Введение в нейронные сети

Лекция 2. Углубление в НС и Keras

Overview

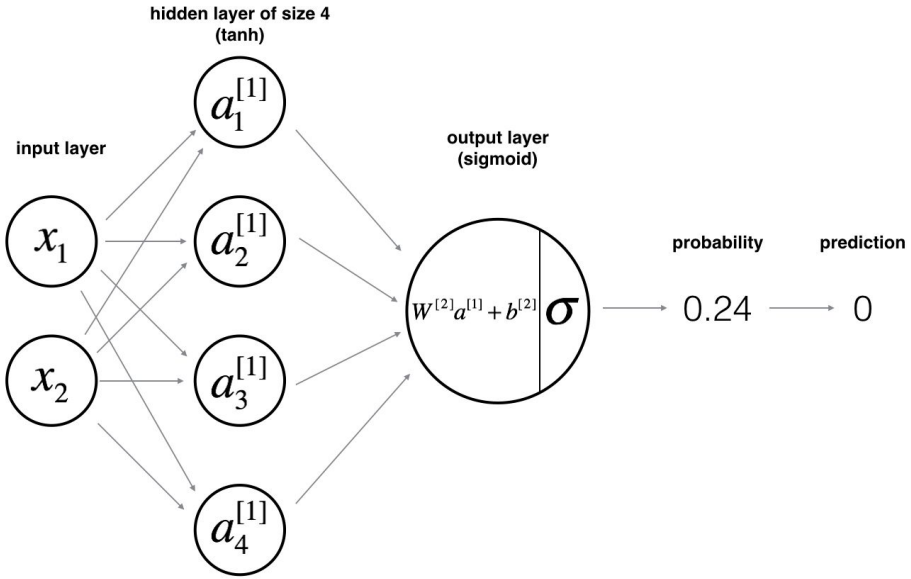
Особенности обучения НС:

- Основные виды функций потерь для различных задач глубокого обучения
- Алгоритмы оптимизации: SGD, momentum GD, RMSProp, Adam
- Проблема переобучения
- Алгоритмы регуляризации (Dropout, Batchnorm, L1, L2-regularization)
- Немного о метриках...

Семинар про Keras:

- Keras functional API
- Сохранение модели на диск
- Keras Callbacks

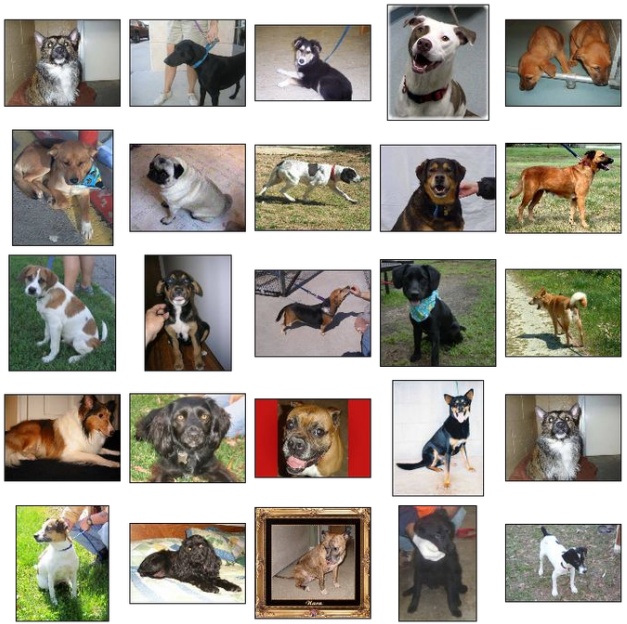
Recap



Однослойная НС. Источник: Stanford University

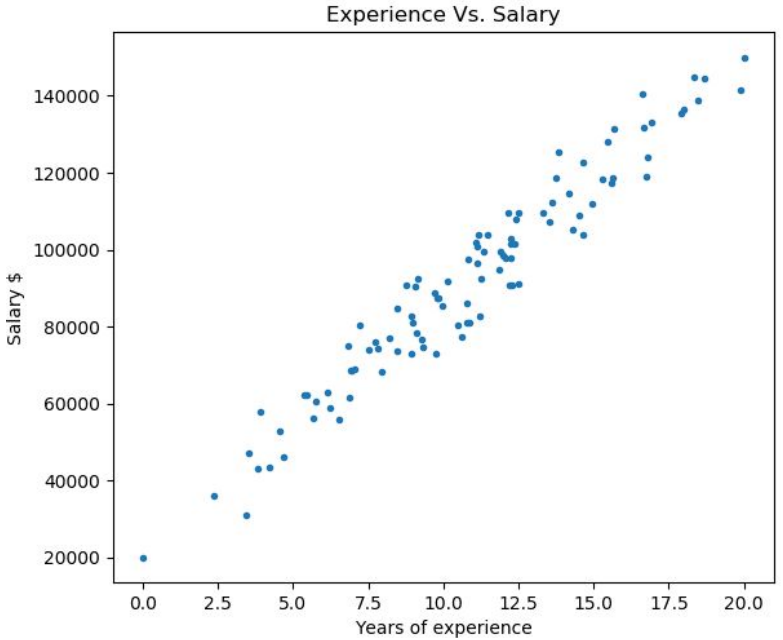
Loss functions

Classification



Dogs VS cats dataset

Regression



Regression. Источник: gaussianwaves.com

Loss functions

Classification

- binary cross-entropy:

$$\mathrm{BCE}(y, \hat{y}) = -\left(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)\right)$$

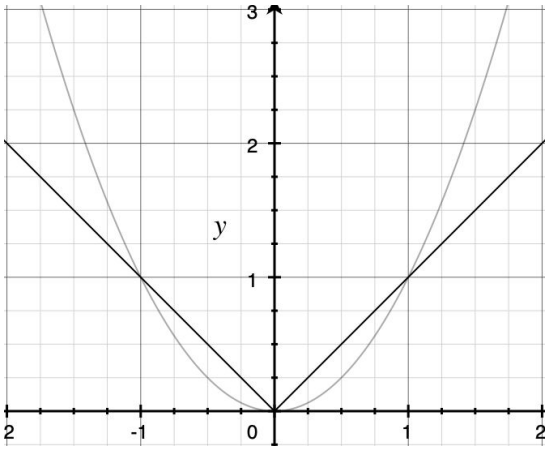
- categorical cross-entropy
- KL-divergence:

$$\mathcal{L}(y, \hat{y}) = y \cdot \log\left(\frac{y}{\hat{y}}\right)$$

Loss functions

Regression:

- MSE
- MAE
- Huber loss



MSE vs MAE

Huber loss - это некий компромисс между MSE и MAE :)

Loss functions. Summary

Самые важные тезисы:

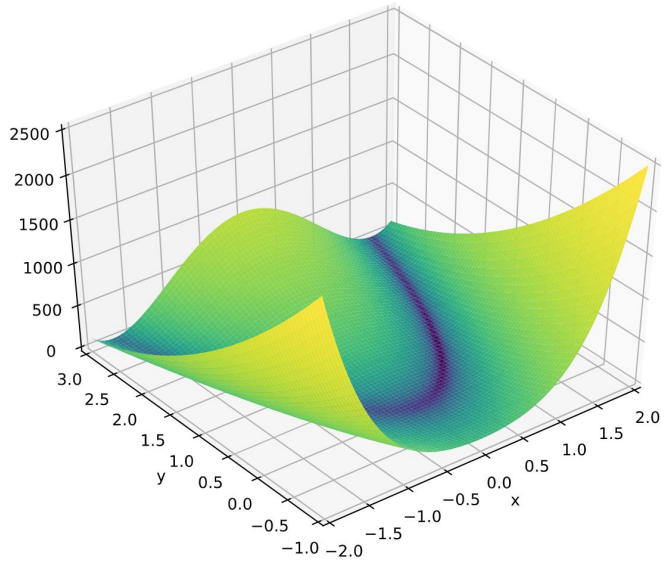
- Выбор Loss function-а определяется постановкой задачи и не зависит (почти всегда) от архитектуры НС,
- Градиент в НС считается от loss function-а по параметрам сети (по весам и смещению)
- Loss function не всегда хорошо интерпретируема, поэтому необходимо использовать контроль за метриками.

Оптимизация в НС. Введение

Функция Розенброка - это невыпуклая функция с одним глобальным минимумом. Она используется для оценки качества алгоритмов оптимизации.

В НС при оптимизации функции потерь существует две основные проблемы:

- седловые точки,
- разные масштабы данных.



Mini-batch gradient descent

Часто при обучении НС мы имеем дело с огромными выборками данных (10-100 ГБ), поэтому не представляется возможным загрузить их в память компьютера. Для решения этой проблемы в современном DL имеют дело с mini-batch gradient descent: градиент подсчитывается на n объектах выборки, после чего обновляются параметры и выбираются следующие n объектов. Такая подвыборка называется **batch,** а алгоритм такой оптимизации **mini-batch gradient descent**.

Скользящее среднее

Скользящее среднее (exponential moving avg) - метод аппроксимации среднего арифметического.

Если задана последовательность Θ, тогда скользящее среднее v можно определить следующим образом (см. формулы справа):

$$v_0 = 0$$

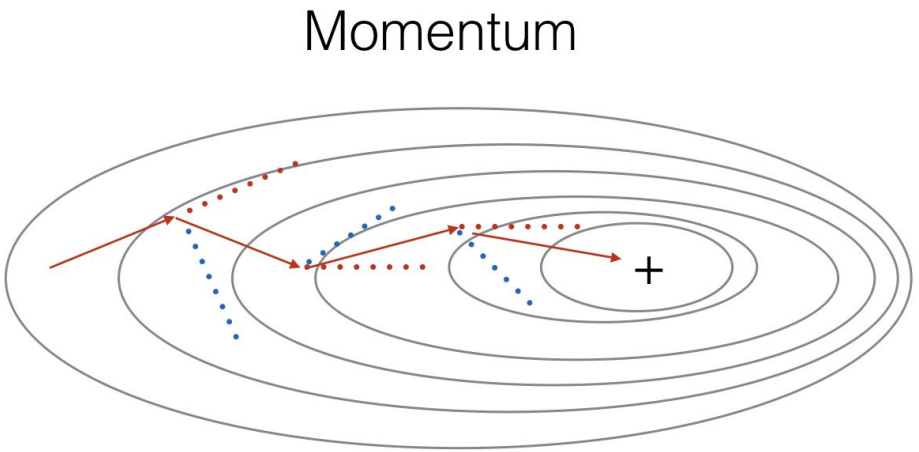
$$v_1 = 0.9 \cdot v_0 + 0.1 \cdot \theta_1$$

$$v_2 = 0.9 \cdot v_1 + 0.1 \cdot \theta_2$$

$$v_t = \beta \cdot v_{t-1} + (1 - \beta) \cdot \theta_t$$

Momentum GD

Идея алгоритма: считать скользящие средние для градиентов, и использовать уже эти векторы, а не сами градиенты для обновления параметров.



Momentum GD. Источник: Stanford University

RMSProp

Алгоритм RMSProp эффективно оптимизирует функции, в которых дисперсия значений по одной оси сильно отличается от дисперсии по другой.

$$S_{dw} = \beta \cdot S_{dw} + (1 - \beta) \cdot (dw)^2$$

$$S_{db} = \beta \cdot S_{db} + (1 - \beta) \cdot (db)^2$$

$$w = w - \frac{\alpha}{\sqrt{S_{dw}}} \cdot \frac{\partial \mathcal{L}}{\partial w}$$

AdaM

AdaM - это алгоритм, комбинирующий и RMSProp, и Momentum GD.
Рассмотрим его реализацию:

1)

$$V_{dw} = \beta_1 \cdot V_{dw} + (1-\beta_1) \cdot dw, \quad S_{dw} = \beta_2 \cdot S_{dw} + (1-\beta_2) \cdot (dw)^2$$
$$V_{db} = \beta_1 \cdot V_{db} + (1-\beta_1) \cdot db, \quad S_{db} = \beta_2 \cdot S_{db} + (1-\beta_2) \cdot (db)^2$$

2)

$$V_{dw}^{correct} = \frac{V_{dw}}{1-\beta_1^t}, \quad S_{dw} = \frac{S_{dw}}{1-\beta_2^t}$$
$$V_{db}^{correct} = \frac{V_{db}}{1-\beta_1^t}, \quad S_{db} = \frac{S_{db}}{1-\beta_2^t}$$

Коррекция смещения

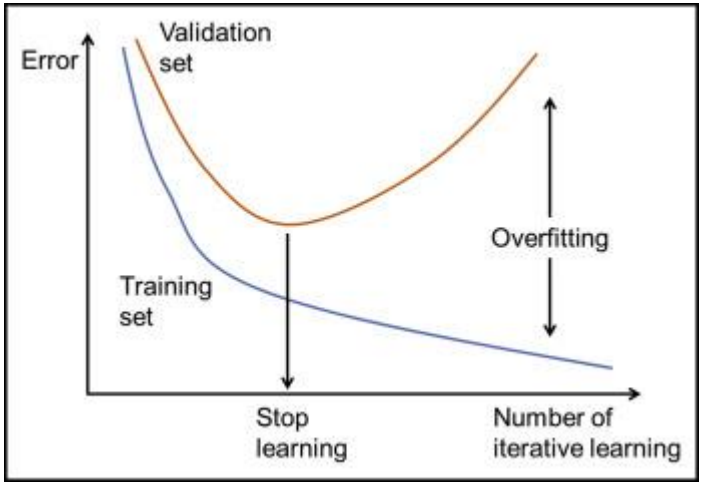
3)

$$w = w - \alpha \frac{V_{dw}^{correct}}{\sqrt{S_{dw}^{correct} + \varepsilon}}$$
$$b = b - \alpha \frac{V_{db}^{correct}}{\sqrt{S_{db}^{correct} + \varepsilon}}$$

Обновление параметров

Overfitting (переобучение)

Переобучение - это чрезмерное подстраивание под данные обучающей выборки, при котором ухудшается качество работы модели.



Регуляризация

Регуляризацией называется контролируемое ухудшение модели, при котором должно предотвращаться чрезмерное подстраивание модели под обучающую выборку.

L2-регуляризация:

$$J(y,\hat{y}) = \sum \mathcal{L}(y^{(i)},\hat{y}^{(i)}) + \frac{\lambda}{2}\sum_{l} ||w^{[l]}||_F^2$$

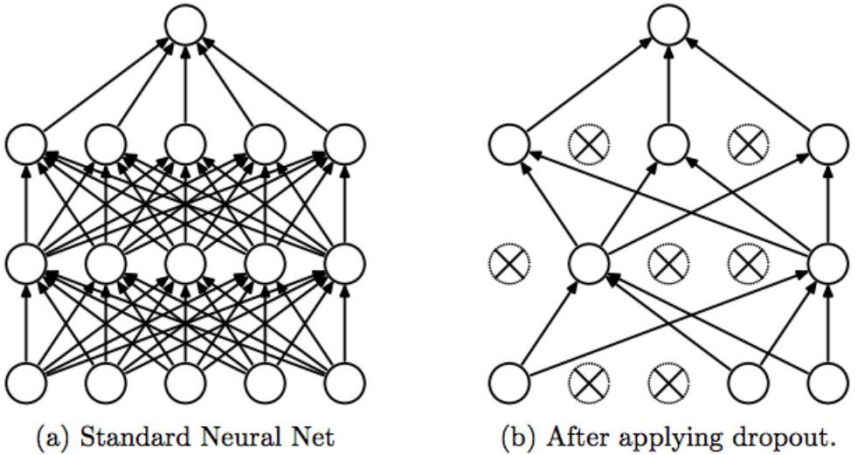
L1-регуляризация:

$$J(y,\hat{y}) = \sum \mathcal{L}(y^{(i)},\hat{y}^{(i)}) + \lambda \sum_{l}\sum_{i,j} ||w_i^{[l]j}||_F$$

Регуляризация. Dropout

Dropout - это техника случайного обнуления заданного количества весов с тем, чтобы увеличить стабильность работы нейросети.

Dropout применятся ТОЛЬКО на стадии обучения модели. Если модель уже обучена, Dropout применять не нужно.



Применение Dropout. Источник: Stanford University

Batch normalization

Batch normalization - это алгоритм, предназначенный для борьбы с затуханием/взрывом градиентов. Также Batch Normalization ускоряет процесс обучения. Идея заключается в нормировке данных на каждом слое, чтобы их среднее и стд. отклонение стали равны 0 и 1 соответственно.

$$\mu^{[l]} = \frac{1}{m}\sum_{i=1}^{m} z^{[l](i)}$$

$$\sigma^{[l]} = \frac{1}{m}\sum_{i=1}^{m}(z^{[l](i)} - \mu^{[l]})$$

$$z^{[l]}_{norm} = \frac{z^{[l]} - \mu^{[l]}}{\sqrt{\sigma^2 + \varepsilon}}$$

$$\hat{z}^{[l]} = \gamma^{[l]} z^{[l]}_{norm} + \beta^{[l]}$$

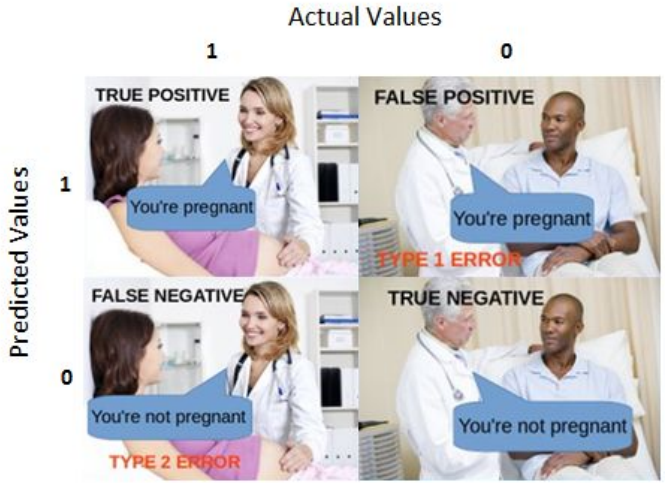
При таком подходе этот слой содержит как обучаемые параметры, так и те, которые считаются с помощью усреднения по всей выборке.

PS. Вспомним про метрики...

Метрики качества, в отличие от функции потерь, никак не используются при обучении. Они рассчитываются исключительно для наглядности и удобства. Однако именно они по большому счету отображают, насколько хорошо алгоритм справляется со сформулированной задачей.

Важнейшие метрики задачи классификации:

- accuracy,
- precision,
- recall.



Метрики...

$$acc = \frac{TP + TN}{TP + TN + FP + FN};$$

$$\mathcal{P} = \frac{TP}{TP + FP};$$

$$\mathcal{R} = \frac{TP}{TP + FN};$$