Нейронные сети в машинном обучении

Майер Алексей

Содержание

Лекция 1. Бэкроп, оптимизаторы, PyTorch [ТРЕБУЕТ ДОРАБОТКИ] 2
- Лектор(-ы) 2
- План курса 2
  - Лекции и семинары 2
  - Домашки и система оценивания 2
- Причём тут нейроны? 2
  - История 2
  - Чуть-чуть матеши 2
  - Какая аналогия с Нейронами? 3
  - Как это можно представить? 3
- Виды функций активаций 3
  - Почему сети? 3
- Теория. Теорема Цыбенко. 3
- Как оценить качество имеющегося решения? 3

Лекция 2. [ТРЕБУЕТ ДОРАБОТКИ] 3
- Задача 4
- Что такое робастность? 4
- Переобучение 4
- Взрыв и затухание градиента 4
- Инициализация весов 4
- Регулярцизация нейросетей 5
  - Меняем функцию потерь 5
  - Меняем архитектуру 5
- нормализации 5

Лекция 1. Бэкроп, оптимизаторы, PyTorch [ТРЕБУЕТ ДОРАБОТКИ]

Лектор(-ы)

Батраков Юрий @BatrakovYuri

Евдокимов Егор @ea_evdokimov

План курса

Задумывался чисто про глубокое обучение. Но пришлось добавить ещё и нейронные сети.

Лекции и семинары

1. Бэкроп, оптимизаторы, PyTorch
2.

Домашки и система оценивания

1. Базовые функции pytorch, классификация/регрессия, методы регуляризации **(15)**
2. Сегментация. Ускорение модели. **(10)**
3. NLP. Классификация текстов. Переводчик. NER. **(15)**
4. GAN + VAE. **(15)**
5. Representation learning. **(10)**
6. RecSys. ALS. **(10)**

+ Экзамен **(40)**

Итог: Сумма баллов **(максимум 115)** делённая на 10. Округление математическое.

Сумма баллов за домашки

Причём тут нейроны?

История

Нейроны – штука, которая накапливает сигнал, при активации передаёт дальше. Нейронов очень много, (>80 млрд.)

Как это смоделировать?

Изначально хотели для использования машинного обучения, кодировать слова числами (просто прогнать словарь), но это оказалось неэффективно.

Давайте моделировать нейрон! Используем для этого:

Чуть-чуть матеши

$(x_1, x_2, ..., x_n)$ – векторное представление объекта, n – количество признаков.

$(w_1, w_2, ..., w_n)$ – коэффициенты

b – смещение (bias) («байас» по русски)

Легче представлять, что векторы $(1, x_1, ..., x_n)$ – для удобства умножения f – какая-то функция (функция активации).

Какая функция? Например больше – меньше. Если больше какай-то константы, то активируется (пороговая функция активации).

Какая f функция? Нелинейная, кусочно дифференцируемая.

$$a = f\left(\sum_{j=1}^{n} w_{i,j} x_j + b\right)$$

Какая аналогия с Нейронами?

Как это можно представить?

Теперь у нас m выходов. Тогда выходы можно представить как матричное произведение матрицы весов на вектор.

$$W = \begin{pmatrix} W_{0,1} & \dots & W_{0,m} \\ \dots & \dots & \dots \\ W_{d,1} & \dots & W_{d,m} \end{pmatrix} \in \mathbb{R}^{d+1}$$

x

Виды функций активаций

ACTIVATION FUNCTION	PLOT	EQUATION	DERIATIVE	RANGE	ЗАЧЕМ
Линейная		$f(x) = x$	$f(x) = 1$	$(-\infty, \infty)$	
Бинарный шаг					Для классификации и удобства
Гиперболический тангенс					Для

Почему сети?

В сетях не один, а гораздо больше слоёв.

Теория. Теорема Цыбенко.

Формальное изложение с википедии.

Как оценить качество имеющегося решения?

Вводится несколько функций. Считаем ошибку. Вставить слайды из лекции.

Лекция 2. [ТРЕБУЕТ ДОРАБОТКИ]

Два линейных слоя подряд это плохо, так как их можно объединить в один слой (умножения на 2 матрицы последовательно можно переписать в одну матрицу). Но иногда это может быть полезно, чтобы сузить размер промежуточный матрицы, чтобы она быстрее передавалась по информационному каналу. Бред, но не всегда автор – дурак.

Задача

Пришла Картинка, мы хотим понять. Кошечка это, или собачка?

Как мы это будем делать?

После прохождение через нейронную сеть, мы получаем некоторый вектор, и засовываем его в SoftMax функцию активации. Она

$$\frac{e^{z_i}}{\sum_{j=1}^{k} z_j}$$

Что такое робастность?

Робастность модели – устойчивость метода или модели к шуму, ошибкам и неожиданным ситуациям.

Переобучение

После переобучения метрики могут улучшаться, но на практике такое не встречается.

Взрыв и затухание градиента

Когда мы пропускаем градиенты через обратное распространение ошибки происходит перемножение большого количества чисел. Если эти числа будут большие, то градиент будет увеличиваться и когда back propagation дойдёт до последних слоёв, то мы собьём веса какие у нас были.

Когда градиенты на каждом слое маленькие, то после перемножения кучи маленьких чиссел совсем маленькие и последующие слои меняться совсем не будет

Инициализация весов

1. Инициализация константой (но веса будут меняться одинаково внутри слоёв = плохо)
2. Инициализация случайной величиной (плохо)
3. Xavier Рассмотрим активацию: $y = w \cdot x + b = \sum_i w_i x_i + b$ Обозначим $w_i x_i = y_i$ Тогда дисперсия этой величины $D(y_i) = ... = E[x_i]^2 D(w_i) + E[w_i]^2 + D(x_i) + D(w_i) D(x_i)$

   Получили, что $D(y_i) = D(w_i) D(x_i)$

   (Data science $\neq$ science)

   Сделаем предположение, что x и w сэмплируются независимо из одного распределение, (условно правда), тогда:

   $$D(y) = D\left(\sum_{i=1}^{*} n\right)$$

   Ранее веса инициализировали из распределения:

   w_i

   Xavier (backward Pass…)

Регулярцизация нейросетей

Тезники для борьбы с переобучением, повышением робастности и для получения более подходящего решения с точки зрения эксперта.

Изменяем:
1. Функции потери
2. Структуру сети
3. Процесс оптимизации
4. Данные

Меняем функцию потерь

Меняем функцию потерь, по которой считаем градиент:

$$\mathrm{Loss}_{\mathrm{result}} = \mathrm{L}_{\mathrm{original}} + L_{\mathrm{regularization}}$$

Добавка может быть любой, подходящей под задачу, но обычно используют

$L_1, L_2 - \text{регуляризацию (weight decay)}$

Регуляризатор может быть применен как и к весам модели, так и к активациям / выходам, в зависимости от желаемых целей.

На практике ещё это требуется для добавления свойств, например, чтобы сделать вырожденным распределение (не по одной величине, а по нескольким) для целей компании.

$L_2 -$ регуляризация

L_1 прореживает веса

L_2 также сжимает, но сильнее

Меняем архитектуру

Меняем:
1. Специальные слои (добавляем!) (нормализации и DropOut)
2. Дистилляция
3. Квантизация
4. Круннинг

нормализации