t \cdot g_t$$

if $\|w_t - w_{t-1}\| < \varepsilon$:
break

Momentum SGD

Мы считали на каждом шаге градиент по формуле

$$g_t = \frac{1}{m} \sum_{i=1}^m \nabla L(w, x_i, y_i).$$

После шага мы забывали его. **А давайте попробуем помнить направление:**

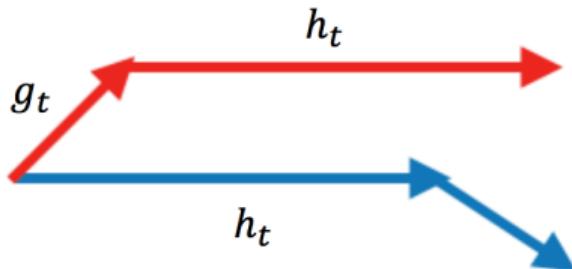
$$\begin{aligned} h_t &= \alpha \cdot h_{t-1} + \eta_t \cdot g_t \\ w_t &= w_{t-1} - h_t \end{aligned}$$

- Движение поддерживается в том же направлении, что и на предыдущем шаге
- Нет резких изменений направления движения.
- Обычно $\alpha = 0.9$.

Momentum SGD

- Бежим с горки и всё больше ускоряемся в том направлении, в котором были направлены сразу несколько предыдущих градиентов, но при этом движемся медленно там, где градиент постоянно меняется
- Хотелось бы не просто бежать с горы, но и хотя бы на полшага смотреть себе под ноги, чтобы внезапно не споткнуться \Rightarrow давайте смотреть на градиент в будущей точке
- Согласно методу моментов αh_{t-1} точно будет использоваться при шаге, давайте искать $\nabla L(w_{t-1} - \alpha \cdot h_{t-1})$.

Nesterov Momentum SGD



$$h_t = \alpha \cdot h_{t-1} + \eta_t \nabla L(w_{t-1} - \alpha \cdot h_{t-1})$$

$$w_t = w_{t-1} - h_t$$

Momentum SGD

- Может сложиться, что некоторые веса уже близки к своим локальным минимумам, по этим координатам надо двигаться медленнее, а по другим быстрее \Rightarrow адаптивные методы градиентного спуска
- Шаг изменения должен быть меньше у тех параметров, которые в большей степени варьируются в данных, и больше у тех, которые менее изменчивы

AdaGrad

$$G_t^j = G_{t-1}^j + g_{tj}^2$$
$$w_t^j = w_{t-1}^j - \frac{\eta_t}{\sqrt{G_t^j + \varepsilon}} \cdot g_t^j$$

- g_t^j — градиент по j -ому параметру
- своя скорость обучения для каждого параметра
- обычно $\eta_t = 0.01$, этот параметр не имеет значения
- G_t^j всегда увеличивается, из-за этого обучение может рано останавливаться

RMSprop

$$G_t^j = \alpha \cdot G_{t-1}^j + (1 - \alpha) \cdot g_{tj}^2$$
$$w_t^j = w_{t-1}^j - \frac{\eta_t}{\sqrt{G_t^j + \varepsilon}} \cdot g_t^j$$

- Обычно $\alpha = 0.9$
- Скорость обучения адаптируется к последнему сделанному шагу

Adam

$$\begin{aligned} h_t^j &= \beta_1 \cdot h_{t-1}^j + (1 - \beta_1) \cdot g_{tj} \\ G_t^j &= \beta_2 \cdot G_{t-1}^j + (1 - \beta_2) \cdot g_{tj}^2 \\ w_t^j &= w_{t-1}^j - \frac{\eta_t}{\sqrt{G_t^j + \varepsilon}} \cdot h_t^j \end{aligned}$$

Комбинируем momentum и индивидуальные скорости обучения

Подведём итоги

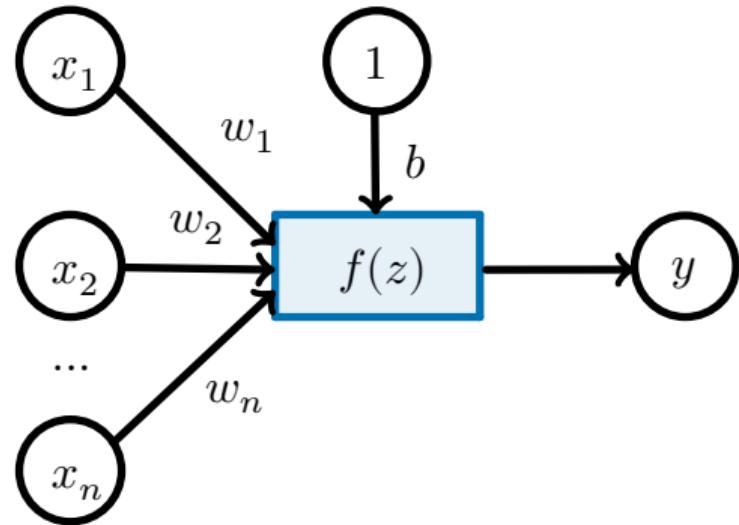
- Momentum SGD сохраняет направление шага и позволяет добиваться более быстрой сходимости
- Адаптивные методы позволяют находить индивидуальную скорость обучения для каждого параметра
- Adam комбинирует в себе оба подхода

Перцептрон

Перцепtron

- Перцепtron – древняя штука, придуманная Розенблатом в 1950-е годы.

Перцептрон

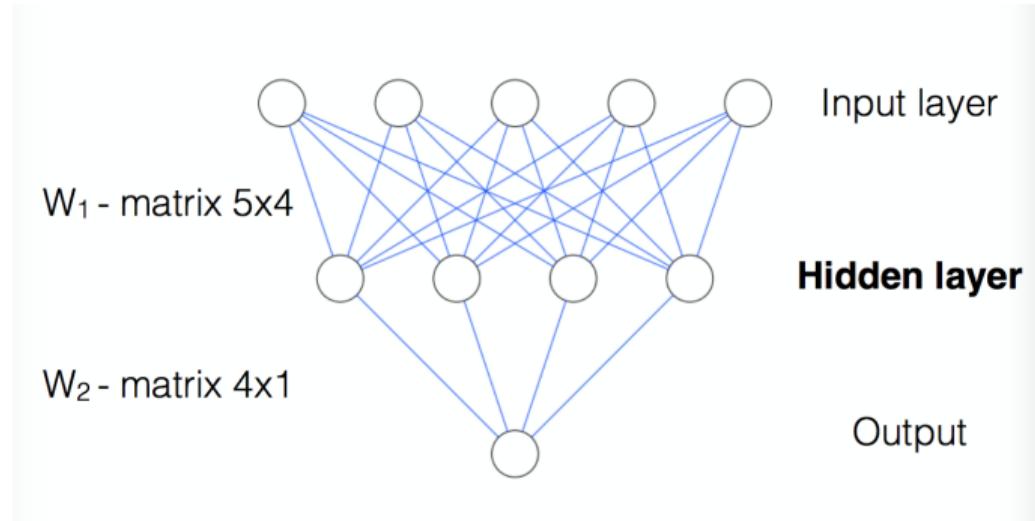


$$y = f(b + w_1 \cdot x_1 + \dots + w_n \cdot x_n) = f(X \cdot W_1)$$
$$h = X \cdot W_1$$

Некоторые модели — частный случай перцептрона

- Для регрессии возьмём $f(z) = z$.
- Для логистической регрессии возьмём $f(z) = \frac{1}{1+e^{-z}}$.

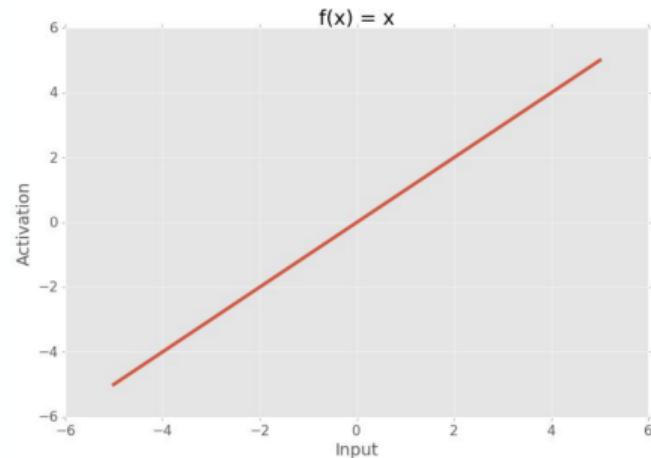
Скрытый слой



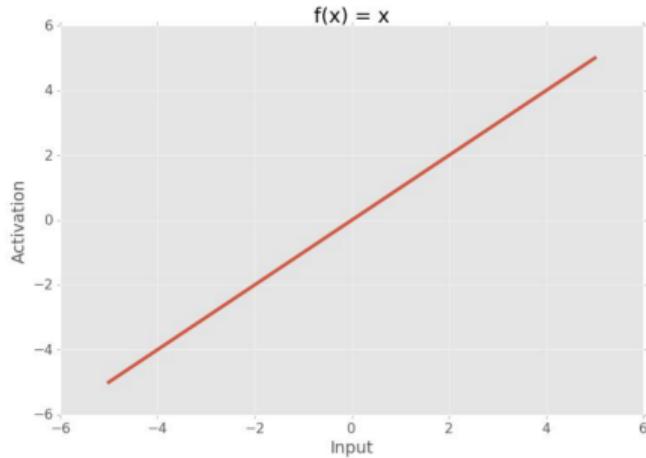
$$h = f(X \cdot W_1)$$

$$\hat{y} = h \cdot W_2$$

Линейная активация



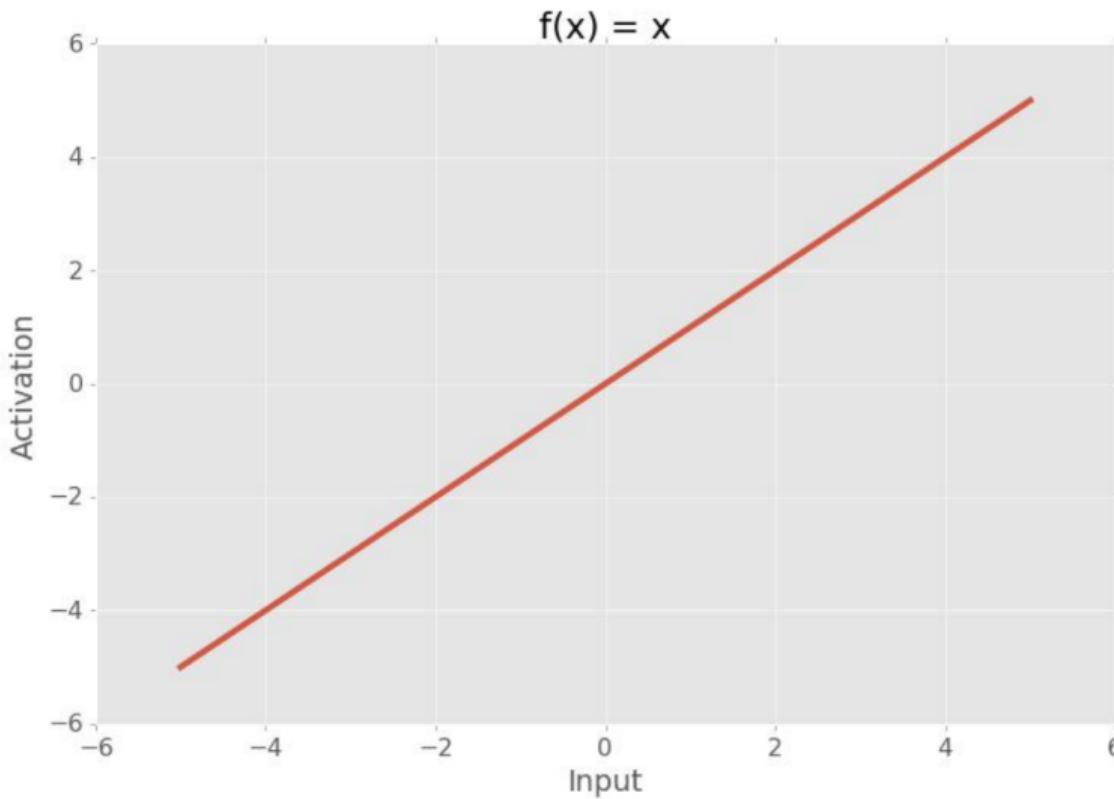
Линейная активация



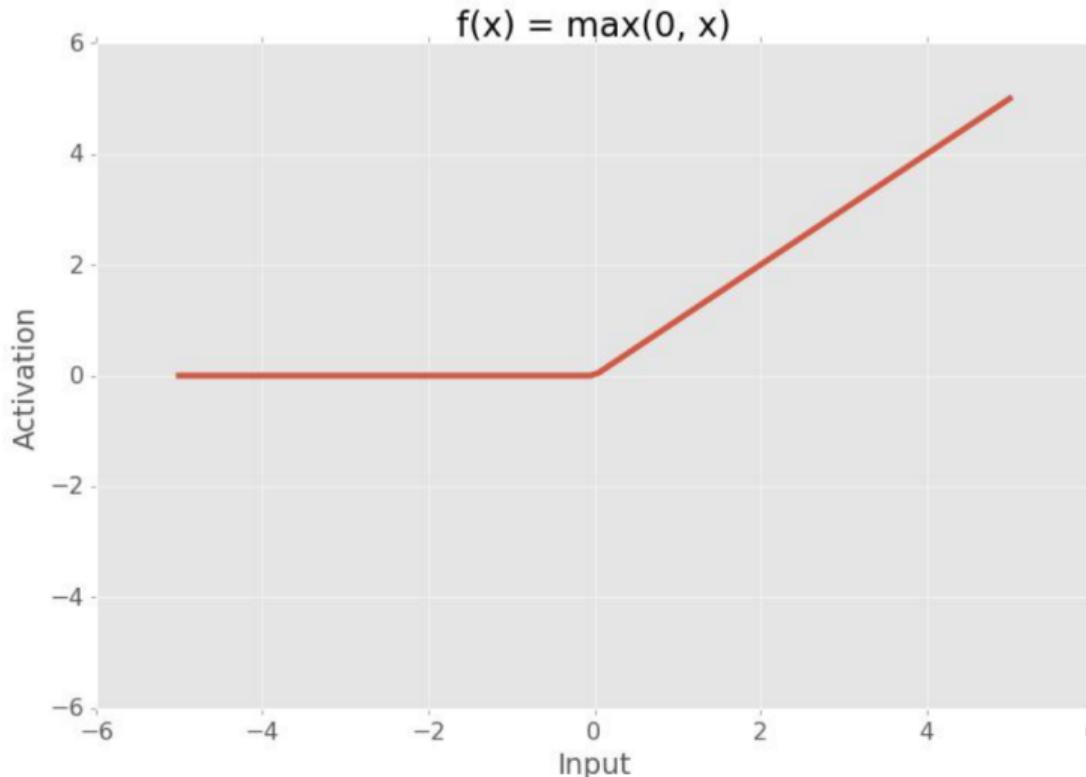
$$\hat{y} = h \cdot W_2 = X \cdot W_1 \cdot W_2 = X \cdot A$$

- После линейной активации выход снова линейный

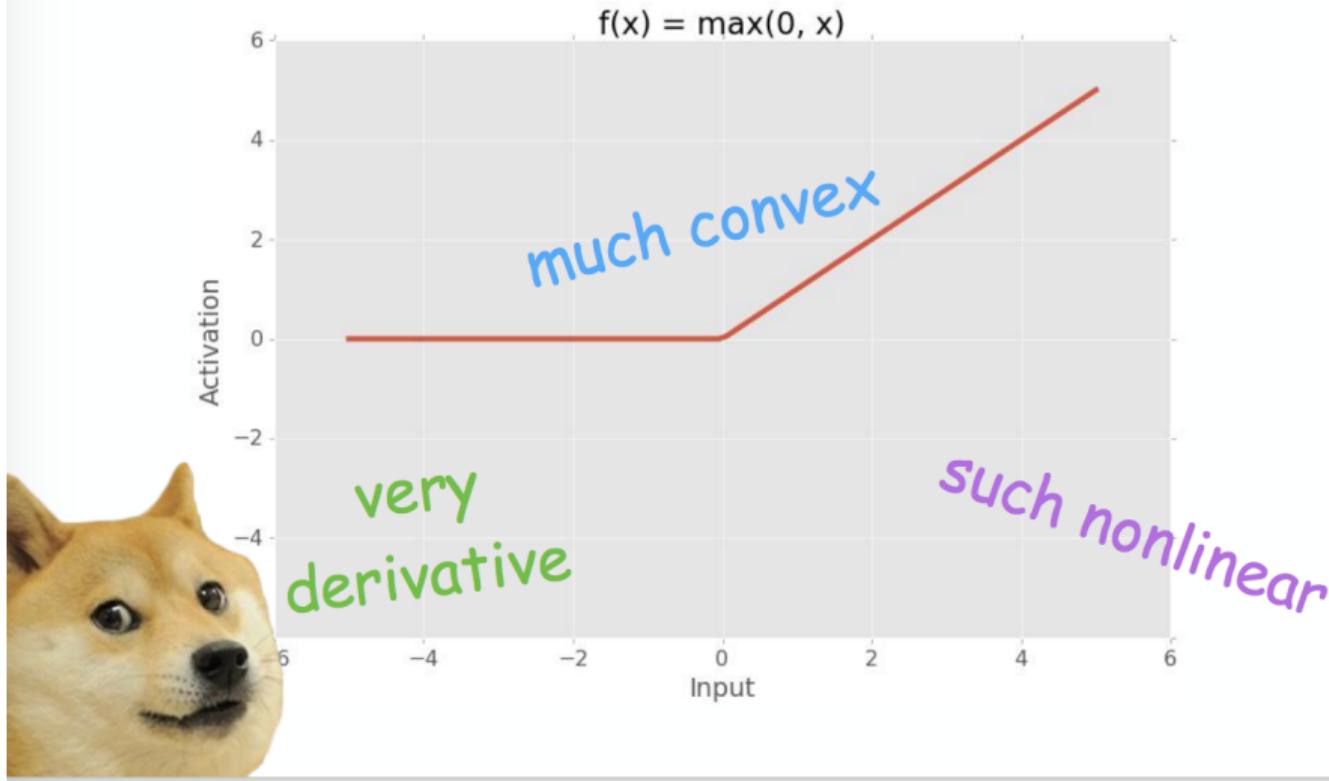
От линейной активации ...



... к нелинейной



ReLU



Как обучить перцептрон?

$$L(W_1, W_2) = \frac{1}{2} \cdot (y - f(X \cdot W_1) \cdot W_2)^2$$

Секрет успеха в умении брать производную и градиентном спуске.

$$f(g(x))' = f'(g(x)) \cdot g'(x)$$

$$\frac{\partial L}{\partial W_2} = -(y - f(X \cdot W_1) \cdot W_2) \cdot f'(X \cdot W_1) \cdot W_1$$

$$\frac{\partial L}{\partial W_1} = -(y - f(X \cdot W_1) \cdot W_2) \cdot W_2 f'(X \cdot W_1) \cdot W_1$$

The Perceptron Convergence Theorem (Rosenblat, 1965)

- Любая непрерывная и ограниченная функция может быть сколь угодно точно аппроксимирована нейронной сетью с одним скрытым слоем с нелинейной функцией активации нейрона.
- Любая функция может быть сколь угодно точно аппроксимирована нейронной сетью с двумя скрытыми слоями с нелинейной функцией активации нейрона.
- Что ещё можно пожелать?

The background of the image is a deep blue ocean. Sunlight filters down from the surface in bright, dappled rays, creating a play of light and shadow on the dark blue water. The surface is slightly textured with small waves.

Going Deeper

Мотивация

- Перцептрон может решить любую проблему, но это дорого
- Глубокие архитектуры часто позволяют выразить то же самое, приблизить те же функции гораздо более эффективно, чем неглубокие
- Каждый новый слой сетки будет работать всё с более сложными фичами

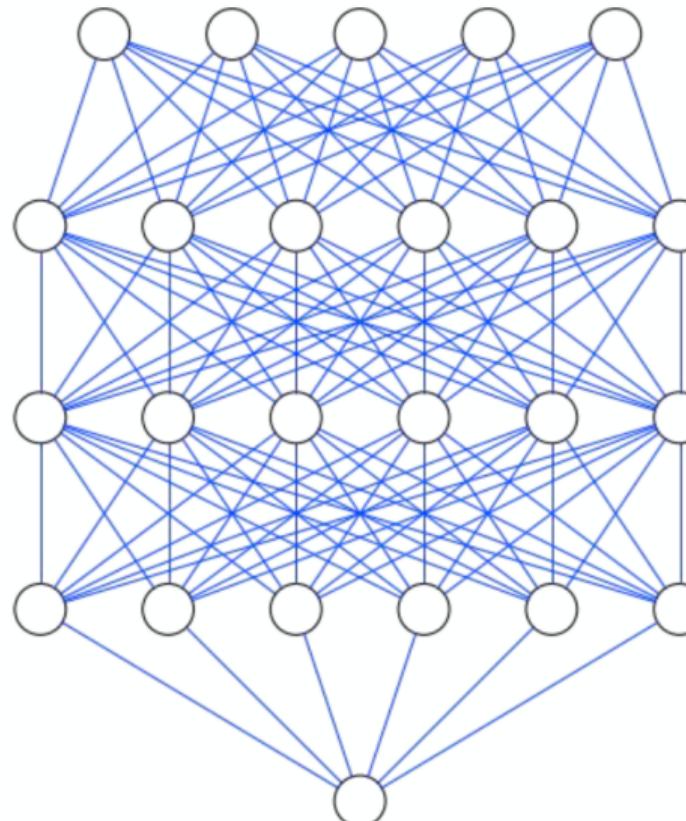
E.g.

Input features

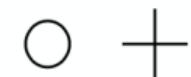
Complex features

Very
Complex features

Extremely
Complex features



Face recognition system



Как обучать?

- Прямое распространение ошибки (forward propagation):

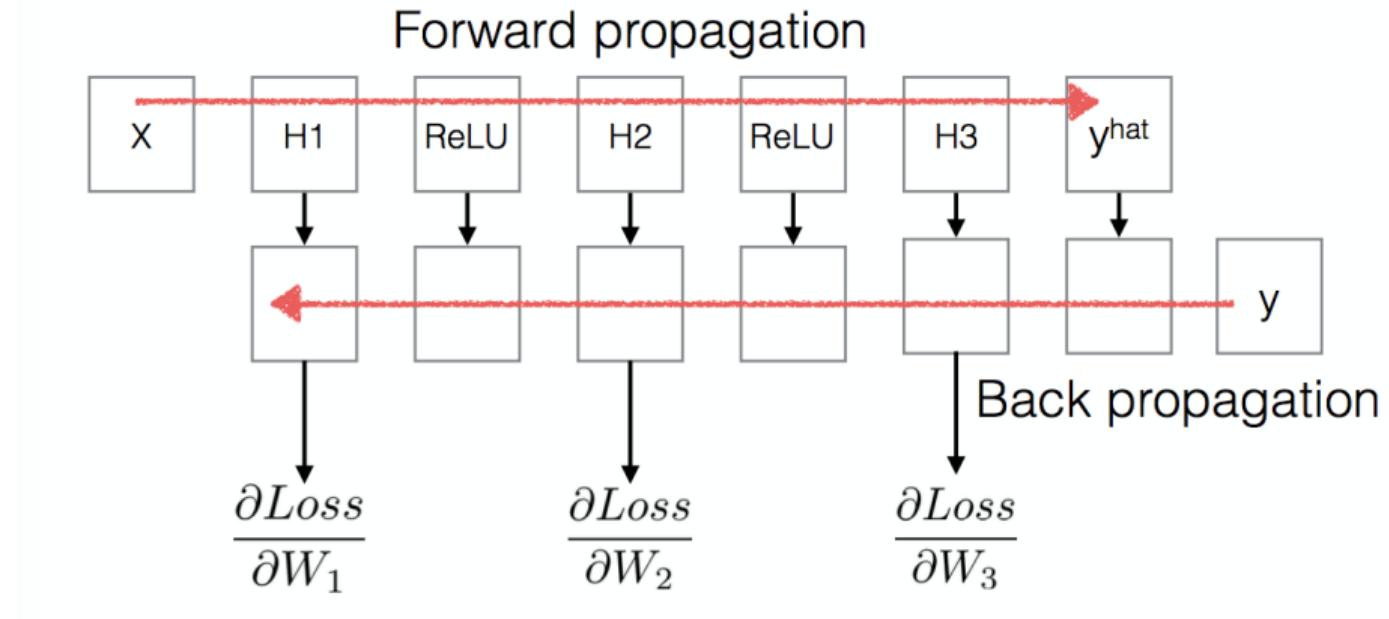
$$X \Rightarrow X \cdot W_1 \Rightarrow f(X \cdot W_1) \Rightarrow f(X \cdot W_1) \cdot W_2 \Rightarrow \dots \Rightarrow \hat{y}$$

- Считаем потери:

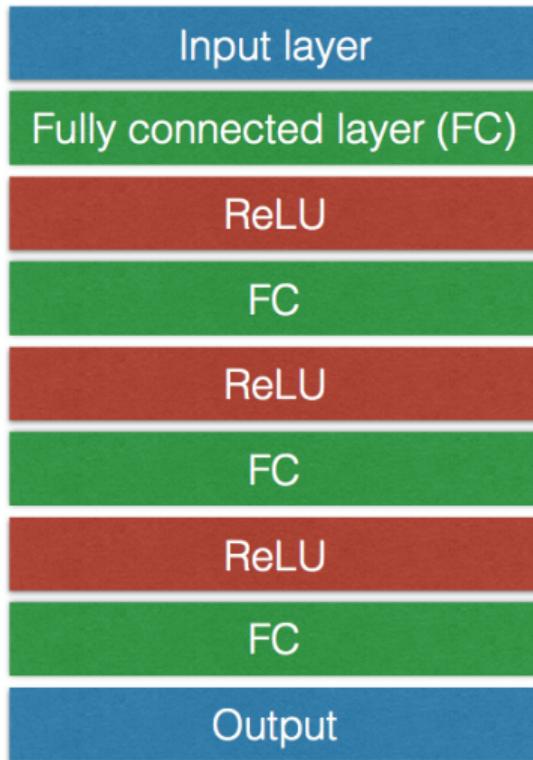
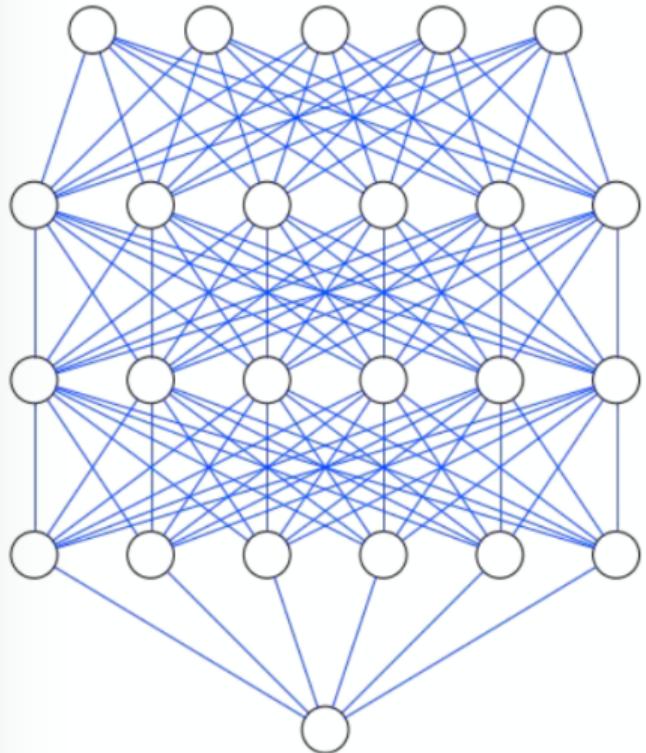
$$Loss = \frac{1}{2}(y - \hat{y})^2$$

- Ищем производные по цепному правилу ... но это довольно дорого
- До определённого момента люди не знали как легко обучать нейросети

Backpropagation



Lego



$XW + b$
 $\max(0, X)$
 $XW + b$

Учим нейросеть руками!