Нейронные сети

Московский физико-технический институт, МФТИ

Москва

План лекции

1. базовые понятия
2. дифференцирование нейронной сети
3. вычисление градиента, различне оптимизации
4. случайное отключение нейрона dropout
5. различные функции активации
6. начальное приближение
7. прореживание сети

Линейная модель классификации

Пусть $f_j(x)$ $j = 1, .., N$ - числовые признаки, модель оптимизации однойслойного нейрона

$$a(x, w) = \sigma(\langle x, w \rangle) = \sigma\left(\sum_{i=1}^{N} w_j f_j(x) - w_0\right),$$

где $w_0, .., w_N$ -веса признаков.
функция $\text{sign}(z)$, $\sigma(z) = \frac{1}{1+e^{-z}}$.

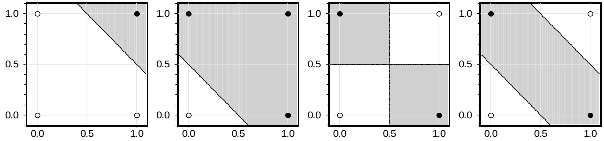
Проблема полноты

Функции И, ИЛИ, НЕ бинарных переменных x^1, x^2

$$x^1 \vee x^2 = [x^1 + x^2 - 3/2 > 0];$$

$$x^1 \wedge x^2 = [x^1 + x^2 - 1/2 > 0];$$

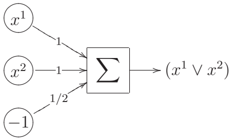
$$\neg x^1 = [-x^1 + 1/2 > 0];$$

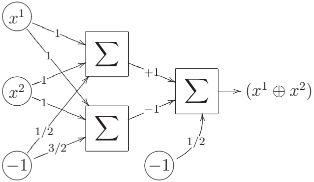


Логическая функция XOR

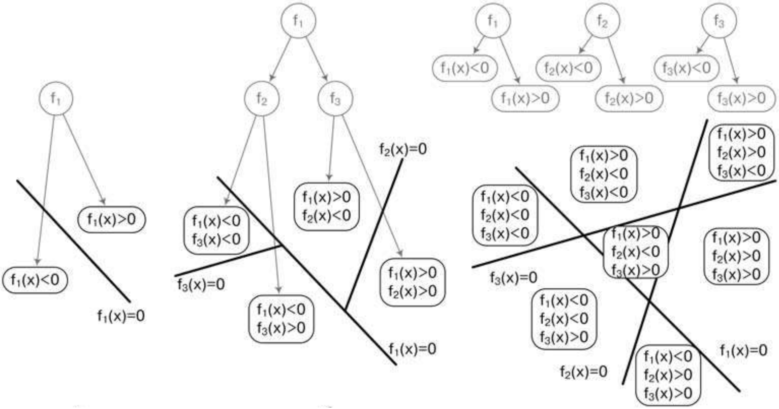
Функция $x^1 \oplus x^2 = [x^1 \neq x^2]$ не реализуема одним нейроном.
Возможные решения вопроса:

- $$x^1 \oplus x^2 = [x^1 + x^2 - 2x^1x^2 - 1/2 > 0];$$
- $$x^1 \oplus x^2 = [(x^1 \vee x^2) - (x^1 \wedge x^2) - 1/2 > 0];$$



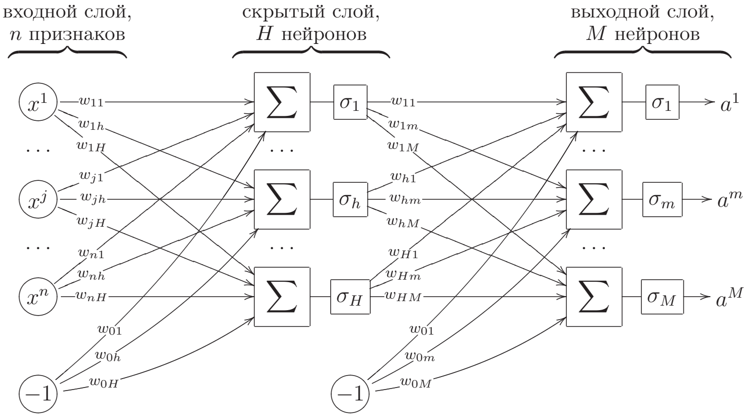


Сила распределенных представлений



Теоретическое обоснование границ применимости нейросетей

- двухслойная сеть позволяет реализовать произвольную булевую функцию,
- двухслойная сеть в $\mathbb{R}^n$ позволяет отделить произвольный выпуклый многогранник,
- трехслоная сеть $\mathbb{R}^n$ позволяет отделить произвольную многоранную область, не обязательно выпоклую, и даже не обязательно связную
- с помощю линейных операций и одной нелинейной функции активации σ можно приблизить любую непрерывную функцию с любой желаемой точностью.



Вектор параметров модели $w = (w_{jh}, w_{hm}) \in R^{Hn+MH+M}$

полносвязная сеть- все со всеми.

Алгоритм стохаостического градиента

Оределим функционал ошибок

$$\mathcal{L}(w) = \frac{1}{l}\sum_{i=1}^{l}\mathcal{L}_i(w) \to \min_w,$$

- выбор случайного обьекта
- вычислить потерю $\mathcal{L}_i = \mathcal{L}_i(w)$
- градиентный шаг $w = w - \eta\mathcal{L}_i'(w)$
- оценить значение функции $Q = (1-\lambda)Q + \lambda\mathcal{L}_i$

Методика обратного дифференцирования

Выходные значения сети на обьекте x_i:

$$a^m(x_i) = \sigma_m \left(\sum_{h=0}^{H} w_{hm} u^h(x_i) \right); \; u^h(x_i) = \sigma_h \left(\sum_{j=0}^{J} w_{jh} f^j(x_i) \right);$$

Без ограничения общности будем рассматривать среднеквадратичную функцию потерь

$$\mathcal{L}_i(w) = \frac{1}{2} \sum_{m=1}^{M} (a^m(x_i) - y_i^m)^2$$

Найти частные производные:

$$\frac{\partial \mathcal{L}_i(w)}{\partial a^m}, \; \frac{\partial \mathcal{L}_i(w)}{\partial u^h}.$$

Метод обратного распространения ошибки (BackProp) 1

Градиент на конечном слое

$$\frac{\partial \mathcal{L}_i(w)}{\partial a^m} = a^m(x_i) - y_i^m = \varepsilon_i^m,$$

тогда производная на выходном слое (функция потерь квадратичная)

$$\frac{\partial \mathcal{L}_i(w)}{\partial u^h} = \sum_{m=1}^{M} \frac{\partial \mathcal{L}_i(w)}{\partial a^m} \sigma'_m() w_{hm} = \sum_{m=1}^{M} \varepsilon_i^m \sigma'_m() w_{hm}$$

BackProp 2

Формулы для
В итоге вычисляем производные по весам $\mathcal{L}_i(w)$

$$\frac{\partial \mathcal{L}_i(w)}{\partial w_{hm}} = \frac{\partial \mathcal{L}_i(w)}{\partial a^m} \frac{\partial a^m}{\partial w_{hm}} = \varepsilon_i^m \sigma'_m() u^h(x_i),$$

$$m = 1..M,\ h = 0..H,$$

$$\frac{\partial \mathcal{L}_i(w)}{\partial w_{jh}} = \frac{\partial \mathcal{L}_i(w)}{\partial u^h} \frac{\partial u^h}{\partial w_{jh}} = \varepsilon_i^h \sigma'_h() f_j(x_i),$$

$$h = 1..H,\ j = 0..n.$$

Алгоритм BackProp 1

1. инициализировать веса w_{jh}, w_{hm}
2. **Цикл**
3. выбрать обьекты x_i из X^l
4. прямые вычисления

$$u^h(x_i) = \sigma_h \left(\sum_{j=0}^{J} w_{jh} f^j(x_i) \right)$$

$$a^m(x_i) = \sigma_m \left(\sum_{h=0}^{H} w_{hm} u^h(x_i) \right)$$

$$\varepsilon_i^m = \frac{\partial \mathcal{L}_i(w)}{\partial a^m}$$

Алгоритм BackProp 2

5. $Q = (1-\lambda)Q + \lambda\mathcal{L}_i(w)$
6. обратные вычисления градиента

$$\varepsilon_i^h = \sum_{m=1}^{M} \varepsilon_i^m \sigma'_m() w_{hm}, h = 1..H$$

7. градиентный шаг

$$w_{hm} = w_{hm} - \eta\varepsilon_i^m \sigma'_m() u^h,$$

$$w_{jh} = w_{jh} - \eta\varepsilon_i^h \sigma'_h() x_i^j,$$

8. **условие выхода из цикла**: пока Q не стабилизируется.

Метод Левенберга-Марквардта

Метод Ньютона-Рафсона

$$w = w - \eta(\mathcal{L}_i''(w))^{-1}\mathcal{L}_i'(w),$$

где $\mathcal{L}_i''(w) = \frac{\partial^2 \mathcal{L}_i(w)}{\partial w_{jk} \partial w_{j'k'}}$ - гессиан.
Возьмем диагональный гессиан

$$w_{jh} = w_{jh} - \eta(\frac{\partial^2 \mathcal{L}_i(w)}{\partial w_{jk}^2} + \mu)^{-1} \partial \mathcal{L}_i(w) \partial w_{jk},$$

Пусть матрица близка к диагональной!
где η - темп обучения,
μ - параметр предотвращающий обнуление
Отношение η/μ - темп обучения на ровных участках функционала, где $\mathcal{L}_i(w)$, где вторая производная обнуляется.

Эврситка для метода SG

- инициализация весов
- порядок предьявления обьектов
- оптимизация величины градиентного шага
- регуляризаця (сокращение весов)

Появляется больше свободы в настройке алгоритма: -выбор функции активации в каждом нейроне
-выбор числа слоев и числа нейронов
-выбор значимых связей

Ускорение сходимости

- Более тщательный выбор начального приближения
  - Либо на случайной подвыборке ;
  - либо по случайному подмножеству входов;
  - либо из различных начальны приближений
- Выбивание из локальных минимумов (jogging of weights)
- Адаптивынй градиентный шаг (скорейший спуск)
- Метод сопряженных градиентов chuking -разбиение суммы $Q(w) = \sum_{l=1}^{l} \mathcal{L}_i(w)$ на блоки (chunks)

Динамическое наращивание сети

- обучение при заведомо небольшом числе нейронов H (мб. скользящий контроль)
- после стабилизации $\mathcal{L}(w)$ - добавление нового нейрона и его оптимизация путем обучения
  - по случайной подвыборке $X' \subseteq X$
  - по обьектам с небольшим значением потерь
  - по случайному поднможеству входов
  - для различных случайных приближениях
- итерации расчета градиента BackProp Время обучения с накоплением в 1.5-2 раза больше, в отличие от сети с фиксированном числом нейронов. Накопленная информация в нейронах сохраняется.

Метод случайных отключений нейронов

- аппроксимируем простое голосование по 2^N сетям с общим набором из N весов, но с различной архитектурой связей
- регуляризация: из всех сетей выбирем более устойчивую к утрате pN нейронов, моделируя надежность мозга
- сокращаем переобучение, заставляя разные части сети решать одну и ту же исходную задачу вместо того, чтобы подстраивать их под компенсацию ошибок друг друга

Прореживание сети (OBD - optimal brain damage)

Пусть w - локальный минимум $\mathcal{L}(w)$, тогда

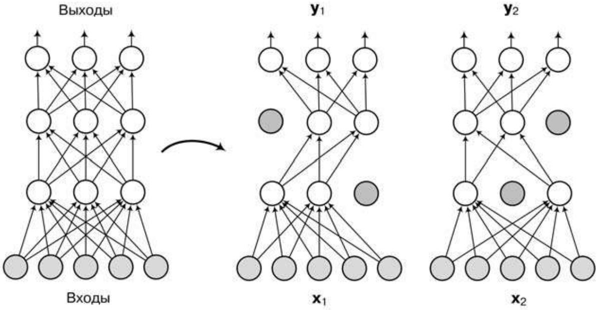
$$Q(w+\delta) = \mathcal{L}(w) + \frac{1}{2}\delta^T Q(w)\delta + o(||\delta||^2),$$

$\mathcal{L}_i''(w) = \frac{\partial^2 \mathcal{L}_i(w)}{\partial w_{jk} \partial w_{j'k'}}$ - гессиан размера $(H(n+1+M+1)+M)^2$,
Пусть гессиан диагонален
Хотим обнулить вес $w_{jh} + \delta_{jh} = 0$. Как изменится $Q(w)$?

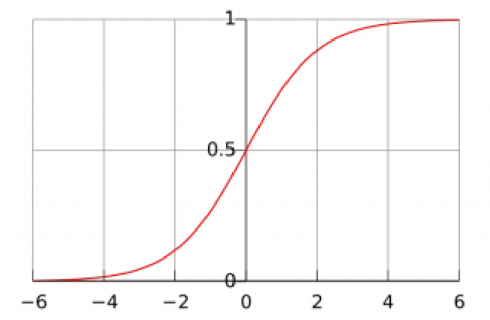
$$\delta^T \partial \mathcal{L}_i(w)\delta = \sum_{j=0}^{n}\sum_{h=1}^{H} \delta_{jh}^2 \frac{\partial^2 \mathcal{L}_i(w)}{\partial w_{jh}^2} + \sum_{h=0}^{H}\sum_{m=0}^{M} \delta_{hm}^2 \frac{\partial^2 \mathcal{L}_i(w)}{\partial w_{hm}^2}.$$

Значимость веса- w_{jh} - это изменение функционала $\mathcal{L}(w)$ при его обнулении $S_{jh} = w_{jh}\frac{\partial^2 \mathcal{L}_i(w)}{\partial w_{jh}^2}$ **Optimal brain surgery**

Dropout

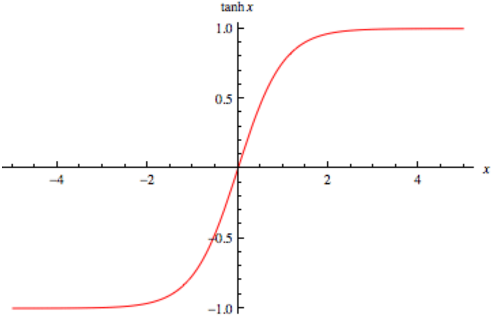


Функция сигмоиды



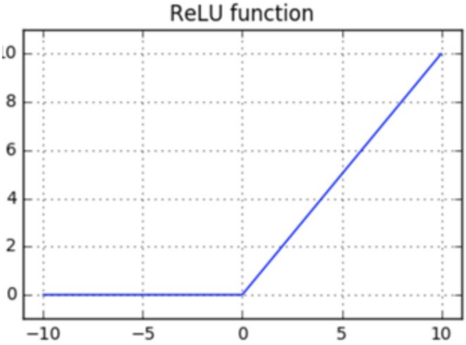
Плюсы: масштабирование, простое дифференцирование
Минусы: нет центрирования, вычислительная сложность, быстрое обнуление в крыльях

Гиперболический тангенс



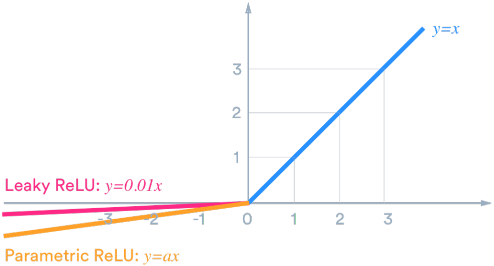
Плюсы:центрирован
Минусы: выч сложность, быстрое обнуление в крыльях

ReLu



Плюсы:легко считается, дает быструю сходимость
Минусы: не центрирован, обнуление

Leaky ReLu



Плюсы:легко считается, дает быструю сходимость, нет обнуления

Различные функции активации

Название функции	**Формула** $f(x)$	**Производная** $f'(x)$
Логистический сигмоид σ	$\frac{1}{1+e^{-x}}$	$f(x)\left(1-f(x)\right)$
Гиперболический тангенс tanh	$\frac{e^x-e^{-x}}{e^x+e^{-x}}$	$1-f^2(x)$
SoftSign	$\frac{x}{1+\lvert x\rvert}$	$\frac{1}{(1+\lvert x\rvert)^2}$
Ступенька (функция Хевисайда)	$\begin{cases} 0, & x<0 \\ 1, & x\geq 0 \end{cases}$	0
SoftPlus	$\log\left(1+e^x\right)$	$\frac{1}{1+e^{-x}}$
ReLU	$\begin{cases} 0, & x<0 \\ x, & x\geq 0 \end{cases}$	$\begin{cases} 0, & x<0 \\ 1, & x\geq 0 \end{cases}$
Leaky ReLU, Parameterized ReLU	$\begin{cases} ax, & x<0 \\ x, & x\geq 0 \end{cases}$	$\begin{cases} a, & x<0 \\ 1, & x\geq 0 \end{cases}$
ELU	$\begin{cases} \alpha\left(e^x-1\right), & x<0 \\ x, & x\geq 0 \end{cases}$	$\begin{cases} f(x)+\alpha, & x<0 \\ 1, & x\geq 0 \end{cases}$

Обобщение

- Нейрон - позволяет решить задачу классификации или регресси
- Нейронная сеть - суперпозиция нейронов с нелинейной функцией активации. (Двух-трех слоев теоретически достаточно для решения любой задачи)
- BackProp - быстрое дифференцирования суперпозиции
- Методы по улучшению сходимости: адаптивный градиентный шаг, функция активации типа ReLu, регуляризация и DropOut, инициалазация нейронов, как отдельных алгоритмов