Практический анализ данных и машинное обучение: искусственные нейронные сети

Соловей Влад и Ульянкин Филипп

20 апреля 2019 г.

Как обучаются нейросети

Agenda

- Регуляризация в нейронных сетках
- Про функции активации, бачнормализацию и инициализацию
- Другие эвристики, которые используются при обучении сеток
- Введение в tensorflow

Память

Что мы должны помнить с прошлой лекции.

1. Для нахождения минимума по сложным функционалам считают стохастический градиентный спуск
2. Полносвязная нейронная сеть с 1 скрытым слоем может представить любую "гладкую" функцию
3. Полносвязная нейронная сеть-просто набор линейных функций с нелинейными активациями


Overfitting
?!?
Too many neurons
Not enough DATA
BAD Network

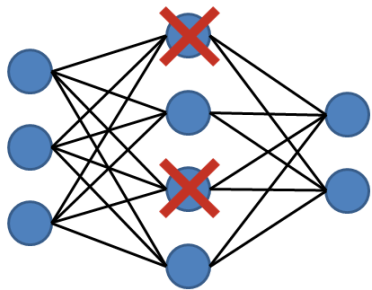
Эвристики, используемые при обучении

- Применимы все те же эвристики, что и в градиентном спуске
- Инициализация весов, выбор шага, порядок предъявления объектов
- Кроме того появляются новые проблемы: выбор функции активации в каждом нейроне, выбор числа слоёв, числа нейронов, выбор значимых связей
- Это приводит к возникновению новых эвристик

Регуляризация в нейронных сетях

Dropout

- Придумали в 2014 году
- С вероятностью p отключаем нейрон
- Делает нейроны более устойчивыми к случайным возмущениям



Dropout

- На каждой итерации мы изменяем только часть параметров, обучение нейронов протекает чуть более независимо
- При тестировании все нейроны присутствуют в сетке, но их выходы домножаются на вероятность p
- На самом деле с байесовской точки зрения, вводя дропаут, мы обучаем ансамбль нейронных сетей

Регуляризация

- L_2: приплюсовываем к функции потерь $\lambda \cdot \sum w_i^2$
- L_1: приплюсовываем к функции потерь $\lambda \cdot \sum |w_i|$
- Можно регуляризовать не всю сетку, а отдельный нейрон или слой
- Не даёт нейрону сфокусироваться на слишком выделяющемся входе
- Очень похожа по своему действию на droput

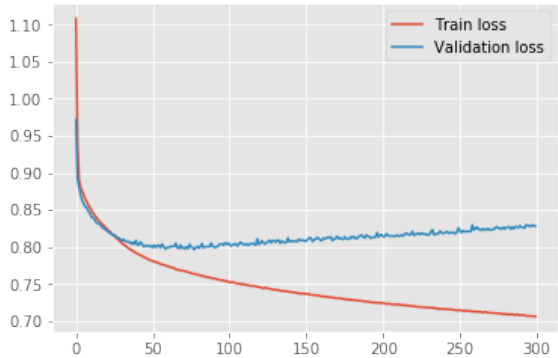
Регуляризация

- В keras можно добавить для каждого слоя на три вида связей:
- **kernel_regulirizer** - на матрицу весов слоя;
- **bias_regulirizer** - на вектор свободных членов;
- **activity_regulirizer** - на вектор выходов.
- **Делается примерно так:**

```
model.add(Dense(256, inpit_dim = 32,
        kernel_regulirizer = regulizers.l1(0.001),
        bias_regulirizer = regulizers.l2(0.1),
        activity_regulirizer = regulizers.l2(0.01)))
```

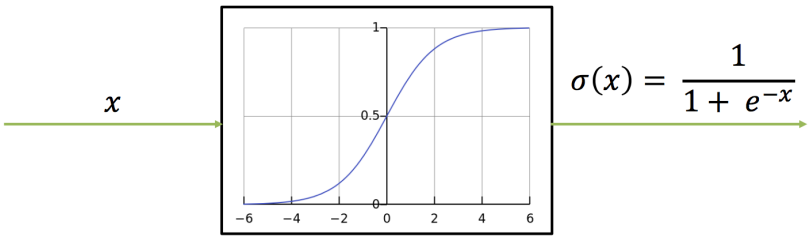
Early stopping

- Подбор числа шагов для обучения на основе провреки на валидации
- Для линейной модели с квадратичной (MSE) функцией потерь и SGD рання остановка эквивалентна L_2 регуляризации

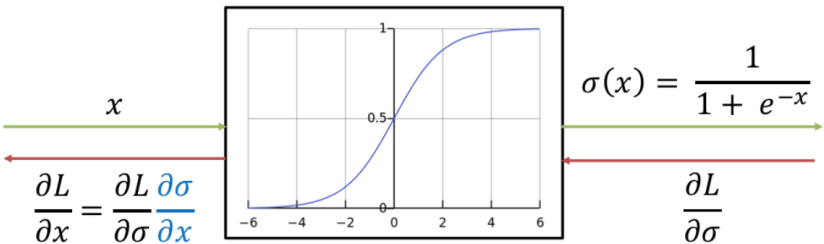


Какими бывают функции активации и как через них пробросить градиент

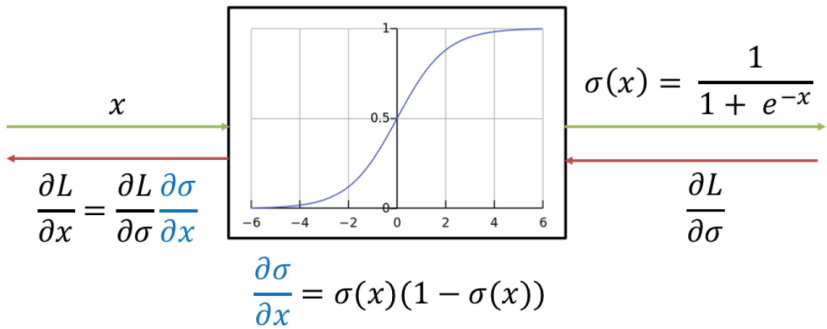
Sigmoid activation



Sigmoid activation

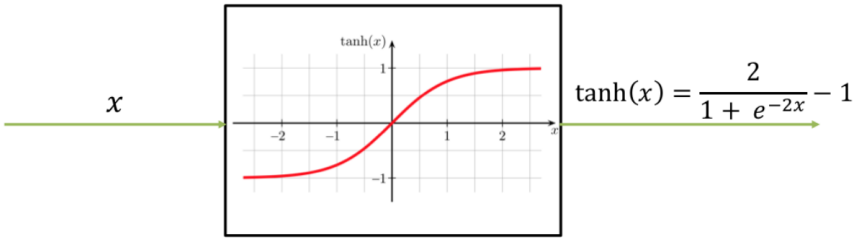


Sigmoid activation



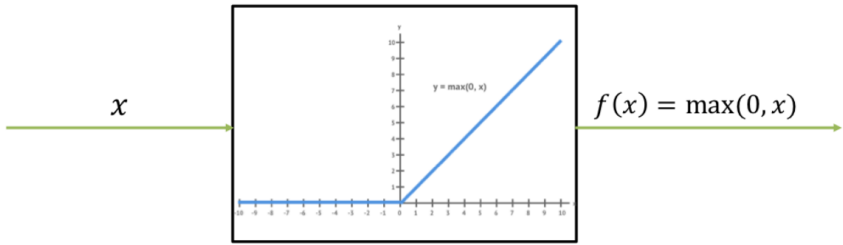
- В глубоких сетях способствует затуханию градиента (vanishing gradients)
- Не центрирована относительно нуля
- Вычислять e^x дорого

Tanh activation



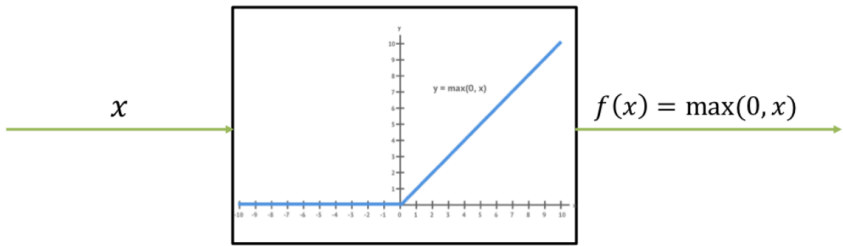
- Центрирован относительно нуля
- Всё ещё похож на сигмоиду

ReLU activation



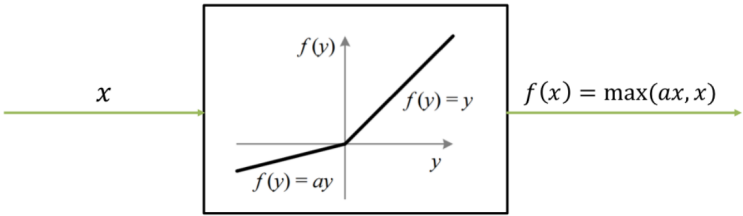
- Быстро вычисляется
- Градиент никогда не зануляется при $x > 0$
- Сходимость сеток ускоряется

ReLU activation



- Сетка может умереть, если активация занулится на всех нейронах
- Не центрирован относительно нуля

Leaky ReLU activation



- Как ReLU, но не умирает
- Важно, чтобы $a \neq 1$, иначе линейность

Что же выбрать

- На самом деле это неважно
- Важно собрать хорошую архитектуру и подобрать метод оптимизации
- Обычно в сетках работают либо с сигмоидом либо с ReLU, если остаётся время, то пробуют другие идеи, но обычно выгрыш в качестве от перебора в функциях активации довольно низкий

Инициализация весов

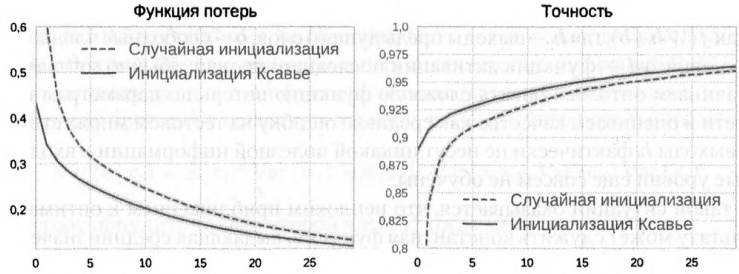
Инициализация весов

- Наши признаки X пришли к нам из какого-то распределения
- Выход слоя $f(XW)$ будет принадлежать другому распределению
- Если инициализировать веса неправильно, дисперсия распределения може от слоя к слою затухать, сигнал будет теряться
- Эмпирически было выяснено, что это может портить сходимость для глубоких сеток

Инициализация весов

- Для симметричных функций с нулевым средним используйте инициализацию Ксавье init="glorot_uniform" или
- Для ReLU и им подобным инициализацию Хе init="he_uniform" или init="he_nomal"
- Эти две инициализации корректируют параметры распределений в зависимости от входа и выхода слоя так, чтобы поддерживать дисперсию равной единице

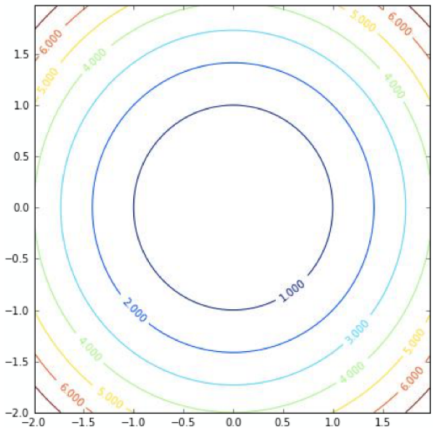
Эксперимент с MNIST

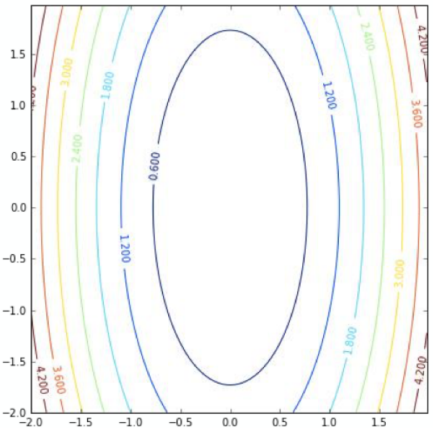


Источник: Николенко, страница 149

Батч-нормализация

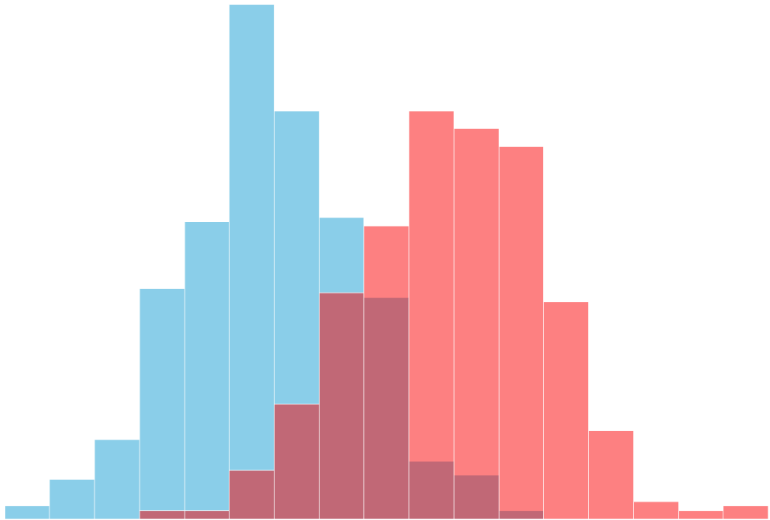
Стандартизация





Какая из ситуаций лучше для SGD?

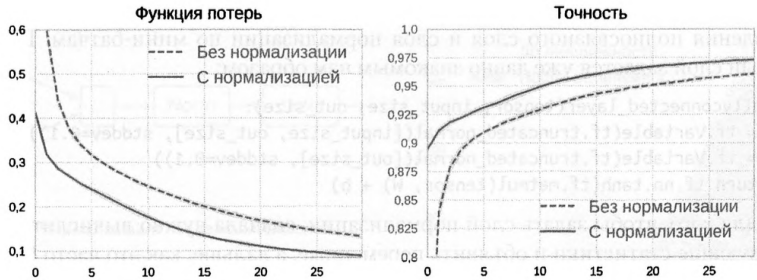
Стандартизация



Batch norm

- Придумали в 2015 году
- Давайте вместо X на входе использовать $\frac{X-\mu_X}{\sigma_X}$
- Давайте на каждом слое вместо $f(XW)$ использовать $\hat{X} = \frac{f(XW)-\mu_f}{\sigma_f}$
- Математическое ожидание и дисперсию будем оценивать по батчу
- На выход будем выдавать $\beta\hat{X} + \gamma$, для того, чтобы у нас было больше свободы, параметры β и γ тоже учим
- Обучение довольно сильно ускоряется

Эксперимент с MNIST



Источник: Николенко, страница 160

Другие эвристики для обучения сеток

Предобучение

- Обучаем каждый нейрон на рандомной подвыборке, каждый нейрон впитает какие-то отдельные её особенности, после скрепляем все нейроны вместе и продолжаем обучение на всей выборке

- Обучаем на корпусе картинок автокодировщик, encoder благодаря этому учится выделять наиболее важные фичи, которые позволяют эффективно сжимать изображения. После срезаем decoder и на его месте достраиваем слои для решения нашей задаче, запускаем обычное дообучение.

Динамическое наращивание сети

- Обучение сети при заведомо недостаточном числе нейронов H
- После стабилизации функции потерь --- добавление нового нейрона и его инициализация путём обучения
  - либо по случайной подвыборке
  - либо по объектам с наибольшими значениями потерь
  - либо по случайному подмножеству входов
  - либо из различных случайных начальных приближений
- Снова итерации BackProp

Эмпирический опыт: Общее время обучения обычно лишь в $1.5 - 2$ раза больше, чем если бы в сети сразу было итоговое число нейронов. Полезная информация, накопленная сетью не теряется при добавлении нейронов.

Прореживание сети

- Начать с большого количество нейронов и удалять незначимые по какому-нибудь критерию
- Пример: обнуляем вес, смотрим как сильно упала ошибка, сортируем все вязи по этому критерию, удаляем N наименее значимых
- После прореживания снова запускаем backprop
- Если качество модели сильно упала, вернуть последние удалённые связи

Вводимся в tensorflow