Лекция 3. Функции активации. Инициализация весов.

Глубинное обучение

Антон Кленицкий

Recap

Stochastic Gradient Descent

Stochastic gradient descent (SGD) - обновляем веса после каждого примера

$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta L(y_i, f(x_i, \theta))$$

- Несмещенная оценка полного градиента
- Если берем примеры из обучающей выборки в случайном порядке!

Mini-batch stochastic gradient descent - обновляем веса после батча из B примеров

$$\theta_{t+1} = \theta_t - \alpha \frac{1}{B} \sum_{i=1}^{B} \nabla_\theta L(y_i, f(x_i, \theta))$$

- Можно эффективно использовать матричные вычисления

Stochastic Gradient Descent

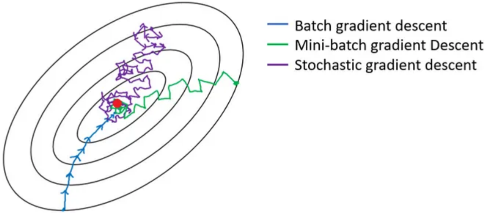


Image credit

- Шумная оценка полного градиента
- Может проскочить мимо неудачного локального минимума или седловой точки

Backpropagation

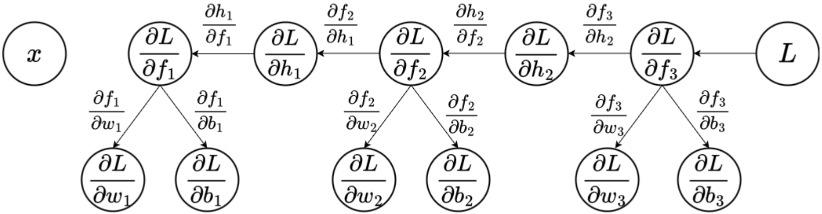
$$\hat{y} = w_3\sigma(w_2\sigma(w_1x + b_1) + b_2) + b_3; \qquad L(y, \hat{y}) = (y - \hat{y})^2$$

$$x \to f_1 \to h_1 \to f_2 \to h_2 \to f_3 \to L$$

$$f_1 = w_1x + b_1$$
$$h_1 = \sigma(f_1)$$
$$f_2 = w_2h_1 + b_2$$
$$h_2 = \sigma(f_2)$$
$$f_3 = w_3h_2 + b_3$$
$$L = (f_3 - y)^2$$

$$\frac{\partial L}{\partial f_3} = 2(f_3 - y)$$
$$\frac{\partial L}{\partial h_2} = \left(\frac{\partial L}{\partial f_3}\right)\frac{\partial f_3}{\partial h_2}$$
$$\frac{\partial L}{\partial f_2} = \left(\frac{\partial L}{\partial f_3}\frac{\partial f_3}{\partial h_2}\right)\frac{\partial h_2}{\partial f_2}$$
$$\frac{\partial L}{\partial h_1} = \left(\frac{\partial L}{\partial f_3}\frac{\partial f_3}{\partial h_2}\frac{\partial h_2}{\partial f_2}\right)\frac{\partial f_2}{\partial h_1}$$
$$\frac{\partial L}{\partial f_1} = \left(\frac{\partial L}{\partial f_3}\frac{\partial f_3}{\partial h_2}\frac{\partial h_2}{\partial f_2}\frac{\partial f_2}{\partial h_1}\right)\frac{\partial h_1}{\partial f_1}$$

Backpropagation



$$\frac{\partial L}{\partial f_3} = 2(f_3 - y)$$

$$\frac{\partial L}{\partial h_2} = \left(\frac{\partial L}{\partial f_3}\right)\frac{\partial f_3}{\partial h_2}$$

$$\frac{\partial L}{\partial f_2} = \left(\frac{\partial L}{\partial f_3}\frac{\partial f_3}{\partial h_2}\right)\frac{\partial h_2}{\partial f_2}$$

$$\frac{\partial L}{\partial h_1} = \left(\frac{\partial L}{\partial f_3}\frac{\partial f_3}{\partial h_2}\frac{\partial h_2}{\partial f_2}\right)\frac{\partial f_2}{\partial h_1}$$

$$\frac{\partial L}{\partial f_1} = \left(\frac{\partial L}{\partial f_3}\frac{\partial f_3}{\partial h_2}\frac{\partial h_2}{\partial f_2}\frac{\partial f_2}{\partial h_1}\right)\frac{\partial h_1}{\partial f_1}$$

$$\frac{\partial L}{\partial w_k} = \left(\frac{\partial L}{\partial f_k}\right)\frac{\partial f_k}{\partial w_k} = \frac{\partial L}{\partial f_k}\cdot h_{k-1}$$

$$\frac{\partial L}{\partial b_k} = \left(\frac{\partial L}{\partial f_k}\right)\frac{\partial f_k}{\partial b_k} = \frac{\partial L}{\partial f_k}\cdot 1$$

Backpropagation summary

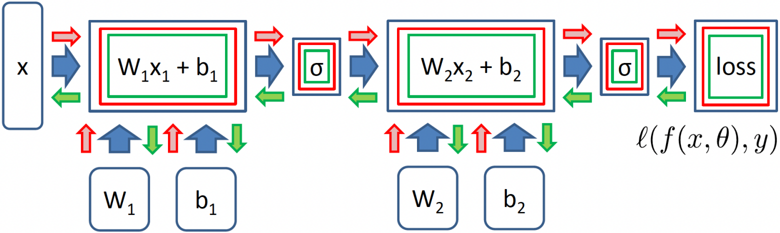


Image credit

- Нейросеть - граф вычислений
- Градиенты распространяются в обратную сторону по графу
- Умножаем на производную выхода по входу для каждого узла графа

Vanishing / exploding gradients

В процессе обратного распространения по сети градиенты могут «затухать» или «взрываться»

$$\frac{\partial L}{\partial f_1} = \left(\frac{\partial L}{\partial f_3}\frac{\partial f_3}{\partial h_2}\frac{\partial h_2}{\partial f_2}\frac{\partial f_2}{\partial h_1}\right)\frac{\partial h_1}{\partial f_1}$$

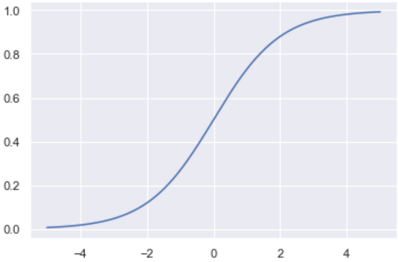
Главное, чтобы градиент хорошо «протекал» по нейросети

Функции активации

Sigmoid

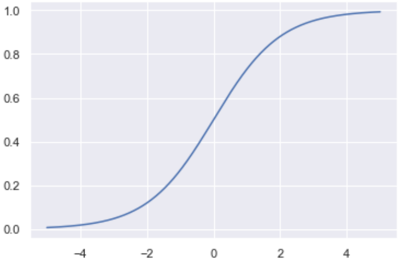
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

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



Sigmoid - проблемы

$\sigma(x)$



$\sigma'(x)$

- Насыщение на краях, градиент близок к нулю
- Максимальное значение производной 0.25
- Выход не центрирован вокруг нуля

Использование сигмоиды

По-прежнему используется в отдельных слоях, например:

- На выходе модели для бинарной классификации
- В гейтах (например, LSTM/GRU)

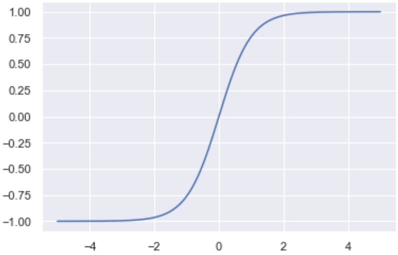
$$h_t = \sigma * h_{t-1} + (1 - \sigma) * h'_t$$

tanh

Гиперболический тангенс

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

$$\tanh(x) = 2\sigma(2x) - 1$$



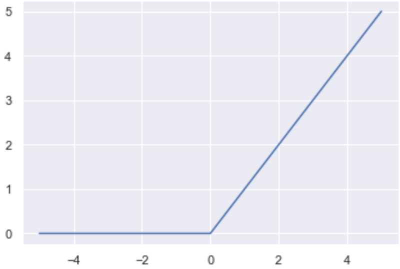
- Максимальное значение производной 1
- Выход центрирован, в нуле значение 0
- По-прежнему насыщение на краях

ReLU

Rectified Linear Unit

$$f(x) = \max(0, x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

$$f'(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$



Плюсы

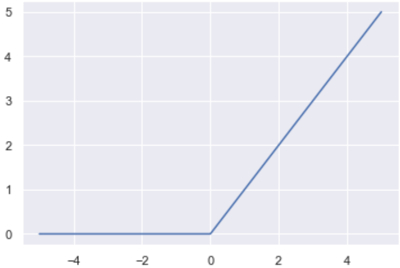
- Нет насыщения, ненулевые градиенты
- Вычислительно очень дешево
- Есть биологические обоснования (разреженность)

ReLU

Rectified Linear Unit

$$f(x) = \max(0, x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

$$f'(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$



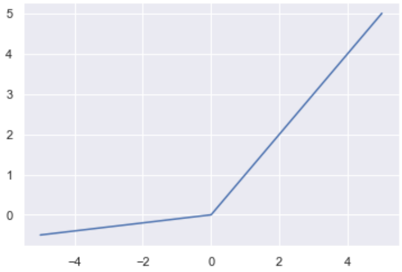
Минус

- "Dead neurons" - нейроны, для которых всегда $x < 0$
- Проблема может усугубляться при большом learning rate

Модификации ReLU

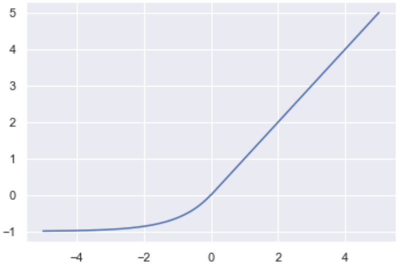
Leaky ReLU, PReLU

$$f(x) = \begin{cases} x, & x \geq 0 \\ \alpha x & x < 0 \end{cases}$$



ELU

$$f(x) = \begin{cases} x, & x \geq 0 \\ \alpha(e^x - 1), & x < 0 \end{cases}$$

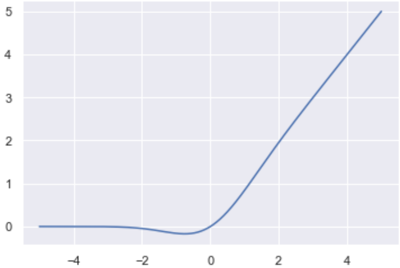


Решаем проблему “Dead neurons”

GELU

Gaussian Error Linear Unit

$$f(x) = xP(X \leq x),$$
$$X \sim N(0, 1)$$



$$f(x) \approx \begin{cases} 0.5x + \frac{1}{2\pi}x^2, & |x| << 1 \\ ReLU(x), & |x| \to \infty \end{cases}$$

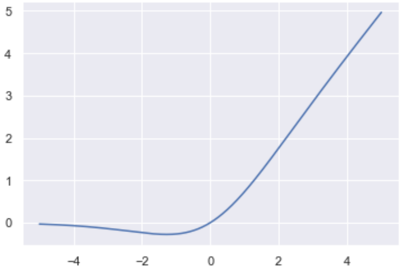
Обычно используется в трансформерах

Swish

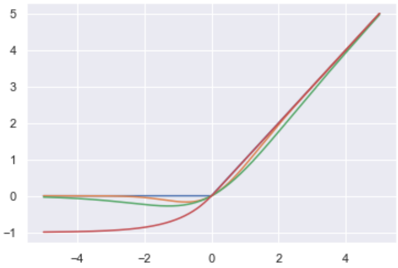
Swish, он же SiLU (Sigmoid Linear Unit)

$$f(x) = x\sigma(x) = \frac{x}{1+e^{-x}}$$

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



Выводы



- Избегать сигмоиды
- ReLU - хороший выбор в большинстве случаев
- Модификации ReLU могут чуть-чуть улучшить качество
- Подбор функции активации - не первый приоритет

Нейросеть с ReLU

$$h_1 = a(\theta_{10} + \theta_{11}x)$$

$$h_2 = a(\theta_{20} + \theta_{21}x)$$

$$h_3 = a(\theta_{30} + \theta_{31}x)$$

$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$$

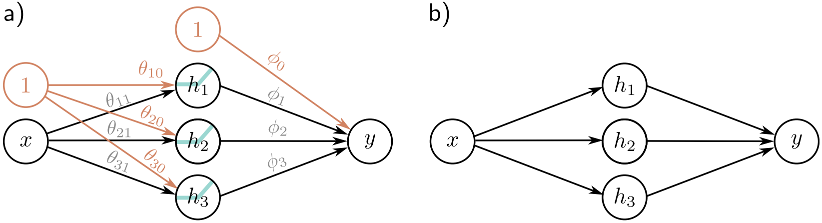


Image credit

Нейросеть с ReLU

Линейное преобразование + ReLU

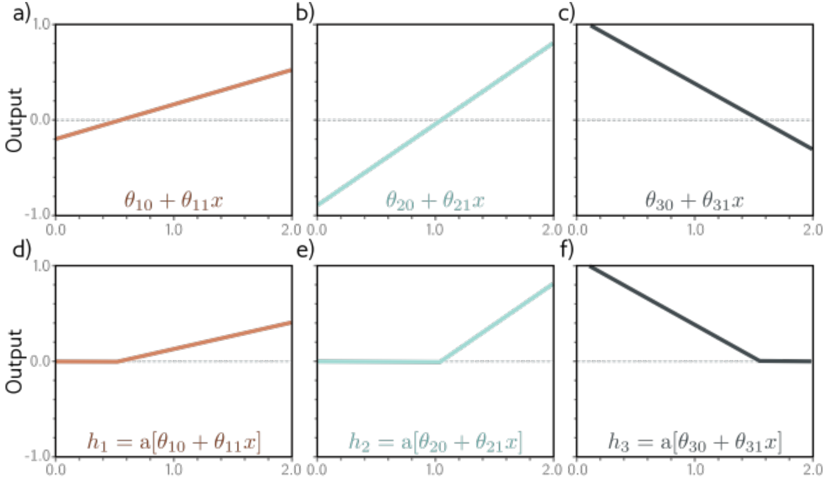


Image credit

Нейросеть с ReLU

Умножение выходов скрытого слоя на веса на выходном слое

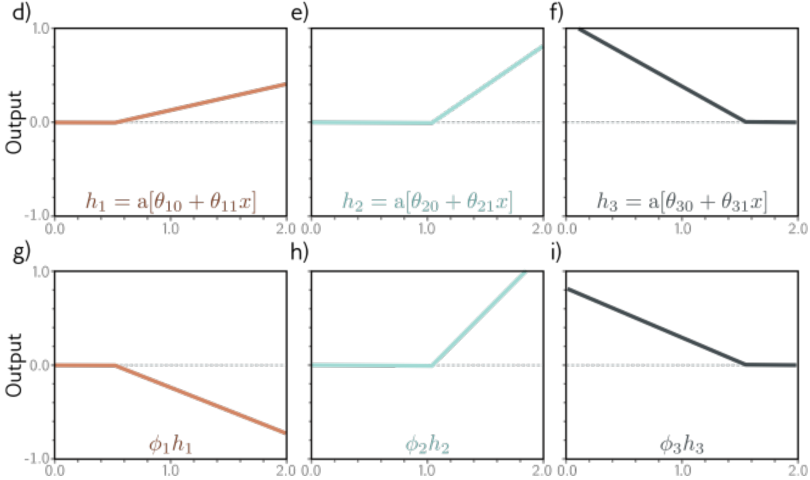
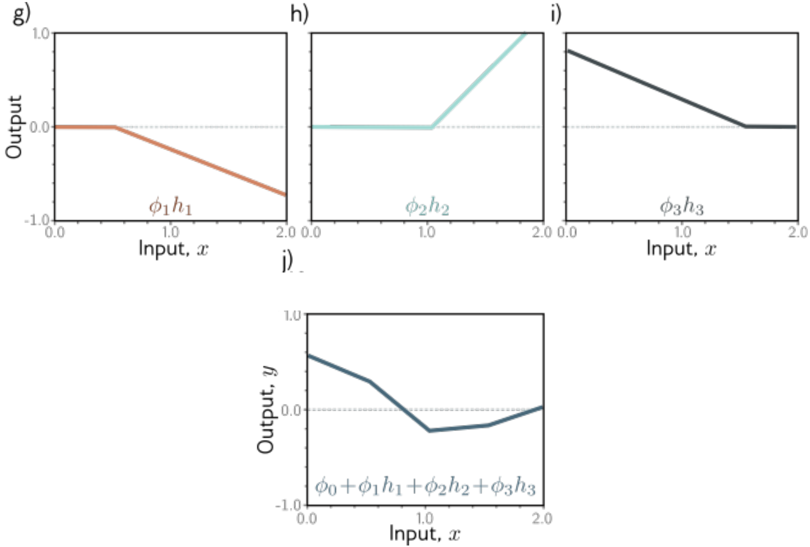


Image credit

Нейросеть с ReLU

Суммирование на выходном слое



Нейросеть с ReLU

Universal approximation theorem

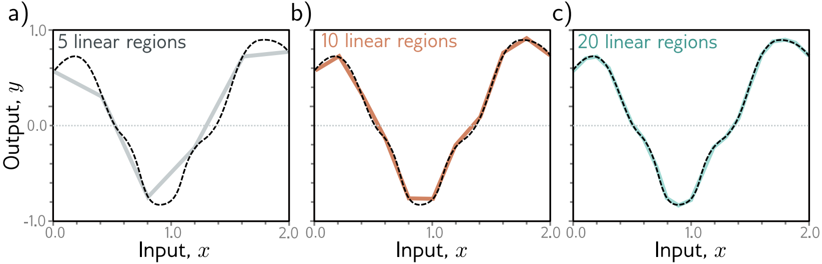


Image credit

Инициализация весов

Инициализация константой

Если попробовать все веса каждого слоя инициализировать одной и той же константой

- При градиентном спуске веса всех нейронов будут меняться одинаково
- Нужно нарушить симметрию между нейронами, чтобы они были разными!

Инициализация случайными значениями

На каждом слое умножаем на матрицу весов и при прямом проходе, и при обратном проходе:

$$f_k = W_k h_{k-1} + b_k$$

$$\frac{\partial f_k}{\partial h_{k-1}} = W_k^T$$

Инициализация случайными значениями

На каждом слое умножаем на матрицу весов и при прямом проходе, и при обратном проходе:

$$f_k = W_k h_{k-1} + b_k$$

$$\frac{\partial f_k}{\partial h_{k-1}} = W_k^T$$

Большие значения весов

- Выходы последних слоев становятся слишком большими
- exploding gradients

Маленькие значения весов

- Выходы последних слоев становятся слишком маленькими
- vanishing gradients

Xavier (Glorot) initialization

Understanding the difficulty of training deep feedforward neural networks - Glorot, X. & Bengio, Y. (2010)

- Для симметричных функций активации
- Идея - дисперсии выходов и градиентов на всех слоях должны быть одинаковыми

Xavier (Glorot) initialization

Understanding the difficulty of training deep feedforward neural networks - Glorot, X. & Bengio, Y. (2010)

- Для симметричных функций активации
- Идея - дисперсии выходов и градиентов на всех слоях должны быть одинаковыми

Рассмотрим один нейрон $y = w^T x = \sum_{i=1}^{n_{in}} w_i x_i$

Xavier (Glorot) initialization

Understanding the difficulty of training deep feedforward neural networks - Glorot, X. & Bengio, Y. (2010)

- Для симметричных функций активации
- Идея - дисперсии выходов и градиентов на всех слоях должны быть одинаковыми

Рассмотрим один нейрон $y = w^T x = \sum_{i=1}^{n_{in}} w_i x_i$

В силу независимости и одинаковой распределенности w_i, x_i

$$Var[y] = Var\left[\sum_{i=1}^{n_{in}} w_i x_i\right] = \sum_{i=1}^{n_{in}} Var[w_i x_i] = n_{in} Var[w_i x_i]$$

Xavier (Glorot) initialization

Understanding the difficulty of training deep feedforward neural networks - Glorot, X. & Bengio, Y. (2010)

- Для симметричных функций активации
- Идея - дисперсии выходов и градиентов на всех слоях должны быть одинаковыми

Рассмотрим один нейрон $y = w^T x = \sum_{i=1}^{n_{in}} w_i x_i$

В силу независимости и одинаковой распределенности w_i, x_i

$$Var[y] = Var\left[\sum_{i=1}^{n_{in}} w_i x_i\right] = \sum_{i=1}^{n_{in}} Var[w_i x_i] = n_{in} Var[w_i x_i]$$

$$Var[w_i x_i] = E[x_i]^2 Var[w_i] + E[w_i]^2 Var[x_i] + Var[w_i] Var[x_i] =$$
$$= Var[w_i] Var[x_i]$$

Xavier (Glorot) initialization

$$Var[y] = n_{in} Var[w_i x_i] = n_{in} Var[w_i] Var[x_i]$$

Xavier (Glorot) initialization

$$Var[y] = n_{in} Var[w_i x_i] = n_{in} Var[w_i] Var[x_i]$$

Получили условие для forward pass

$$n_{in} Var[w_i] = 1$$

Xavier (Glorot) initialization

$$Var[y] = n_{in} Var[w_i x_i] = n_{in} Var[w_i] Var[x_i]$$

Получили условие для forward pass

$$n_{in} Var[w_i] = 1$$

Можно получить такое же условие для backward pass

$$n_{out} Var[w_i] = 1$$

Xavier (Glorot) initialization

$$Var[y] = n_{in} Var[w_i x_i] = n_{in} Var[w_i] Var[x_i]$$

Получили условие для forward pass

$$n_{in} Var[w_i] = 1$$

Можно получить такое же условие для backward pass

$$n_{out} Var[w_i] = 1$$

Как объединить?

$$Var[w_i] = \frac{2}{n_{in} + n_{out}}$$

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

(Kaiming) He initialization

Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification - He, K. et al. (2015)

- Для несимметричных функций активации (ReLU)
- Идея - дисперсии выходов или градиентов на всех слоях должны быть одинаковыми

(Kaiming) He initialization

Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification - He, K. et al. (2015)

- Для несимметричных функций активации (ReLU)
- Идея - дисперсии выходов или градиентов на всех слоях должны быть одинаковыми

$$
\begin{aligned}
Var[w_i x_i] &= E[x_i]^2 Var[w_i] + E[w_i]^2 Var[x_i] + Var[w_i] Var[x_i] = \\
&= E[x_i]^2 Var[w_i] + Var[w_i] Var[x_i] \\
&= Var[w_i] \left(E[x_i]^2 + Var[x_i] \right) = Var[w_i] E[x_i^2]
\end{aligned}
$$

(Kaiming) He initialization

Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification - He, K. et al. (2015)

- Для несимметричных функций активации (ReLU)
- Идея - дисперсии выходов или градиентов на всех слоях должны быть одинаковыми

$$
\begin{aligned}
Var[w_i x_i] &= E[x_i]^2 Var[w_i] + E[w_i]^2 Var[x_i] + Var[w_i] Var[x_i] = \\
&= E[x_i]^2 Var[w_i] + Var[w_i] Var[x_i] \\
&= Var[w_i] \left(E[x_i]^2 + Var[x_i] \right) = Var[w_i] E[x_i^2]
\end{aligned}
$$

Для ReLU

$$E[x_i^2] = \frac{1}{2} Var[y_{prev}]$$

$$Var[y] = n_{in} Var[w_i x_i] = \frac{1}{2} n_{in} Var[w_i] Var[y_{prev}]$$

(Kaiming) He initialization

Либо используем условие для forward pass

$$Var[w_i] = \frac{2}{n_{in}}$$

$$w_i \sim N(0, \sqrt{2/n_{in}})$$

или

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

Либо условие для backward pass

$$Var[w_i] = \frac{2}{n_{out}}$$

Ортогональная инициализация

Ортогональные матрицы - повороты и отражения

$$A^T A = I$$

Столбцы (и строки) ортнормированны: $\sum_i A_{ij} A_{ik} = \delta_{ik}$

Не изменяют длины векторов:

$$||Ax|| = ||x||$$

Поэтому градиенты не будут взрываться и затухать

Задачи и функции потерь

Maximum likelihood

Модель $f(x, \theta)$ определяет параметры распределения вероятности $P(y|x)$:

$$P(y|x) = P(y|f(x, \theta))$$

Maximum likelihood

Модель $f(x, \theta)$ определяет параметры распределения вероятности $P(y|x)$:

$$P(y|x) = P(y|f(x, \theta))$$

Функция правдоподобия:

$$L(\theta) = P(D|\theta) = \prod_{i=1}^{N} P(y_i|f(x_i, \theta))$$

Maximum likelihood

Модель $f(x, \theta)$ определяет параметры распределения вероятности $P(y|x)$:

$$P(y|x) = P(y|f(x, \theta))$$

Функция правдоподобия:

$$L(\theta) = P(D|\theta) = \prod_{i=1}^{N} P(y_i|f(x_i, \theta))$$

Метод максимального правдоподобия (maximum likelihood estimation):

$$\theta = \underset{\theta}{\operatorname{argmax}}\, L(\theta) = \underset{\theta}{\operatorname{argmax}} \left[\prod_{i=1}^{N} P(y_i|f(x_i, \theta))\right]$$

Maximum likelihood

$$\theta = \underset{\theta}{\operatorname{argmax}} \left[\log \left(\prod_{i=1}^{N} P(y_i | f(x_i, \theta)) \right) \right]$$

Maximum likelihood

$$\theta = \underset{\theta}{\operatorname{argmax}} \left[\log \left(\prod_{i=1}^{N} P(y_i | f(x_i, \theta)) \right) \right]$$

$$= \underset{\theta}{\operatorname{argmax}} \left[\sum_{i=1}^{N} \log \left(P(y_i | f(x_i, \theta)) \right) \right]$$

Maximum likelihood

$$\theta = \underset{\theta}{\operatorname{argmax}} \left[\log \left(\prod_{i=1}^{N} P(y_i | f(x_i, \theta)) \right) \right]$$

$$= \underset{\theta}{\operatorname{argmax}} \left[\sum_{i=1}^{N} \log \left(P(y_i | f(x_i, \theta)) \right) \right]$$

$$= \underset{\theta}{\operatorname{argmin}} \left[- \sum_{i=1}^{N} \log \left(P(y_i | f(x_i, \theta)) \right) \right]$$

Negative log-likelihood loss:

$$\mathcal{L} = - \sum_{i=1}^{N} \log \left(P(y_i | f(x_i, \theta)) \right)$$

Регрессия

Предположим, что шум распределен нормально с центром в нуле:

$$y = f(x, \theta) + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

то есть

$$\begin{aligned} P(y|x, \theta, \sigma) &= \mathcal{N}(y|f(x, \theta), \sigma^2) \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} exp\left[-\frac{(y - f(x, \theta))^2}{2\sigma^2}\right] \end{aligned}$$

Регрессия

Предположим, что шум распределен нормально с центром в нуле:

$$y = f(x, \theta) + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

то есть

$$\begin{aligned} P(y|x, \theta, \sigma) &= \mathcal{N}(y|f(x, \theta), \sigma^2) \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} exp\left[-\frac{(y - f(x, \theta))^2}{2\sigma^2}\right] \end{aligned}$$

$$\begin{aligned} \mathcal{L} &= -\sum_{i=1}^{N} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} exp\left[-\frac{(y - f(x, \theta))^2}{2\sigma^2}\right]\right) \\ &= \sum_{i=1}^{N}\left[-\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) + \frac{(y - f(x, \theta))^2}{2\sigma^2}\right] \rightarrow (y - f(x, \theta))^2 \end{aligned}$$

Получили обычный метод наименьших квадратов

Регрессия

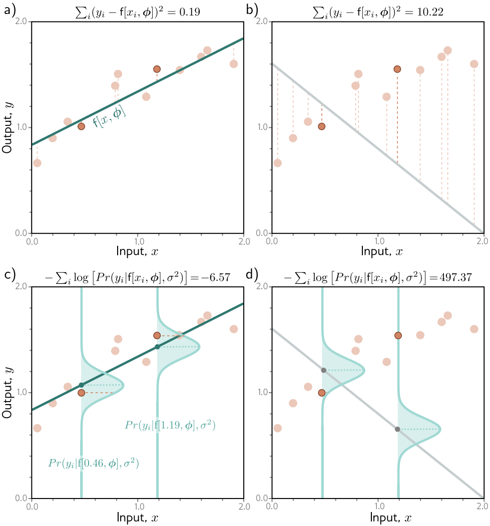


Image credit

Регрессия - другие варианты

- Можем предсказывать и среднее, и дисперсию:

$$\theta = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \left[-\log\left(\frac{1}{\sqrt{2\pi f_2(x,\theta)^2}} \right) + \frac{(y - f_1(x,\theta))^2}{2 f_2(x,\theta)^2} \right]$$

- Если использовать распределение Лапласа, то получим MAE
- Если использовать распределение Пуассона, то можем предсказывать счетчики событий
- Если использовать бета-распределение, то можем предсказывать пропорции

Бинарная классификация

Бинарная классификация

Распределение Бернулли

$$P(y|\lambda) = \lambda^y(1-\lambda)^{1-y} = \begin{cases} \lambda, & y = 1 \\ 1-\lambda, & y = 0 \end{cases}$$

Чтобы оценивать λ - вероятность от 0 до 1, используем сигмоиду:

$$f(x,\theta) = \sigma(x,\theta)$$

$$P(y|f(x,\theta)) = f(x,\theta)^y(1-f(x,\theta))^{1-y}$$

Бинарная классификация

Распределение Бернулли

$$P(y|\lambda) = \lambda^y(1-\lambda)^{1-y} = \begin{cases} \lambda, & y = 1 \\ 1-\lambda, & y = 0 \end{cases}$$

Чтобы оценивать λ - вероятность от 0 до 1, используем сигмоиду:

$$f(x, \theta) = \sigma(x, \theta)$$

$$P(y|f(x,\theta)) = f(x,\theta)^y(1 - f(x,\theta))^{1-y}$$

Функция потерь - binary cross-entropy

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} [-y_i \log f(x_i, \theta) - (1 - y_i) \log(1 - f(x_i, \theta))]$$

Многоклассовая классификация

Softmax:

$$softmax_k(z) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$$

$$P(y = k|f(x, \theta)) = softmax_k\left[f(x, \theta)\right]$$

Многоклассовая классификация

Softmax:

$$softmax_k(z) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$$

$$P(y = k | f(x, \theta)) = softmax_k \left[f(x, \theta) \right]$$

Функция потерь - cross-entropy (aka softmax loss)

$$\mathcal{L}(\theta) = -\sum_{i=1}^{N} \log \left(softmax_{y_i} \left[f(x, \theta) \right] \right)$$

Проблема софтмакса

$$softmax_k(z) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$$

Из-за численного переполнения

- При больших отрицательных значениях z можем получить 0 в знаменателе
- При больших положительных значениях z можем получить бесконечность

Проблема софтмакса

$$softmax_k(z) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$$

Из-за численного переполнения

- При больших отрицательных значениях z можем получить 0 в знаменателе
- При больших положительных значениях z можем получить бесконечность

Трюк:

$$\frac{e^{z_k+c}}{\sum_{j=1}^{K} e^{z_j+c}} = \frac{e^{z_k} e^{c}}{\sum_{j=1}^{K} e^{z_j} e^{c}} = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$$

$$z_k \to z_k - \max_j z_j$$

Какие еще бывают функции потерь

- Hinge loss
- Focal loss
- Dice loss
- Triplet loss
- Contrastive loss

И многие другие..

В следующий раз

- Оптимизация в нейросетях
- Адаптивные методы градиентного спуска
- Методы регуляризации для нейросетей