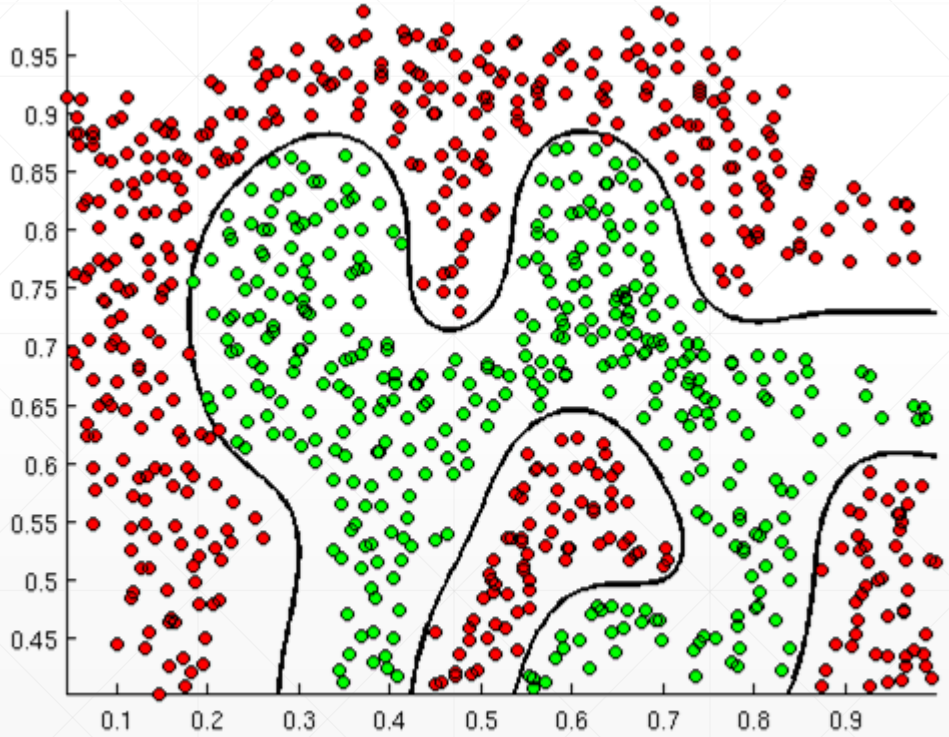
Введение в Deep Learning. «Копаем глубже»

Гончаров Павел kaliostrogoblin3@gmail.com
Нестереня Игорь nesterione@gmail.com

Предпосылки: нелинейность

Представим себе задачу классификации, где решением является **сложная нелинейная функция**. Для нее нужно найти гипотезу с полиномом высокой степени.



Теперь представим, что входной набор признаков – **изображения** 50x50 = 2500 признаков.

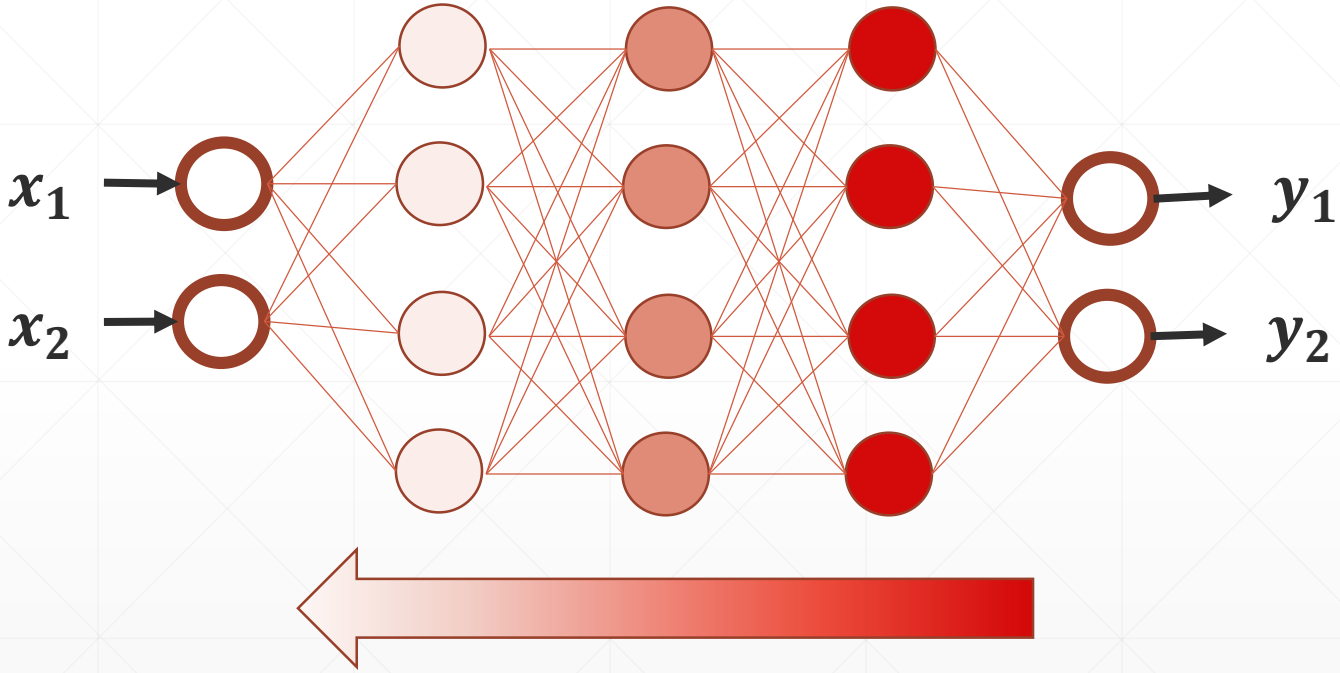
Если взять полином второй степени (квадраты исходных признаков), то количество входных значений ≈ 3 млн.

Выход – искусственные нейронные сети (ИНС), многослойный персептрон.

Обучение нейронной сети онлайн: http://playground.tensorflow.org/

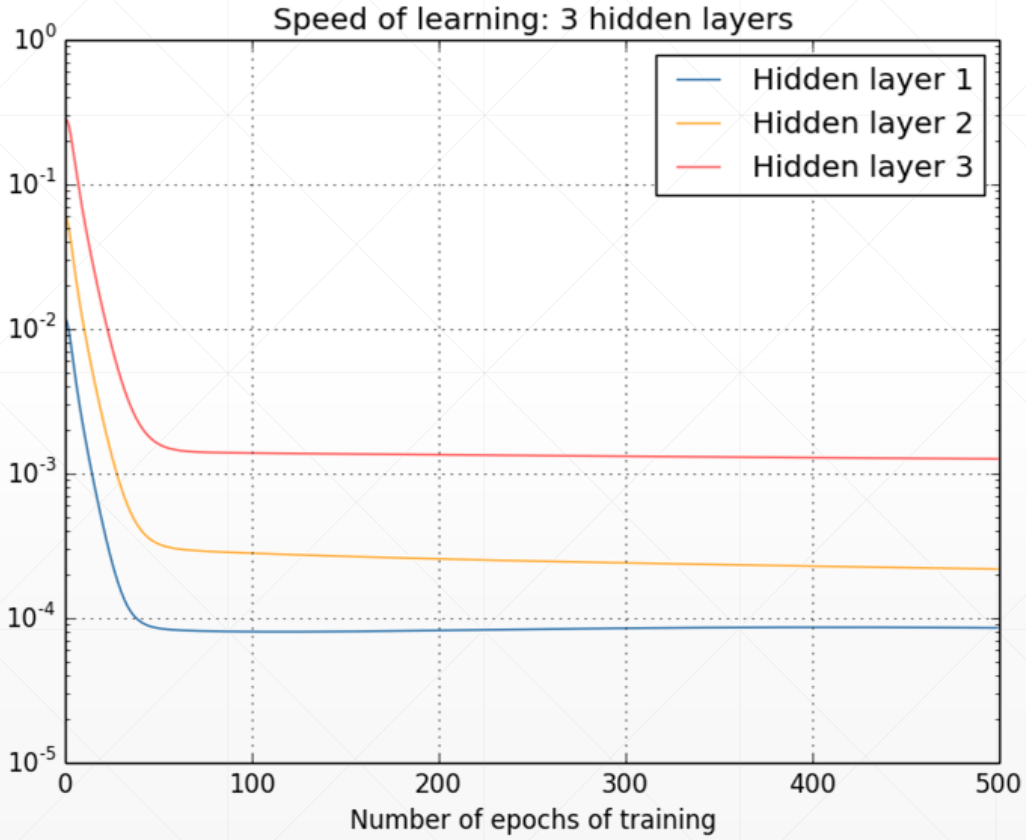
Проблемы: взрыв или затухание градиента

- **Первая проблема, которая возникает при обучении многослойной сети – это взрывной рост или затухание градиента («exploding and vanishing gradients»).**



Обратное распространение ошибки

Изменение цвета показывает затухание градиента



На графике видно, что чем ближе скрытый слой ко входному, тем медленнее он обучается, медленнее происходит корректировка его параметров.

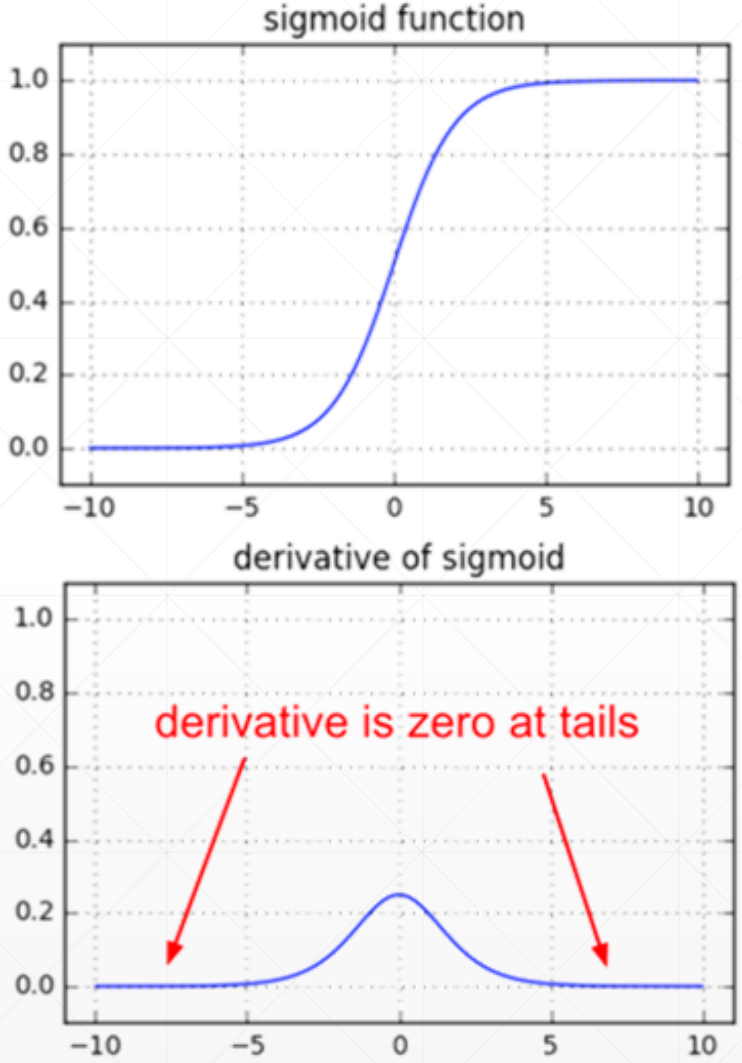
Взрывной или затухающий градиент

Обучение – это изменение параметров сети (весов). Дельта изменения весов первого слоя – это скорость обучения, умноженная на градиент ошибки слоя (**произведение всех градиентов предыдущих слоев**)

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$

Для такого случая проблема с градиентом зависит от начальной инициализации весов:

- при очень малых значениях происходит затухание
- при достаточно крупных будет происходить рост



Как решить проблему с градиентом?

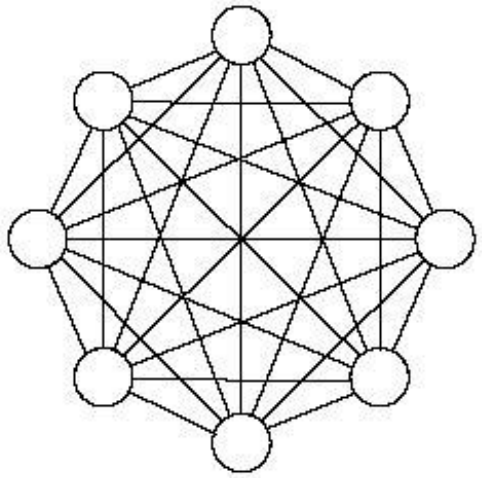
Существует множество трюков, которые помогут решить проблему с обучением многослойной сети. Yann LeCun в своей статье «Efficient BackProp» подробно описал пути решения – http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf.

Можно выделить три основных способа, которые могут помочь в задаче обучения многослойной сети:

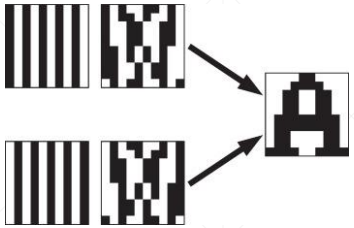
- **предобучение нейросети** для специальной настройки весов;
- специальная **нормализованная инициализация параметров**;
- **выбор другой функции активации**, более подходящей.

Раз обратное распространение ошибки так плохо работает, может есть другой способ обучения сети?

Полносвязные сети Хопфилда



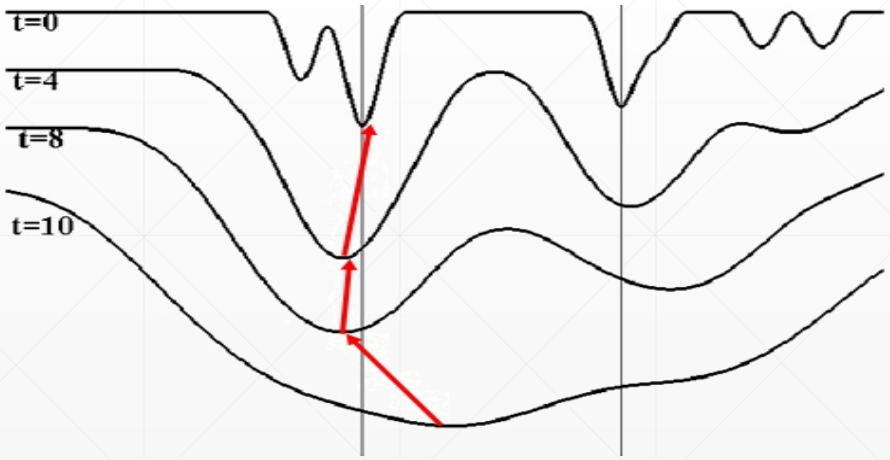
Это **полносвязная** сеть из **бинарных** нейронов si с **симметричной весовой матрицей** $w_{ij} = w_{ji}$, $w_{ii} = 0$. Эволюция ХНС приводит ее в некоторое состояние устойчивого равновесия. Функционал энергии сети (ошибки) – это билинейная функция Ляпунова:



$$E(s) = -\frac{1}{2}\sum_{ij} s_i w_{ij} s_j$$

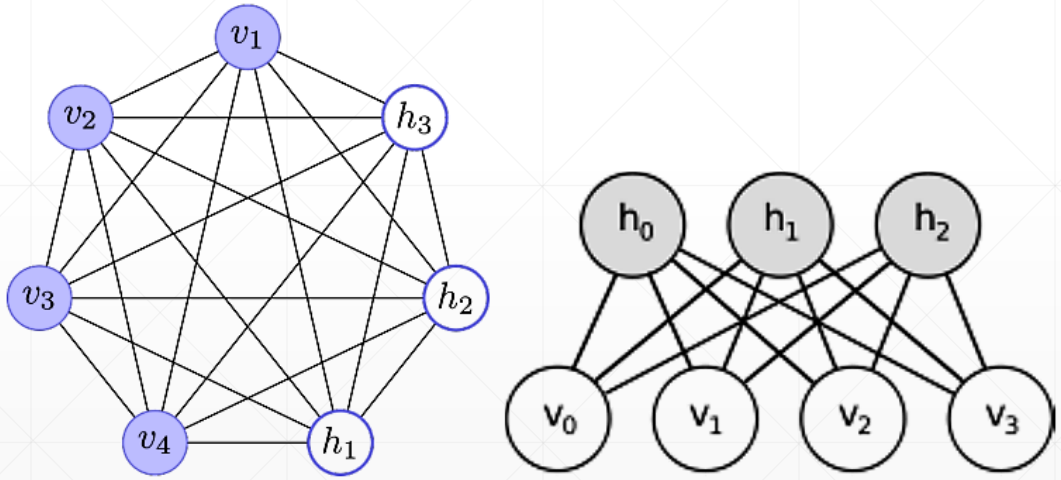
Обучение определяется уравнением динамики среднего поля: $\boldsymbol{v_i = \frac{1}{2}\left(1 + \tanh\left(\frac{H_i}{T}\right)\right)}$, где H_i – локальное среднее поле нейрона, v_i – это активность нейрона. Температура убывает по схеме **«имитационного закаливания» (simulated annealing)**

Пример «разогревания» функции ошибки



Ограниченная машина Больцмана

Джеффри Хинтон в 2006 году предложил новый подход, скомбинировав идеи полносвязных сетей с сетями обратного распространения ошибки. Это были **стохастические сети** у которых вероятности состояний подчинены распределению Больцмана, как функции энергии состояния и температуры системы нейронов – **машины Больцмана (Boltzmann machine).**



(a) Boltzmann machine (b) RBM

Затем Хинтон разделил нейроны на две группы – видимые (v) и скрытые (h). Данное ограничение и определило название – **ограниченная машина Больцмана**. Функция энергии такой сети:

$$E(v,h) = -\sum_{i \in visible} a_i v_i - \sum_{j \in hidden} b_j h_j - \sum_{i,j} v_i h_j W_{i,j},$$

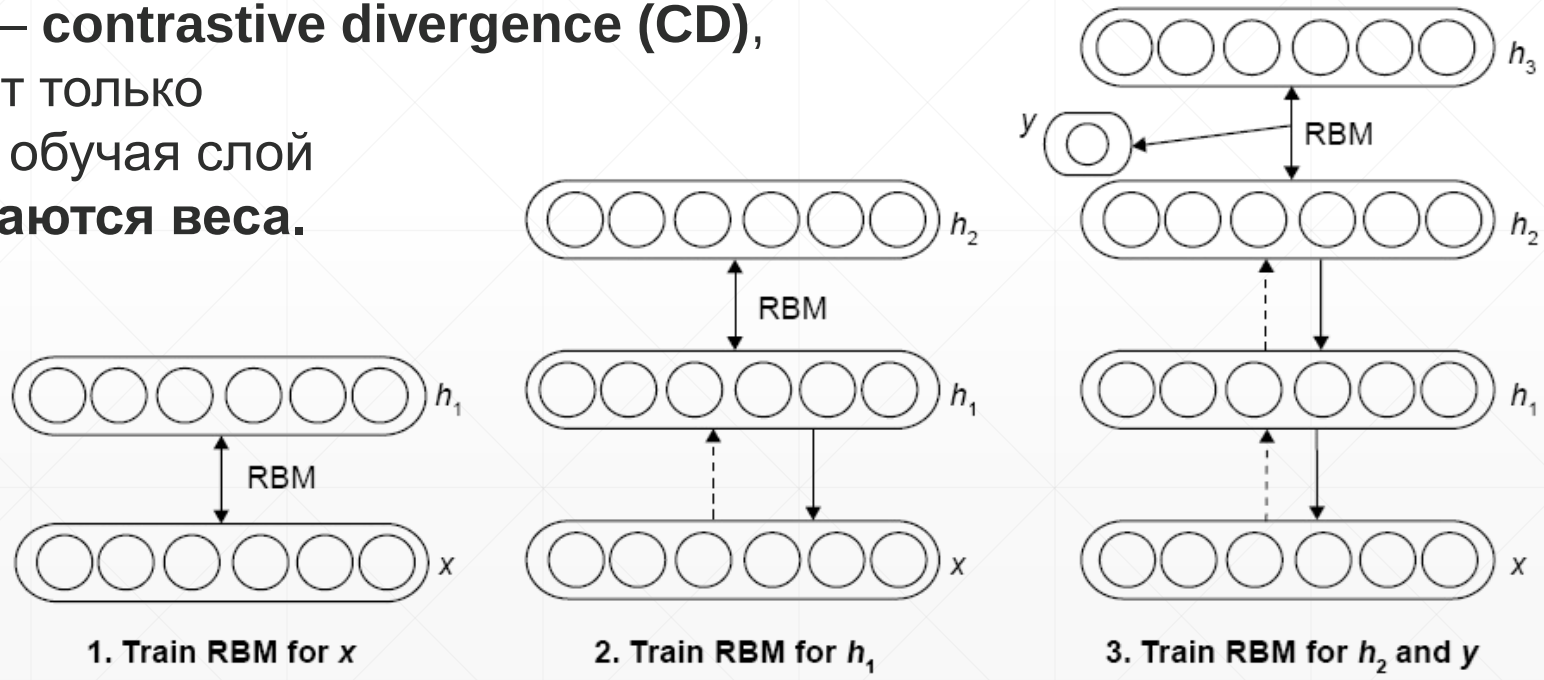
где v_i, h_j - это состояния видимых и скрытых нейронов, соответственно.

Restricted Boltzmann machine (RBM) – ограниченная машина Больцмана

Сеть глубокого доверия

Оказалось, что после обучения RBM, ее скрытые состояния **можно использовать для обучения другой RBM** более высокого уровня. Такой метод упаковки слоев (**stacking**) позволяет **выполнить настройку весов** для эффективного обучения глубокой сети. Так как обучение RBM **методом Монте-Карло по схеме Марковской цепи** – очень долгий процесс, Хинтон предложил свой алгоритм – **contrastive divergence (CD)**, при котором цепь выполнит только **один шаг**. Таким образом, обучая слой за слоем RBM – **настраиваются веса.**

Deep Belief Network



После обучения всех RBM добавляется слой классификации и выполняется backprop.

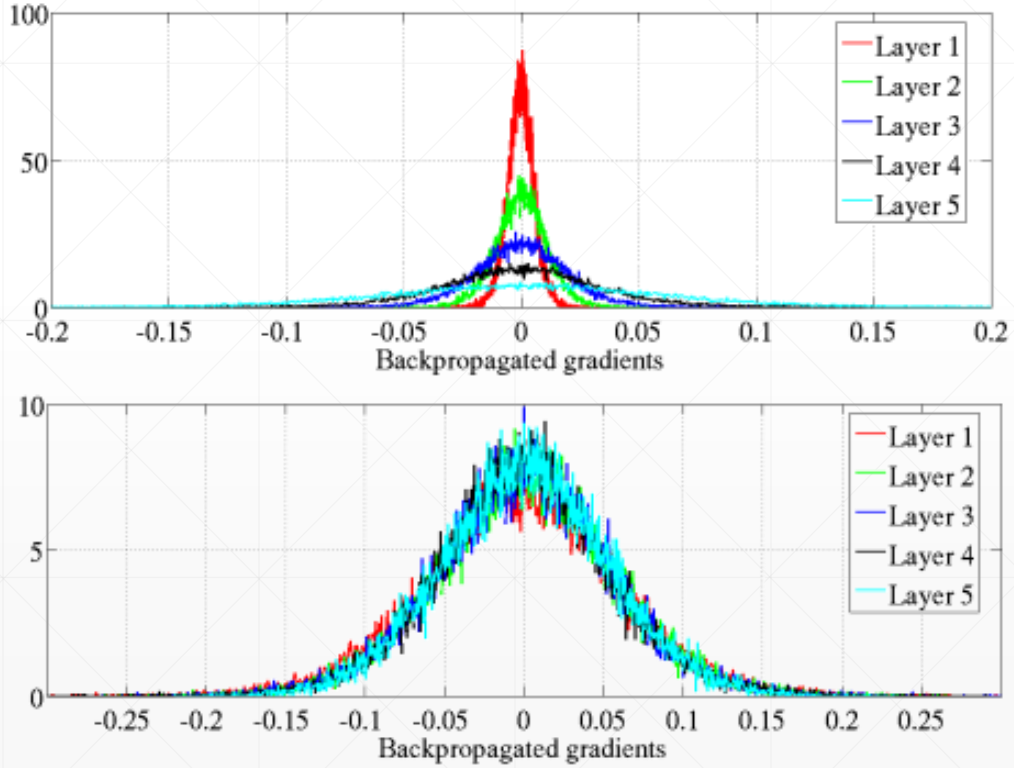
Нормализованная инициализация весов

Обучение сети глубокого доверия все равно слишком долго. Но можно **сразу инициализировать веса** так, чтобы предобучение не понадобилось. Для этого нужно, чтобы **дисперсия (расхождение) сигнала, проходящего через нейросеть была равная на каждом слое**.

Чтобы добиться этого, **нужно при установке параметров слоя учесть число нейронов на предыдущем слое**:

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right]$$

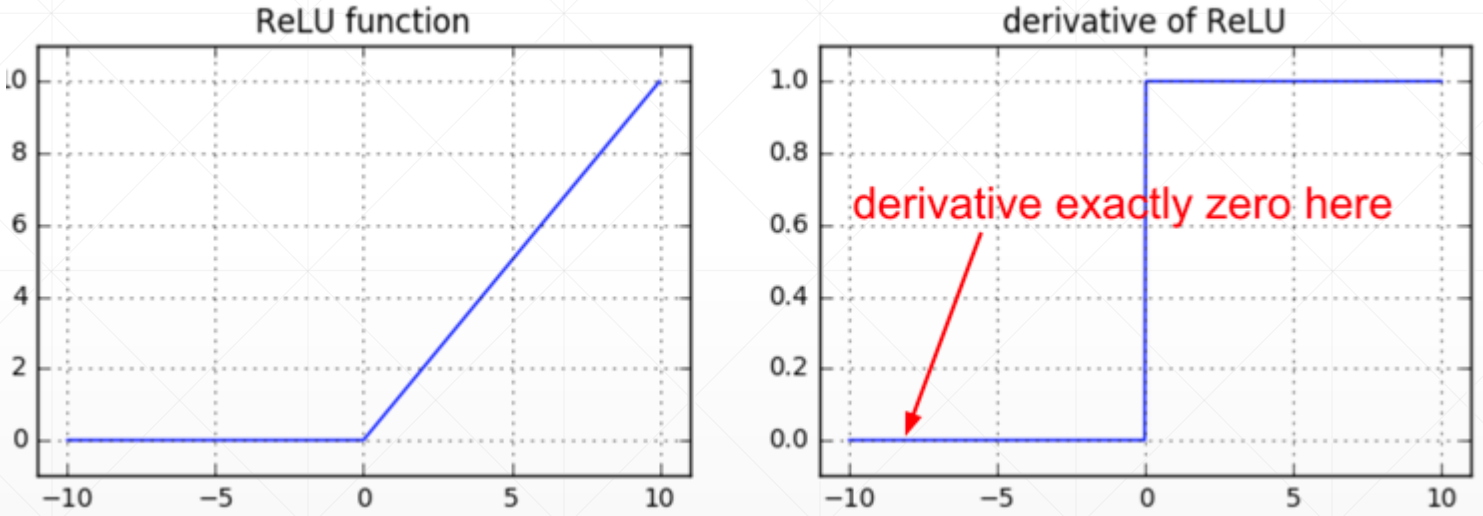
Это инициализация Глорота, подобранная спец. для функции активации – гиперболический тангенс.



В Keras по умолчанию *kernel_initializer='glorot_uniform'*

Выбор функции активации. ReLU

Дельта изменения весов первого слоя – это скорость обучения, умноженная на градиент ошибки слоя (**произведение всех градиентов предыдущих слоев**). Проблема сигмоида в том, что он **в любом случае имеет малое значение производной – 0.25.**

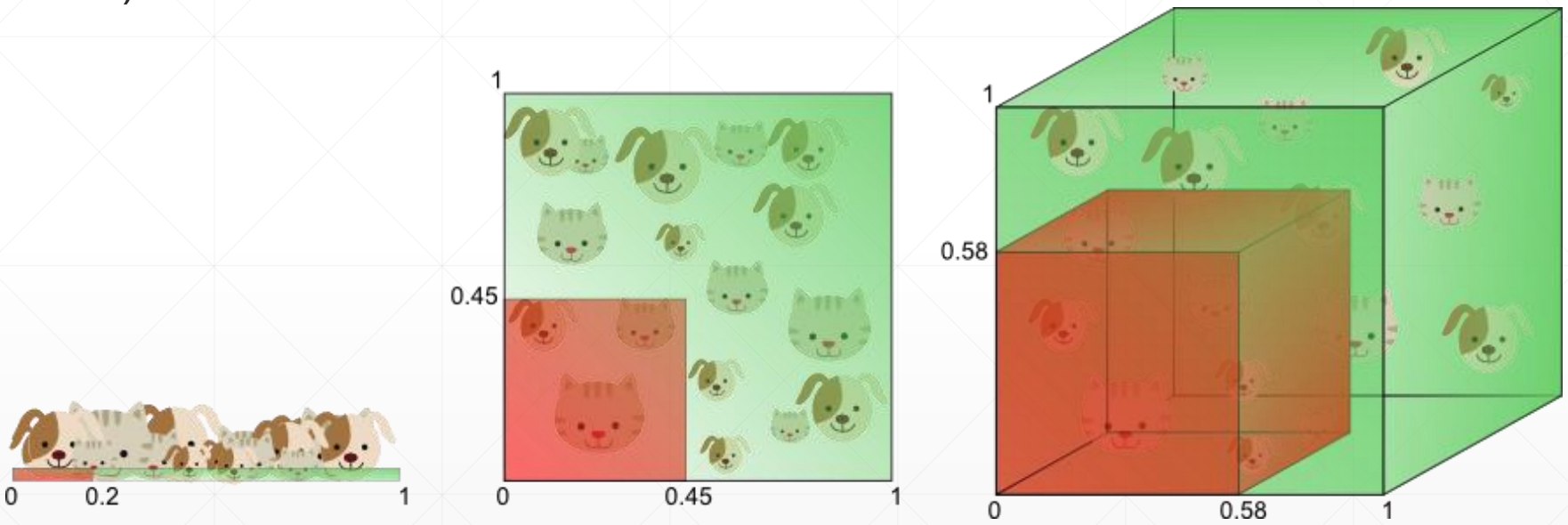


Усеченное линейное преобразование (ReLU) – это функция активации $f(x) = \max(0, x)$. ReLU имеет **только два значения производной: 1/0**.

Однако ReLU может вызвать проблему мертвого нейрона (**«dead ReLU» problem**) – такой нейрон **не активируется совсем**.

Проблемы: проклятье размерности

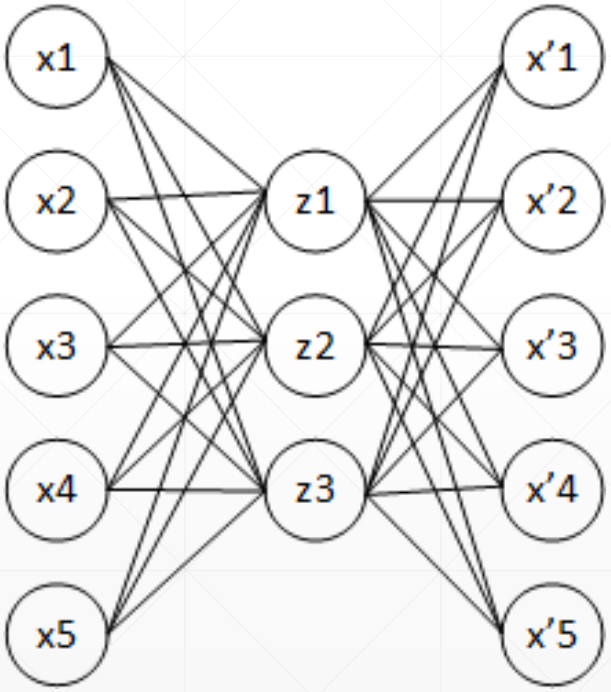
- Вторая проблема – это **проклятье размерности**. Глубокие нейросети очень хорошо подходят для обработки многомерных данных, но **чем больше размерность, тем больше примеров нужно, чтобы описать все случаи** (увеличивается по экспоненте).



Пример: для того, чтобы классифицировать кошек и собак по одному признаку нужно 100 примеров, тогда для двух признаков число комбинаций – 100^2, для трех – 100^3.

Сокращение размерности

Решение проблемы – **сокращение размерности входных данных**. Если перейти от стомерного случая к трехмерному, то потребуется **в N^{98} меньше примеров** и не будет необходимости строить сеть из десятка скрытых слоев.



Автоэнкодер (AE) – специальная архитектура ИНС, для обучения без учителя. Входной и выходной слои содержат **одинаковое число нейронов**, а средний из скрытых слоев состоит из **значительно меньшего их числа, образуя bottle neck**, заставляя нейросеть искать корреляцию в поступающих на вход данных, **выполняя их сжатие**.

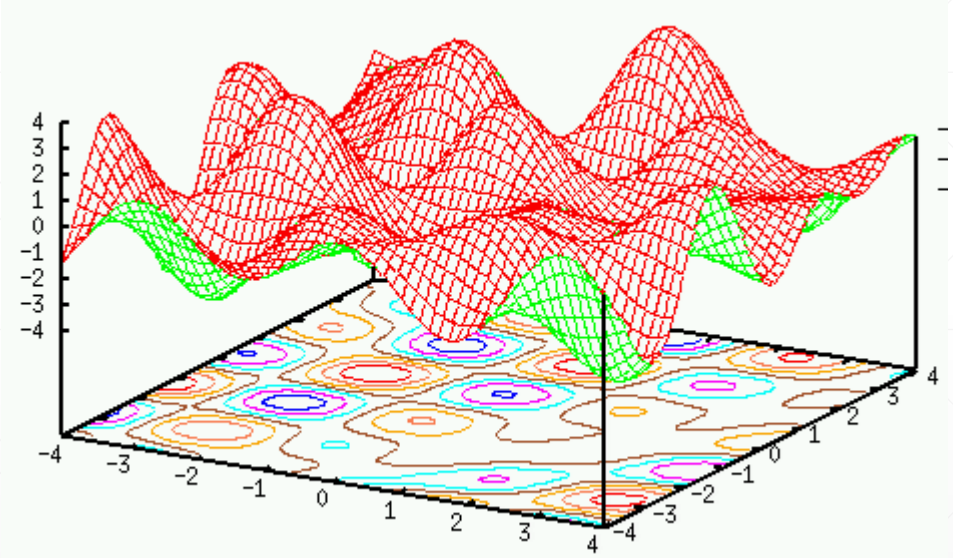
Цель: в процессе обучения получить на выходном слое отклик, наиболее близкий к входному.

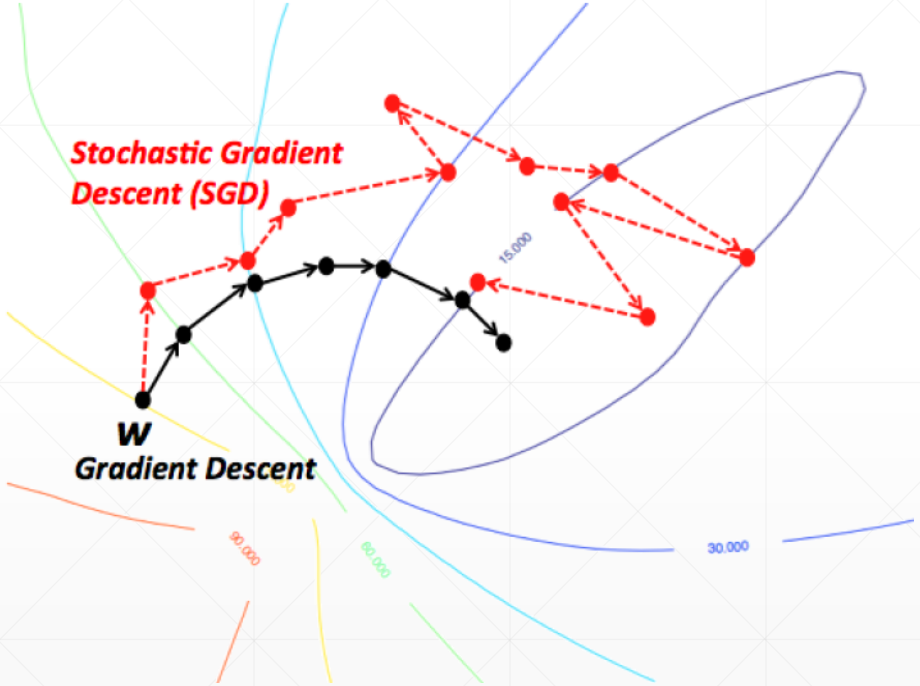
$$\frac{1}{m}\sum_{i}\left(h_w(x^{(i)}) - x^{(i)}\right)^2 \to min$$

Подробно автоэнкодеры будут рассмотрены в 12 лекции данного курса.

Проблемы: застревание в локальном минимуме

- У глубоких сетей с большим количеством слоев сложная функция ошибки, в которой **много локальных минимумов.** Обычный градиентный спуск сходится к ближайшему локальному минимуму.





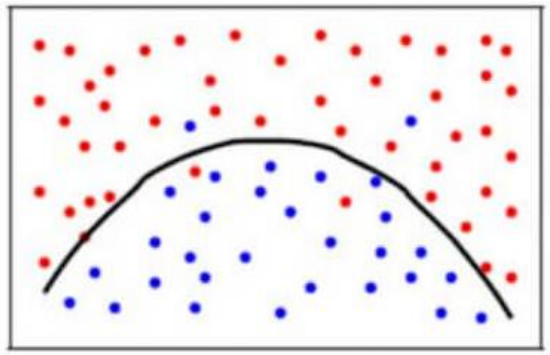
При стохастическом градиентном спуске значение градиента аппроксимируются градиентом функции стоимости, вычисленном только на небольшой части всех данных.

Обзор существующих методов оптимизации – http://ruder.io/optimizing-gradient-descent/

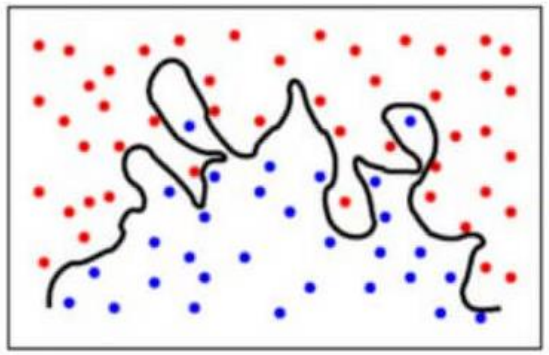
Проблемы: переобучение

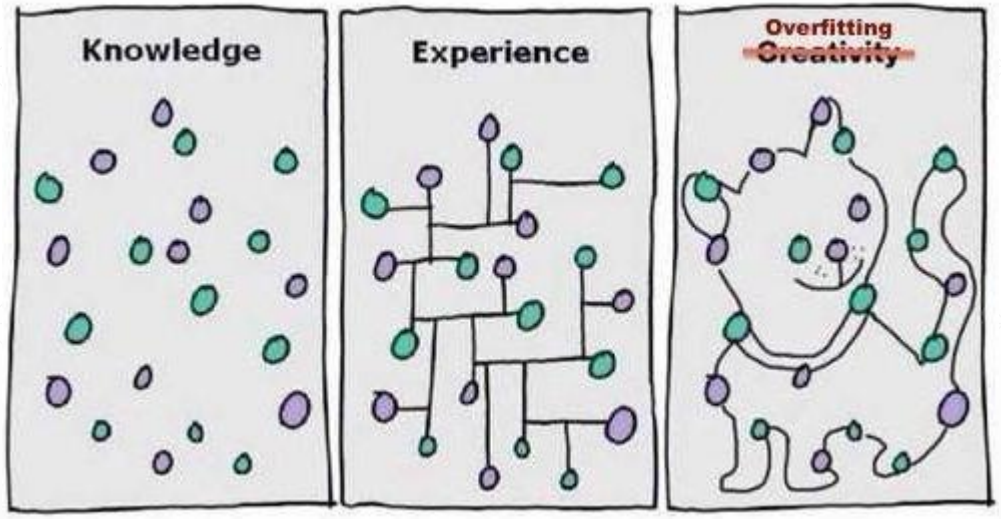
- Увеличение глубины сети неизбежно ведет к **росту числа параметров**. Большое число параметров приводит к **переобучению сети (overfitting)**.

Верная гипотеза



Переобучение





Переобучение (overfitting) — это излишне точное соответствие нейронной сети конкретному набору обучающих примеров, при котором сеть теряет способность к обобщению и не работает на тестовых примерах.

Как бороться с переобучением

1. **Ранняя остановка обучения**
2. **Техника регуляризации**

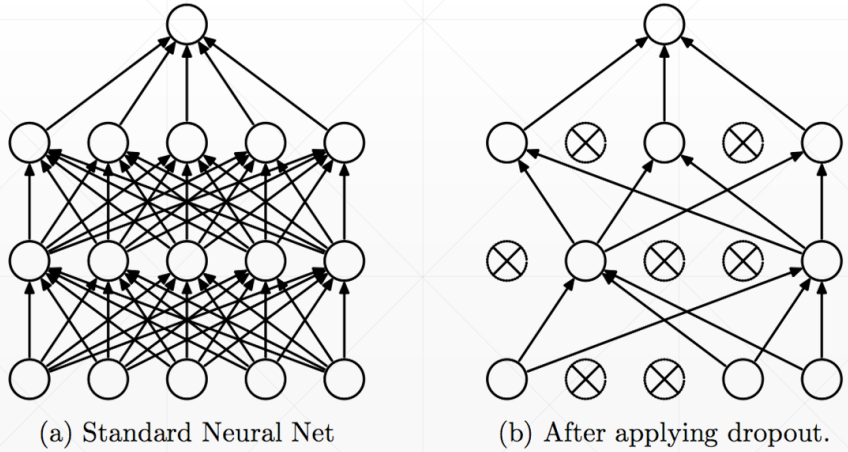
Метод регуляризации ограничивает реакцию нейросети на влияние шумов добавлением штрафного члена в функцию ошибки сети.

$$J(w) = J(w) + \frac{1}{2}\lambda \sum_i w_i^2$$

Коэфициент λ не должен быть большим и обычно $\lambda = 0.001$.

3. **Dropout**

Случайное прореживание нейронов – **dropout**, когда каждый вес обнуляется с вероятностью p либо умножается на $1/p$ с вероятностью $1 - p$
Dropout используют **только при обучении** сети.



(a) Standard Neural Net (b) After applying dropout.

В Keras есть специальный параметр «dropout»

Властью deep learning пускай этот хлеб станет котом!