Нейронные сети.
Перцептроны. Обучение нейронных сетей

Алексей Колесов

Белорусский государственный университет

25 октября 2018 г.

Содержание

Перцептрон Розенблатта

- решаем задачу бинарной классификации, ответы $\{-1, 1\}$
- на вход приходит $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$, $y(x)$ — ответ
- ищем $w_0, \ldots, w_d$, что $\mathrm{sign}(w_0 + w_1 x_1 + \ldots + w_d x_d)$ как можно более похоже на $y(x)$

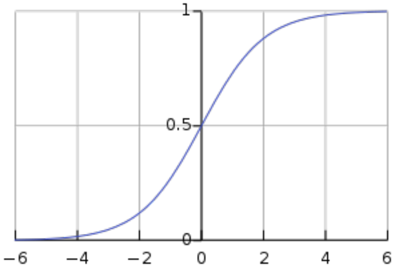
Как обучать?

- оптимизировать долю правильных ответов (accuracy) — сложно
- пусть M — множество x, где модель даёт неверный ответ
- $E_P(w) = -\sum_{x \in M} y(x)\langle w, x\rangle$ — критерий перцептрона
- можно оптимизировать стохастическим градиентным спуском

Функция активации

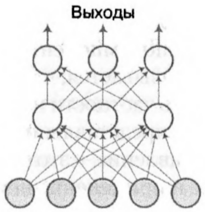
- если оставить перцептрон линейным, то выразительной модели не получится
- применим к выходу функцию активации
- стандартный выбор: сигмоид $\sigma(x) = \dfrac{1}{1+e^{-x}}$

Сигмоид



$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

Больше перцептронов богу перцептронов



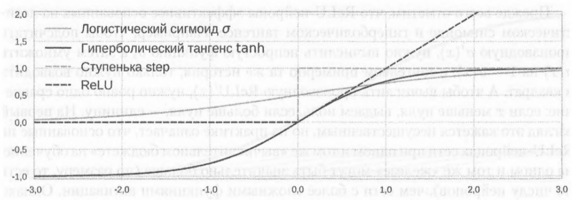
- можем объединять несколько перцептронов в граф, обучать с помощью sgd
- лучше объединять в слои: внутри слоя нет связей — можно векторизировать вычисления

Содержание

Современные функции активации

- σ — плохая функция активации: быстро насыщается, тяжело считать
- есть много других вариантов

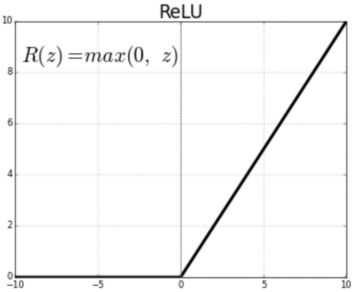
tanh



$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$
$$\tanh'(x) = 1 - \tanh^2(x)$$

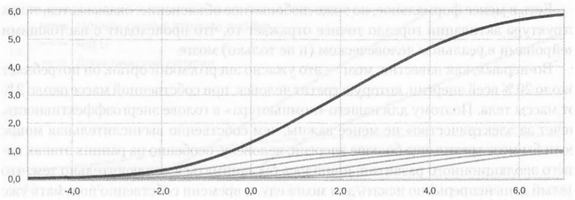
- аффиноe преобразование σ
- круче растёт, быстрее достигает насыщения ($\sigma'(0) = \frac{1}{4}$, $\tanh'(0) = 1$)
- ноль — точка с большой производной (в отличие от сигмоида)

Rectified linear unit



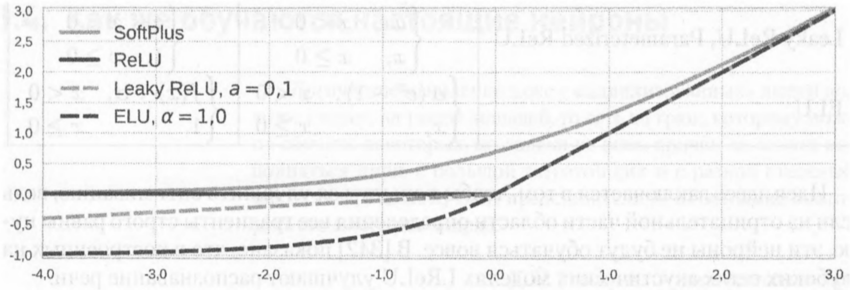
- эффективно вычисляются (в том числе производная)
- не насыщается
- умеет отличать «сильно» активированные нейроны от «несильно»

Rectified linear unit



- рассмотрим $f(x) = \sum_{i=0}^{\infty} \sigma(x + \frac{1}{2} - i) \approx \log(1 + e^x)$
- если $x < 0$, то примерно ноль
- если больше, то примерно 1

Модификации ReLU



Функции активации

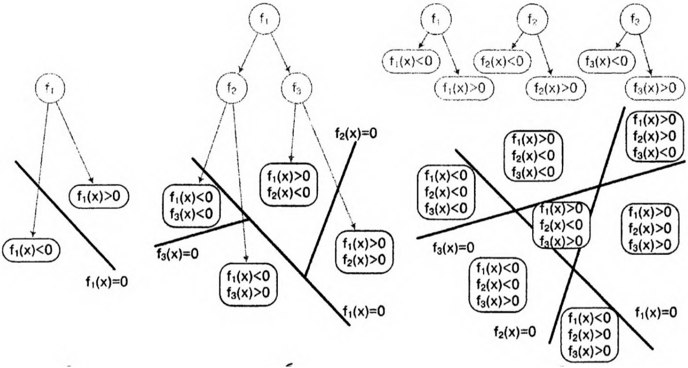
- по умолчанию, используйте ReLU или аналог
- можно попробовать tanh
- перебор аналогов не даёт большой пользы

Содержание

Глубокие сети это хорошо

- глубина сети — количество слоёв
- любую непрерывную функцию можно приблизить сколько угодно точно нейронной сетью с одним скрытым слоем

Зачем глубина



- для глубокой сети нужно меньшее число нейронов
- чем выше слой, тем более общая у него информация

Сложность обучения нейронных сетей

- если один или несколько слоёв насытились, то возникает проблема «затухающих» градиентов
- если градиент большой у нескольких слоёв, то проблема «взрывающихся» градиентов

Почему сейчас всё работает?

- мы чуть лучше научились оптимизировать нейронные сети
- данных стало больше
- компьютеры стали быстрей

Содержание

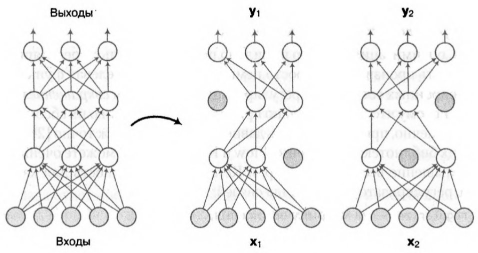
Стандартные методы

- добавить в функцию потерь $\lambda \sum |w_i|$ — L_1
- добавить в функцию потерь $\lambda \sum w_i^2$ — L_2
- можно применять к весам, к bias, к активациям

Early stopping

- параллельно с тренировкой регулярно проверяем качество на отложенном сете
- если ухудшается — останавливаемся
- в некотором смысле эквивалентен L_2

Dropout



- при тренировке с вероятностью p зануляем активацию нейрона
- при инференсе умножаем выход на $1 - p$

Итоги

- используйте early stopping
- если есть переобучение, добавьте щепотку dropout
- l_1, l_2 можно, но увлекаться не стоит

Содержание

Инициализация весов

- градиентный спуск — локальный метод
- при плохой инициализации найдём плохой минимум («All you need is good init» https://arxiv.org/abs/1511.06422)
- особенно важно для глубоких сетей

Предобучение

- предобучение без учителя помогает учить глубокие сети
- учим слой «предсказывать» свой вход
- учим второй слой «предсказывать» первый
- получаем Deep Boltzmann machine
- дообучаем с помощью обучения с учителем
- такая схема учит не только $p(y|x)$, но и $p(x)$

Рецепт

- обучать от первых слоёв к последним: меньше вычислений, нет затухания градиентов
- предобучение без учителя: больше возможных данных
- полученная модель лучше сходится

Инициализация Глоро (Xavier)

Рассмотрим дисперсию двух последовательных слоёв сети:

$$y = w^T x + b = \sum_i w_i x_i + b$$

Пусть w_i и x_i независимы:

$$\begin{aligned} y_i &= w_i x_i \\ \mathrm{Var}(y_i) = \mathrm{Var}(w_i x_i) &= \mathbb{E}[x_i^2 w_i^2] - (\mathbb{E}[x_i w_i])^2 \\ \mathrm{Var}(y_i) = \mathbb{E}[x_i]^2 Var(w_i) + \mathbb{E}[w_i]^2 &\mathrm{Var}(x_i) + \mathrm{Var}(w_i) Var(x_i) \end{aligned}$$

Если активация симметрична относительно нуля и инициализация имеет нулевое среднее:

$$\mathrm{Var}(y_i) = \mathrm{Var}(w_i)\,\mathrm{Var}(x_i)$$

Инициализация Глоро (Xavier)

Имеем:

$$y_i = w_i x_i$$
$$\mathrm{Var}(y_i) = \mathrm{Var}(w_i)\,\mathrm{Var}(x_i)$$

Если предположить, что w_i из одного распределения и независимы (и x_i тоже), то:

$$\mathrm{Var}(y) = \mathrm{Var}\left(\sum_{i=1}^{\mathrm{n_{out}}} \mathrm{Var}(w_i x_i)\right) = \mathrm{n_{out}}\,\mathrm{Var}(w_i)\,\mathrm{Var}(x_i)$$

Инициализация Глоро (Xavier)

Имеем:

$$\mathrm{Var}(y) = \mathrm{n_{out}}\,\mathrm{Var}(w_i)\,\mathrm{Var}(x_i)$$

Если использовать инициализацию $w_i \sim U\left[-\frac{1}{\sqrt{\mathrm{n_{out}}}}, \frac{1}{\sqrt{\mathrm{n_{out}}}}\right]$, то:

$$\mathrm{Var}(w_i) = \tfrac{1}{12}\left(\frac{1}{\sqrt{\mathrm{n_{out}}}} + \frac{1}{\sqrt{\mathrm{n_{out}}}}\right) = \frac{1}{3\,\mathrm{n_{out}}}$$

$$\mathrm{n_{out}}\,\mathrm{Var}(w_i) = \tfrac{1}{3}$$

Инициализация Глоро (Xavier)

Рассмотрим обратный ход:

$$\frac{\partial L}{\partial y_i^{(l)}} = f'(y_i^{(l)}) \sum_j w_{i,j}^{(l+1)} \frac{\partial L}{\partial y_j^{(l+1)}}$$

Если у функции единичная производная около нуля (tanh), то получаем такую же ситуацию, что в прямом: коэффициент $\mathrm{n_{in}}\,\mathrm{Var}(w_i)$

Инициализация Глоро (Xavier):

$$\mathrm{Var}(w_i) = \frac{2}{\mathrm{n_{in}} + \mathrm{n_{out}}}$$

Например,

$$w_i \sim U\left[-\frac{\sqrt{6}}{\sqrt{\mathrm{n_{in}} + \mathrm{n_{out}}}}, \frac{\sqrt{6}}{\sqrt{\mathrm{n_{in}} + \mathrm{n_{out}}}}\right]$$

Инициализация Хе

Если проделать то же самое для ReLU:

$$w_i \sim N\left(0, \sqrt{\frac{2}{\mathrm{n_{in}}}}\right)$$

Выводы

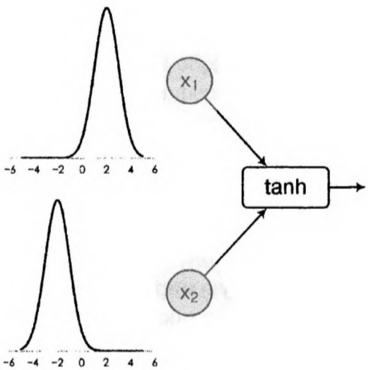
- предобучение применяется всё реже
- для симметричных — Ксавьер
- для ReLU — Хе
- для bias — ноль

Содержание

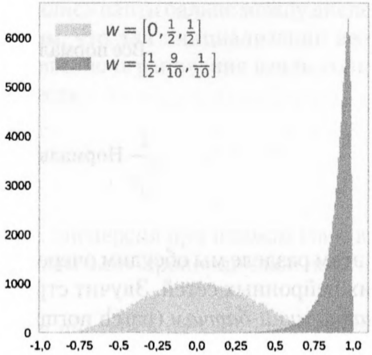
Минибатчи

- более точная оценка на градиент (что необязательно хорошо)
- векторизация вычислений
- участие в batchnorm!

Internal covariance shift



а



б

Решение есть!

- проблеме уделяют внимание целые книги
- в нейронных сетях полезно применять whitening для входов
- в общем, нормализация — лучшее решение
- нормализацию надо учитывать при обучении!

Учёт нормализации при обучении

- давайте «отбеливать» выход каждого слоя
- при подсчёте в лоб нужно считать производную по всему сету! ($\mathrm{Norm}(x, X)$)
- для отбеливания нужно считать матрицу ковариаций
- выход: давайте считать статистики по батчу, а ковариации поэлементно: $\hat{x}_k = \dfrac{x_k - \mathbb{E}[x_k]}{\sqrt{\mathrm{Var}(x_k)}}$

Batchnorm

- при сигмоиде, мы двигаем результат в «линейную» область
- давайте разрешим «отменять» нормализацию
- $y_k = \gamma_k \hat{x}_k + \beta_k$

Batchnorm

Пусть на входе батч: $B = x_1, \ldots, x_m$. Тогда batchnorm:

- $\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$
- $\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_B)^2$
- $\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$
- $y_i = \gamma \hat{x}_i + \beta$

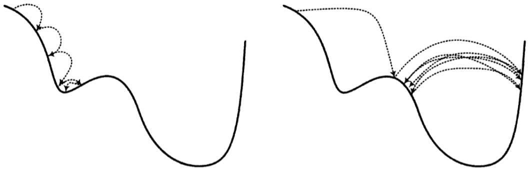
Не забыть взять производные!

Итоги

- batchnorm — революция в машинном обучении: любая свёрточная сеть на изображениях «работает»
- в полносвязных сетях есть истории успеха, но меньше, в рекуррентных — пока нет хорошего решения
- есть другие способы нормализации: layer norm
- надо не забыть «заморозить» статистики при инференсе

Содержание

Learning rate



- большой lr $\Rightarrow$ сложно попасть в минимум
- маленький lr $\Rightarrow$ сложно дойти до минимума
- хорошая идея: сначала большая скорость, затем меньше

Learning rate scheduling

- линейный: $\eta = \eta_0(1 - \frac{t}{T})$
- экспоненциальный: $\eta = \eta_0 e^{-\frac{t}{T}}$
- циклический: `https://arxiv.org/pdf/1506.01186.pdf`

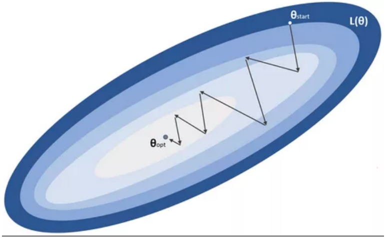
Когда уменьшать learning rate?

- каждую итерацию
- каждую эпоху
- при ухудшении валидационного лосса

Проблемы предложенных подходов

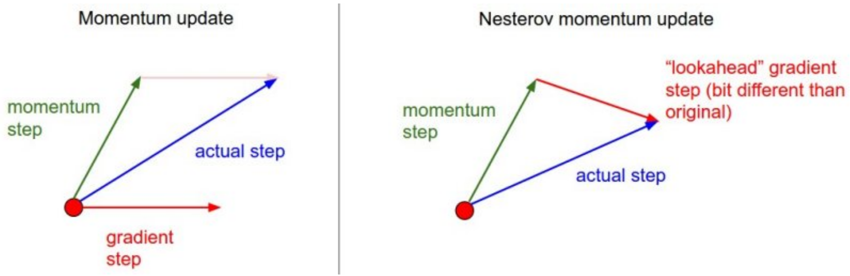
- у нас стало больше параметров, которые нужно подбирать!
- мы никак не учитываем функцию, которую оптимизируем

Momentum (метод импульсов)



$$u_t = \gamma u_{t-1} + \eta \nabla_\theta E(\theta_t)$$
$$\theta_{t+1} = \theta_t - u_t$$

Nesterov Momentum



$$u_t = \gamma u_{t-1} + \eta \nabla_\theta E(\theta_t - \gamma u_{t-1})$$

Методы высших порядков: метод Ньютона

$$E(\theta) \approx E(\theta_t) + \nabla_\theta E(\theta_t)(\theta - \theta_t) + \frac{1}{2}(\theta - \theta_t)^T H(E(\theta))(\theta - \theta_t)$$
$$u = -[H(E(\theta))]^{-1} \nabla E(\theta)$$

Адаптивные методы

- пока скорость обучения не была «персонализирована» по параметрам
- адаптивные метод одни параметры обновляют часто, другие нет

Adagrad

$$g_{t,i} = \nabla_{\theta_i} L(\theta)$$
$$G_{t,i} = G_{t-1,i} + g_{t,i}^2$$
$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,i} + \varepsilon}} \cdot g_{t,i}$$

- η не играет особой роли: знаменатель править скорость обучения
- G только растёт, значит скорость обязательно падает

RMSProp и Adadelta

$$G_{t,i} = \rho G_{t-1,i} + (1-\rho) g_{t,i}^2$$
$$U_{t,i} = \rho U_{t-1,i} + (1-\rho) u_{t,i}^2$$
$$u_{t+1} = -\sqrt{U_t} \cdot \frac{\eta}{\sqrt{G_t + \varepsilon}} \cdot g_t$$

- давайте хранить экспоненциальное среднее, вместо всей суммы (RMSProp)
- давайте домножим на среднее изменение параметров, для получения «правильных» размерностей (Adadelta использует обе идеи)

Adam

$$m_t = \beta_1 m + (1 - \beta_1) g_t$$
$$v_t = \beta_2 m + (1 - \beta_2) g_t^2$$
$$u_t = \frac{\eta}{\sqrt{v_t + e}} m_t$$

- вместо градиентам используем сглаженную версию
- сейчас — выбор по умолчанию
- можно не подбирать lr (шутка)
  https://arxiv.org/abs/1712.07628

Итоги

- скорость обучения полезно изменять
- momentum ускоряет сходимость в большинстве практических случаев
- методы высших порядков (пока) неприменимы в нейронных сетях
- для начала стоит использовать `adam`, потом потюнить `sgd + momentum`

Итоги

- рассмотрели устройство современных нейронных сетей
- рассмотрели «ингредиенты» обучения нейронных сетей: регуляризацию, инициализацию весов, batchnorm, адаптивные оптимизаторы

Литература

- С. Николенко, А. Кадурин, Е. Архангельская — Глубокое обучение (главы 3-4)