KARPOV.COURSES >>>
КОНСПЕКТ

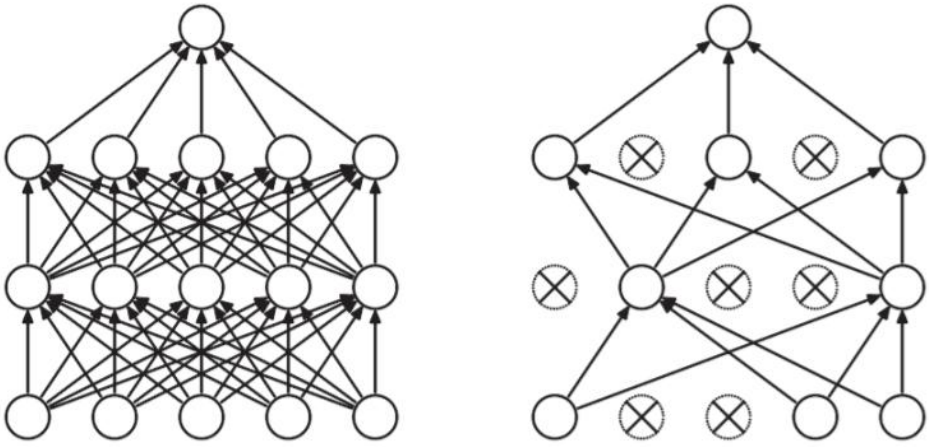
Конспект > 4 урок > Сверточные нейронные сети. Часть II

> Регуляризация нейронных сетей

Для регуляризации линейных моделей используются, например, L1 и L2 регуляризаторы.

А как можно улучшить обобщающую способность нейронных сетей?

Для регуляризации нейронных сетей используется метод Dropout, при котором нейроны полносвязного слоя случайно выключаются в ходе обучения. Существует некоторая интерпретация, что на этапе применения прогнозы разных архитектур как бы усредняются, и это препятствует переобучению модели.



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

Рассмотрим данный метод немного подробнее. Допустим, существует полносвязный слой из четырех нейронов, для каждого из которых вероятность того, что он не зануллится, равна `p`. При обучении модели один из четырех нейронов полносвязного слоя выключается с вероятностью `1-p`, и следующий слой получает три нейрона вместо ожидаемых четырех. Такой процесс происходит для каждого из нейронов.

Этапы метода Dropout:

1. Определение отдельного слоя без параметров:

$d(x)$

2. Обучение модели:

$d(x) = p1x * m$

m - вектор такого же размера как и x, из 0 и 1, вероятность получить 1 равна p, вероятность получить 0 равна $1 - p$.

Деление на p необходимо для сохранения суммарного масштаба выходов.

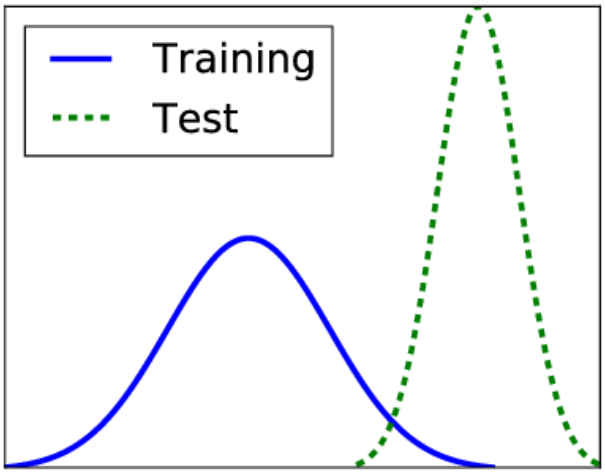
3. Применение модели:

$d(x) = x$

На этапе применения модели прогнозы отдельных архитектур усредняются, и модель выдает лучший результат за счет отсутствия переобучения.

> Нормализация выходов слоёв

Модели машинного обучения чувствительны к изменению в распределении тренировочных и тестовых данных, так называемому Covariate shift или Domain shift.



Существуют методы борьбы с этим, например, модификация функции потерь, позволяющая учитывать похожесть объектов из обучающей выборки на тестовую выборку.

А что происходит в глубинном обучении?

В глубинном обучении каждый слой обучается на выходе из предыдущего слоя, и если этот выход сильно изменится в процессе обучения (например, нам попался какой-то специфический батч объектов), то веса следующего слоя не будут к этому готовы. Произойдет Internal Covariate Shift, из-за чего инициализация станет практически
рандомной, и модель не может адекватно функционировать.

Для борьбы с этим придумали специальный слой Batch Normalization. Он был задуман как способ модифицировать данные, которые попадают ему на вход, так, чтобы у них было какое-то фиксированное обучаемое распределение.

Этапы Batch normalization:

1. Определение отдельного слоя:

$bn(x)$

2. Оценка среднего и дисперсии каждой компоненты входного вектора:

$\mu B = n1i = 1 \sum nBi$

$\sigma B2 = n1i = 1 \sum n(Bi - \mu B)2$

3. Масштабирование всех выходов слоя:

$B^i = \sigma B2 + \varepsilon$

$Bi - \mu B$

4. Определение нужных (обучаемых) среднего и дисперсии параметрами γ и β:

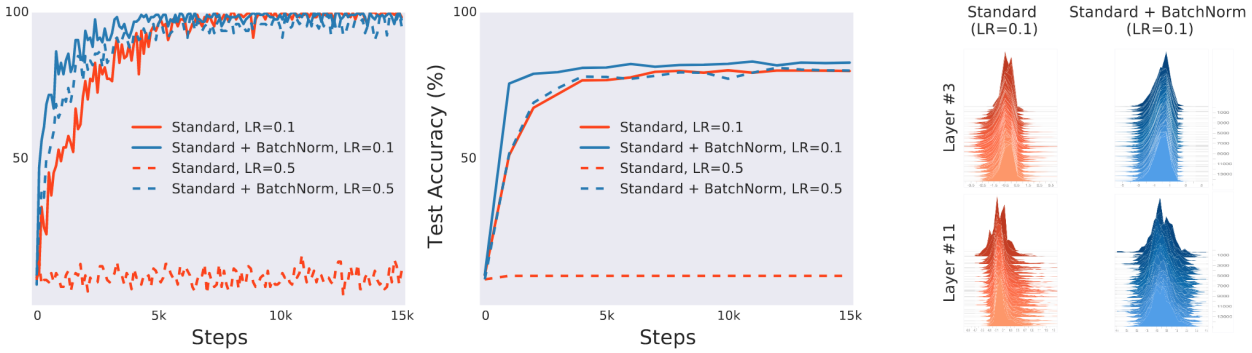
$bn(B) = B^* \gamma + \beta$

Размерность γ и β равна размерности входных векторов.

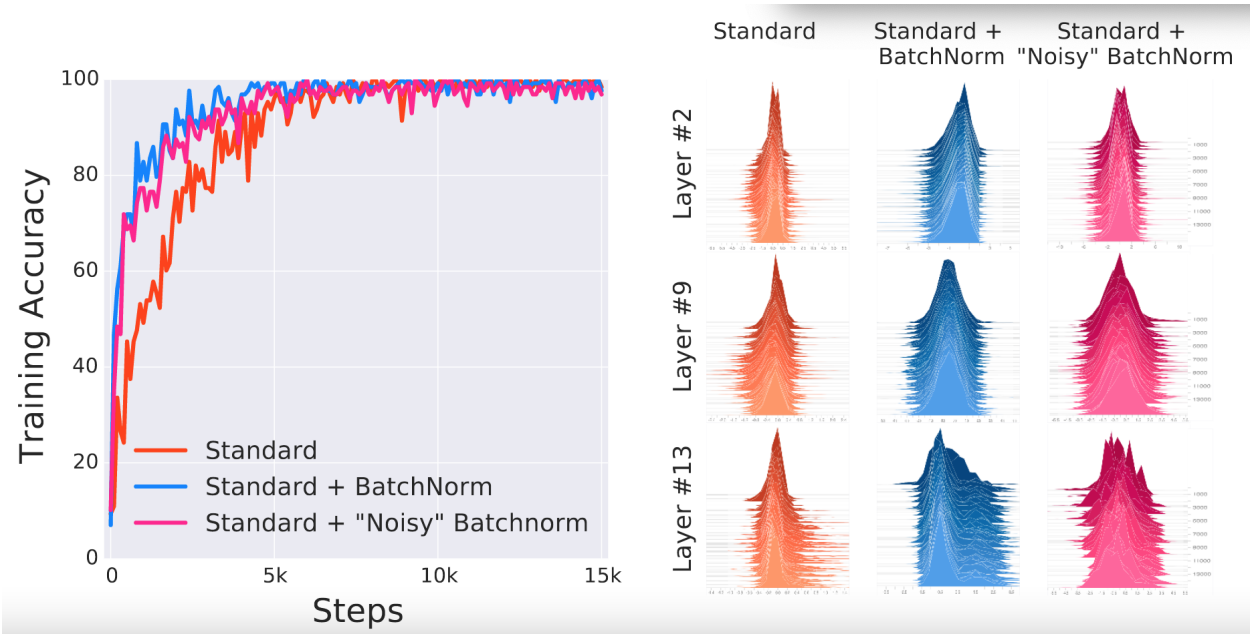
Во время применения нейронной сети все вычисления те же самые, но μ_B и σ_B^2 не вычисляются заново, а используются их средние значения по всем батчам во время обучения.

Через несколько лет после выхода статьи, где представили BatchNorm (2015 год), вышла другая статья, где авторы поэкспериментировали с батч нормом и сделали свои выводы про этот слой:

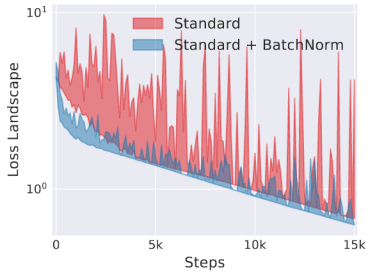
1) Батч-нормализация позволяет **увеличить длину шага** в градиентном спуске, что дает возможность модели быстрее обучаться.



2) Гипотеза, что батч-нормализация решает проблему **internal covariate shift** оказалась **неверной**: даже если шум добавляли после применения батчнорма, то модель всё равно оказывалась лучше, чем без батчнорма.

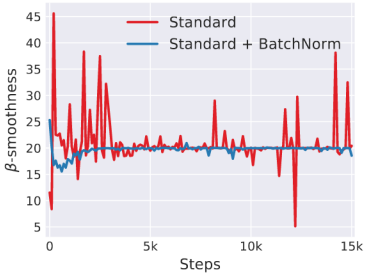


3) **Функционал ошибки** становится сильно более гладким, за счет чего оптимизация нейронных сетей и идет **быстрее** и проще.



(a) loss landscape

(b) gradient predictiveness



(c) “effective” β-smoothness

Таким образом, метод **BatchNorm** оказался не очень эффективным методом в борьбе с Internal Covariate Shift, однако он всё равно позволяет улучшить качество модели.

> Нормализация входных данных. Инициализация параметров. Аугментация данных

Нормализация входных данных осуществляется обычно путем **масштабирования**, то есть вычитания среднего и деления на корень из стандартного отклонения, вычисленных на обучающей выборке.

Инициализация параметров в слоях в нейронных сетях сильно отличается от инициализации параметров в классическом машинном обучении. В последнем инициализировать можно практически чем угодно - обычно инициализацию проводят единицами.

Правила инициализации параметров в слоях нейронных сетей:

1. Инициализация не должна быть симметричной;

2. Плохо инициализировать одним и тем же числом.

Пример хорошего варианта, который не дает числам улететь в бесконечность:

$wi \sim n$

$2N(0,1)$, где n - число входов.

Из-за того, что нейронные сети в основном работают с картинками, возникает проблема ограниченного набора данных при решении задач. Аугментация

данных - это методика создания дополнительных данных из имеющихся данных. Например, при работе с картинками это методы flip, crop, blur и так далее.

Примеры аугментации данных.

Еще примеры аугментации данных.