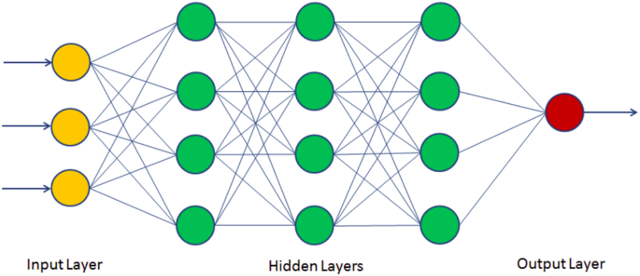
Многослойный персептрон

Виктор Китов

v.v.kitov@yandex.ru



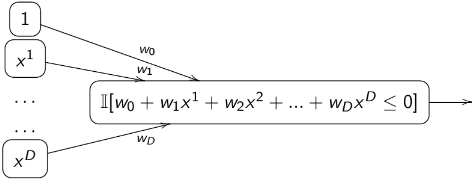
Содержание

История

- Нейронные сети появились как попытка моделировать работу человеческого мозга.

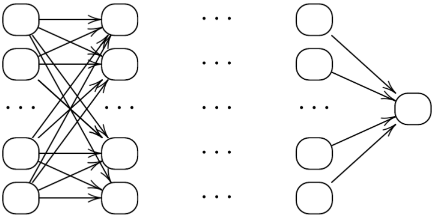
- Человеческий мозг состоит из взаимосвязанных нейронов.
  - порядка 86 миллиардов нейронов
  - нейроны связаны аксонами - вытянутыми отростками нервных клеток
  - взаимодействие нейронов осуществляется электро-химическими сигналами по аксонам

Простая модель нейрона



- Несколько входов посылают сигналы, которые домножаются на вес связи
- Нейрон принимает суммарный сигнал
- Нейрон активируется в полупространстве $w_0 + w_1x^1 + w_2x^2 + ... + w_Dx^D \leq 0$.
- w_0 отвечает за смещение

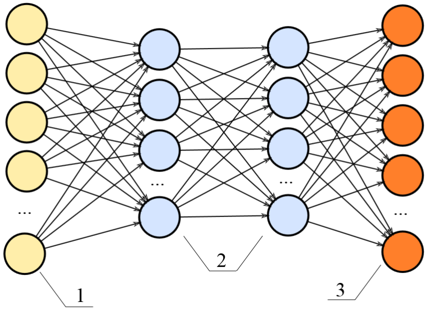
Архитектура многослойного персептрона



многослойный персептрон - ациклический направленный граф

- Несколько слоев, связи между соседними слоями - каждый с каждым.
- Каждый нейрон имеет свои собственные связи.

Слои



- Слои многослойного персептрона:
    - 1-входной слой (не учитывается в полном количестве слоев сети)
    - 2-скрытые слои
    - 3-выходной слой

Многослойный персептрон и ансамбли

- В стэкинге фиксируются базовые модели при настройке агрегирующей ф-ции.
- В бустинге фиксируются предыдущие базовые модели.
- В многослойном персептроне ранние и поздние нейроны настраиваются одновременно.
    - более сильное переобучение

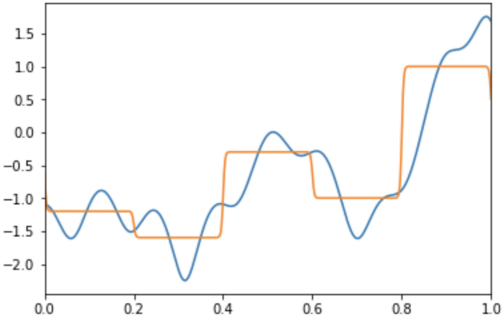
Содержание

Одномерная регрессия

- 1-мерная регрессия:

$$f(x) = \sum_i f(b_i)\mathbb{I}[x \in [a_i, b_i]] = \sum_i f(b_i)\left(\mathbb{I}[x \le b_i] - \mathbb{I}[x \le a_i]\right)$$

$$= \sum_i f(b_i)\mathbb{I}[x \le b_i] - \sum_i f(b_i)\mathbb{I}[x \le a_i]$$ 2-х слойный персептрон



Многомерная регрессия

- AND/OR функции для $x_1, x_2 \in \{0,1\}$ можно сделать 1 слойным персептроном:[1]:

$$\text{AND function} \mathbb{I}[x_1 + x_2 \geq 2] = \mathbb{I}[-x_1 - x_2 \leq -2]$$
$$\text{OR function} \mathbb{I}[x_1 + x_2 \geq 1] = \mathbb{I}[-x_1 - x_2 \leq -1]$$

- **D-мерная регрессия:**
  - один слой приближает линейную ф-цию
    - 2-х слойный персептрон моделирует индикаторную ф-цию на произвольном выпуклом многоугольнике (через AND)
    - 3-х слойный персептрон приближает произвольную непрерывную функцию (Липшицеву) (как взвешенную сумму индикаторов выпуклых многоугольников)
- Таким образом, 3-х слоев достаточно для приближения всех регулярных зависимостей.

[1] How to make XOR (exclusive OR) function?

Классификация

- **Классификация:**
  - один слой выделяет полупространства
  - 2-х слойный персептрон моделирует индикаторную ф-цию на произвольном выпуклом многоугольнике (через AND)
    - приближает произвольное выпуклое множество
  - 3-х слойный персептрон выделяет произвольный многоугольник (через OR) как объединение выпуклых многоугольников
- Таким образом, 3-х слоев достаточно для приближения любого множества.

Выбор числа слоёв

- Зачем использовать больше 3-х слоёв?
- 3-х слойные сети способны приближать любые регулярные зависимости, но может потребоваться слишком много нейронов - переобучение.
- Более глубокие слои могут переиспользовать ранние нейроны.
    - нужно меньше нейронов, меньше связей, меньше переобучение

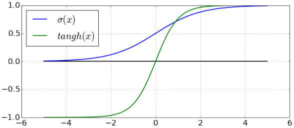
Содержание

Непрерывные активации

- $\mathbb{I}[w^T x - w_0 \le 0]$ - кусочно-постоянная, производная=0, не можем оптимизировать веса.
- Заменим $\mathbb{I}[w^T x - w_0 \le 0]$ непрерывной функцией активации $\phi(w^T x - w_0)$.

Основные функции активации

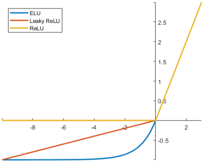
- сигмоида: $\sigma(x) = \frac{1}{1+e^{-x}}$
  - 1 нейрон с сигмоидой моделирует логистическую регрессию
- гиперболический тангенс: $\text{tangh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = 2\sigma(2x) - 1$
  - преимущество: если $\mathbb{E}x = 0$, то $\mathbb{E}\,\text{tangh}(x) = 0$.



- Проблема: $\phi'(x) \approx 0$ вне интервала (-3,3).

Основные функции активации

- Rectified linear unit (ReLU): $\phi(x) = \max(0, x)$
  - аналог с гладкой производной - SoftPlus: $\phi(x) = \ln(1 + e^x)$
- Leaky ReLU: $\phi(x) = \begin{cases} x, & x \geq 0 \\ 0.01x, & x < 0 \end{cases}$
- Parametric ReLU (α оценивается): $\phi(x|\alpha) = \begin{cases} x, & x \geq 0 \\ \alpha x, & x < 0 \end{cases}$
- Exponential LU (α задано): $\phi(x) = \begin{cases} x, & x \geq 0 \\ \alpha(e^x - 1), & x < 0 \end{cases}$



Содержание

Регрессия

- Регрессия: $\phi(I) = I$
- Скалярная регрессия $y \in \mathbb{R}$:

$$MSE(x, y) = \frac{1}{N}\sum_{n=1}^{N}(\widehat{y}(\mathbf{x}_n) - y_n)^2$$

$$MAE(x, y) = \frac{1}{N}\sum_{n=1}^{N}|\widehat{y}(\mathbf{x}_n) - y_n|$$

- Векторная регрессия $\mathbf{y} \in \mathbb{R}^K$:

$$MSE(x, y) = \frac{1}{N}\sum_{n=1}^{N}\|\widehat{\mathbf{y}}(\mathbf{x}_n) - \mathbf{y}_n\|_2^2$$

Классификация, вероятности классов

- **Бинарная классификация**: $y \in \{0, 1\}$

$$p(y = +1|x) = \frac{1}{1 + e^{-l}}$$

$$\mathcal{L}(x, y) = -\ln p(y|x) = -\ln p(y = 1|x)^{\mathbb{I}[y=1]}[1 - p(y = 1|x)]^{\mathbb{I}[y \neq 1]}$$

- **Многоклассовая классификация**: $y \in 1, 2, ... C$

$$\{SoftMax(l_1, ... l_C)\}_j = p(y = j|x) = \frac{e^{l_j}}{\sum_{k=1}^{C} e^{l_k}}, j = 1, 2, ... C$$

$$\mathcal{L}(x, y) = -\ln p(y|x) = -\ln \prod_{c=1}^{C} p(y = c|x)^{\mathbb{I}[y=c]}$$

Классификация, рейтинги классов

- **Бинарная классификация**: $y \in \{-1, 1\}$

$$O(x) = \text{отн. предпочтительность положит. класса}$$

$$hinge(x, y) = [1 - yO(x)]_+$$

- **Многоклассовая классификация**: $y \in 1, 2, ...C$:

$$\{O_1(x), ...O_C(x)\} \text{ - рейтинги классов } 1, ...C$$

$$hinge_1(x, y) = \left[\max_{c \neq y} O_c + 1 - O_y\right]_+$$

$$hinge_2(x, y) = \sum_{c \neq y} [O_c + 1 - O_y]_+$$

Содержание

Метод стохастического градиентного спуска (SGD)

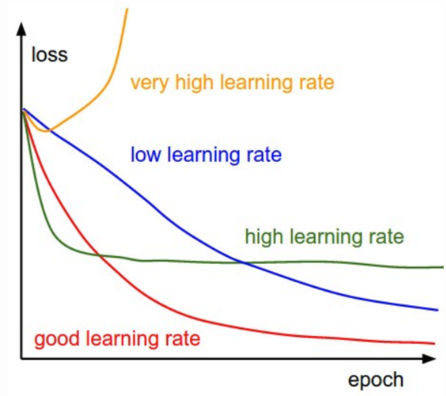
- Обозначим $\mathcal{L}(\widehat{y}, y)$ - функция потерь, $W = \#$[весов], ε_t - шаг оптимизации на шаге t.
- Можем оптимизировать, используя стохастический градиентный спуск:

```
Инициализируем случано w, t = 0

повторять до сходимости:
    сэмплируем случайный объект (x_n, y_n)
    w := w − ε_t ∇_w L(w, x_n, y_n)
  t := t + 1
```

- Можно сэмплировать группу объектов ("минибатч")
- Сходимость: $t > threshold$, $\|L(w_{t+1}) - L(w_t)\| < threshold$, $\|w_{t+1} - w_t\| < threshold$
- Нормализация признаков позволяет ускорить сходимость.

Выбор шага оптимизации важен для сходимости



- На практике часто берут $\varepsilon = const$ и уменьшают при выходе на стабильные потери.
- Формально: сходимость при достаточно медленном $\varepsilon_t \to 0$.

SGD с инерцией (momentum)

```
Инициализируем случано w, t = 0

повторять до сходимости:
    сэмплируем случайный объект (x_n, y_n)
    Δw := αΔw + (1 − α) ∇_w L(w, x_n, y_n)
    w := w − ε_t Δw
    t := t + 1
```

- SGD с инерцией использует сглаженную по всем наблюдениям более точную оценку градиента
  - можем сделать ε_t выше и быстрее сойтись.

- Инерция Нестерова: "заглядывание вперед" при расчете градиента: $\nabla_w \mathcal{L}(w - \varepsilon_t \alpha \Delta w, x_n, y_n)$

Оценка градиента

- Вычисление $\nabla\mathcal{L}(w)$ через разностную аппроксимацию $(\varepsilon_i = [0, ...0, \varepsilon, 0, ...0])$

$$\frac{\partial\mathcal{L}}{\partial w_i} = \frac{\mathcal{L}(w+\varepsilon_i) - \mathcal{L}(w)}{\varepsilon} + O(\varepsilon) \tag{1}$$

  или более точная оценка

$$\frac{\partial\mathcal{L}}{\partial w_i} = \frac{\mathcal{L}(w+\varepsilon_i) - \mathcal{L}(w-\varepsilon_i)}{2\varepsilon} + O(\varepsilon^2) \tag{2}$$

  имеет вычислительную сложность $O(K^2)$, K - #весов
  - нужно посчитать K производных
  - сложность вычисления каждой: $O(K)$

Алгоритм обратного распространения ошибки (backpropagation) требует только $O(K)$ для расчета всех производных.

Пример вычисления градиента[2]

- Рассмотрим бинарную классификацию $y \in \{+1, -1\}$, сеть предсказывает $p(y=+1|x)$:

$$p(y=+1|x) = \frac{1}{1+e^{-w^T x}}$$

- Качество - вероятность истинного класса:

$$S = p(y|x) = \mathbb{I}[y=+1]p(y=+1|x) + \mathbb{I}[y=-1](1-p(y=+1|x))$$

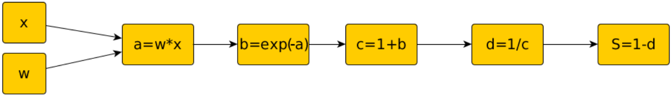
- Предположим $y=-1$, и мы обновляем w, чтобы $1-p(y=+1|x)$ была выше:

$$w := w + \varepsilon\frac{\partial S}{\partial w} = w + \varepsilon\frac{\partial[1-p(y=+1|x)]}{\partial w}$$

[2] Пример BackProp для векторного случая.

Алгоритм обратного распространения ошибки

$$\frac{\partial[1-p(y=+1|x)]}{\partial w}=\frac{\partial}{\partial w}\left[1-\frac{1}{1+e^{-w^Tx}}\right]-?$$



Рассчитаем все промежуточные активации слева-направо за $O(K)$, а производные - справа-налево за $O(K)$:

$$\frac{\partial S}{\partial d}=-1,\ \frac{\partial S}{\partial c}=\frac{\partial S(d(c))}{\partial c}=\frac{\partial S}{\partial d}\frac{\partial d}{\partial c}=\frac{\partial S}{\partial d}\left(-\frac{1}{c^2}\right);\ \frac{\partial S}{\partial b}=$$
$$\frac{\partial S(c(b))}{\partial b}=\frac{\partial S}{\partial c}\frac{\partial c}{\partial b}=\frac{\partial S}{\partial b}\cdot 1;\ \frac{\partial S}{\partial a}=\frac{\partial S(b(a))}{\partial a}=\frac{\partial S}{\partial b}\frac{\partial b}{\partial a}=-\frac{\partial S}{\partial b}e^{-a}$$
$$\frac{\partial S}{\partial w}=\frac{\partial S(a(w))}{\partial w}=\frac{\partial S}{\partial a}\frac{\partial a}{\partial w}=\frac{\partial S}{\partial a}x$$

Особенности оптимизации нейросетей

- Зависимость $\widehat{y}(x)$ в общем случае невыпукла.
- $\mathcal{L}(\widehat{y}, y)$ - невыпукла => много локальных минимумов.
- На найденный минимум влияют:
    - начальное приближение
    - объекты минибатчей
    - метод обучения и динамика ε_t
- Можно настраивать разными способами, а потом
    - выбрать наилучшее решение по валидации
    - усреднить несколько решений

Содержание

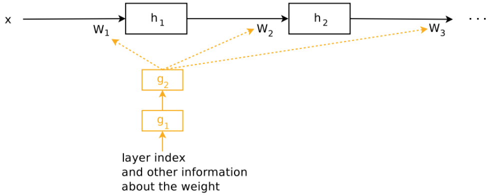
Особые применения

Нейросети (как и др. модели) могут предсказывать не точки, а плотности распределений:

- предсказывать гистограмму из K столбцов.
- предсказывать параметры семейства, например μ, Σ для $y|x \sim \mathcal{N}(\mu(x), \Sigma(x))$

Гиперсеть[4]

- Гиперсеть (hypernetwork) генерирует параметры θ для другой сети $F_\theta(x)$:



- Настройка: $\theta = g\,(layer, ...)$, $\nabla\mathcal{L}(F_\theta(x), y)$ меняет параметры только гиперсети $g\,(\cdot)$.
- Пример: для разных подклассов задачи использовать разные параметры основной сети. Гиперсеть их подстраивает, в зависимости от подзадачи[3].

[3] https://arxiv.org/pdf/1906.00695.pdf
[4] https://arxiv.org/pdf/1609.09106.pdf

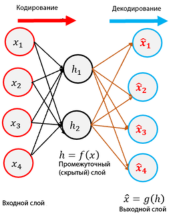
Автокодировщик

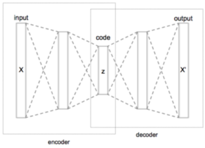
- Автокодировщик - архитектура, пытающаяся восстанавливать вход:

$$\widehat{x} = g(f(x)), \quad \widehat{x} \approx x$$

  - $f(x)$: кодировщик
  - $h = f(x)$: компактное промежуточное представление
  - $g(h) = g(f(x))$: декодировщик
- Мотивация:
  - компрессия данных: $dim(h) < dim(x)$
  - снижение размерности, анализ данных в 2D, 3D.
  - извлечение информативных признаков $x \to h$ для др. задач
    - регрессия, классификация, ранжирование и др.
  - инициализация первых слоев др. архитектуры кодировщиком $f(x)$
  - фильтрация шума: сложно воспроизвести нетипичность
  - детекция аномалий: там, где $\|\widehat{x} - x\|$ велико.

Недоопределенный автокодировщик





Недоопределенный автокодировщик

Недоопределенный автокодировщик (Undercomplete Autoencoder)

$$dim(h) < dim(x)$$

$$\mathcal{L}(\widehat{x}, x) = \|\widehat{x} - x\|_2^2$$

f и g могут содержать несколько слоев.

Другие автокодировщики

- **Разреженный автокодировщик** (sparse autoencoder)
  - может иметь $dim(h) \geq dim(x)$
  - компактность h обеспечивается регуляризацией

$$\mathcal{L}(\widehat{x}, x) = \|\widehat{x} - x\|_2^2 + \lambda R(h)$$

например, $R(h) = \|h\|_1$ или $R(h) = \|h\|_2^2$

Другие автокодировщики

- **Разреженный автокодировщик** (sparse autoencoder)
  - может иметь $dim(h) \geq dim(x)$
  - компактность h обеспечивается регуляризацией

$$\mathcal{L}(\widehat{x}, x) = \|\widehat{x} - x\|_2^2 + \lambda R(h)$$

$$\text{например, } R(h) = \|h\|_1 \text{ или } R(h) = \|h\|_2^2$$

- **Фильтрующий автокодировщик** (denoising autoencoder) восстанавливает x по $x + \varepsilon$:

$$\widehat{x} = g(f(x + \varepsilon)), \quad \widehat{x} \approx x$$

  - примеры шума: Гауссов, маскирование нулем, соль и перец.

Другие автокодировщики

- **Разреженный автокодировщик** (sparse autoencoder)
  - может иметь $dim(h) \geq dim(x)$
  - компактность h обеспечивается регуляризацией

$$\mathcal{L}(\widehat{x}, x) = \|\widehat{x} - x\|_2^2 + \lambda R(h)$$

$$\text{например, } R(h) = \|h\|_1 \text{ или } R(h) = \|h\|_2^2$$

- **Фильтрующий автокодировщик** (denoising autoencoder) восстанавливает x по $x + \varepsilon$:

$$\widehat{x} = g(f(x + \varepsilon)), \quad \widehat{x} \approx x$$

  - примеры шума: Гауссов, маскирование нулем, соль и перец.
- **Сжимающий автокодировщик** (contractive autoencoder) выучивает h, устойчивое к малым изменениям

$$\mathcal{L}(\widehat{x}, x) = \|\widehat{x} - x\|_2^2 + \lambda \sum_i \left\| \frac{\partial h_i}{\partial x} \right\|_2^2$$

Заключение

- Нейросети - универсальный аппроксиматор: может моделировать сложные нелинейные зависимости.
- ReLU, LeakyReLU - рекомендуемые функции нелинейности.
- Алгоритм обратного распространения - считает $\nabla\mathcal{L}_w(\widehat{y}, y)$ за линейное отн. #весов время.
- Функция потерь невыпукла, возможны локальные оптимумы.
    - можно искать решение разными способами, а потом выбрать лучшее
- Автокодировщик - способ получить информативные признаки без разметки объектов.