Введение в нейронные сети

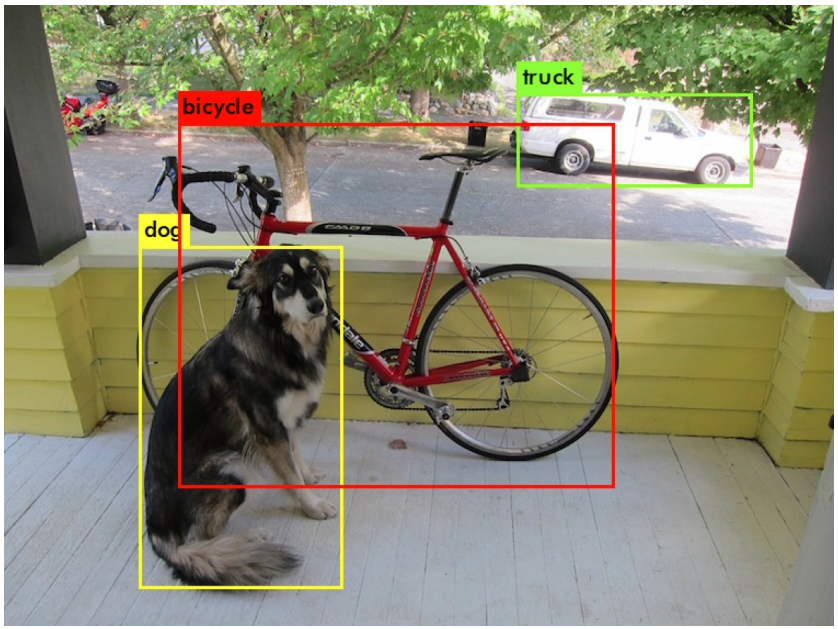
Илья Осокин

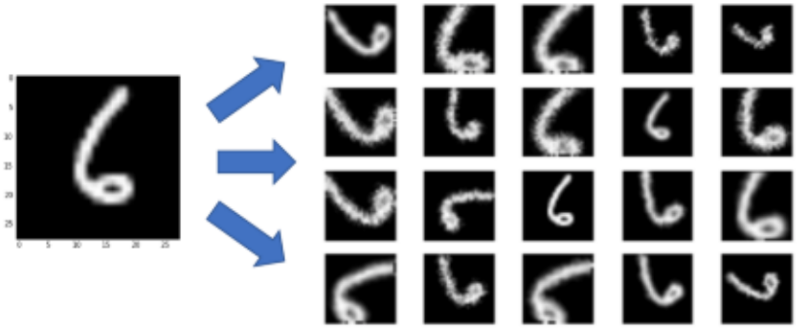
Московский физико-технический институт

23 ноября 2020 г.

План

Набор данных (dataset)





Функция потерь (loss function)

- Принимает на вход ответ модели и правильный ответ
- Зависит от типа данных и задачи
- Минимизируется в процессе обучения

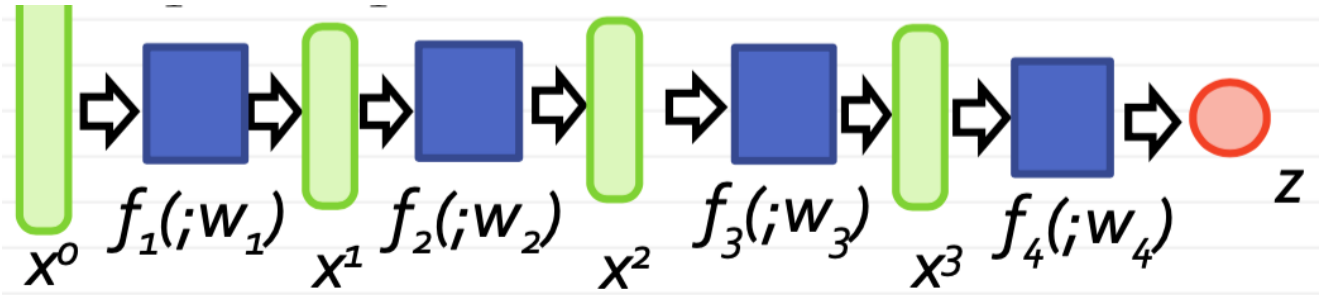
Mean Squared Error: $\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}$

Mean Average Error: $\frac{\sum_{i=1}^{n}\|y_i - \hat{y}_i\|}{n}$

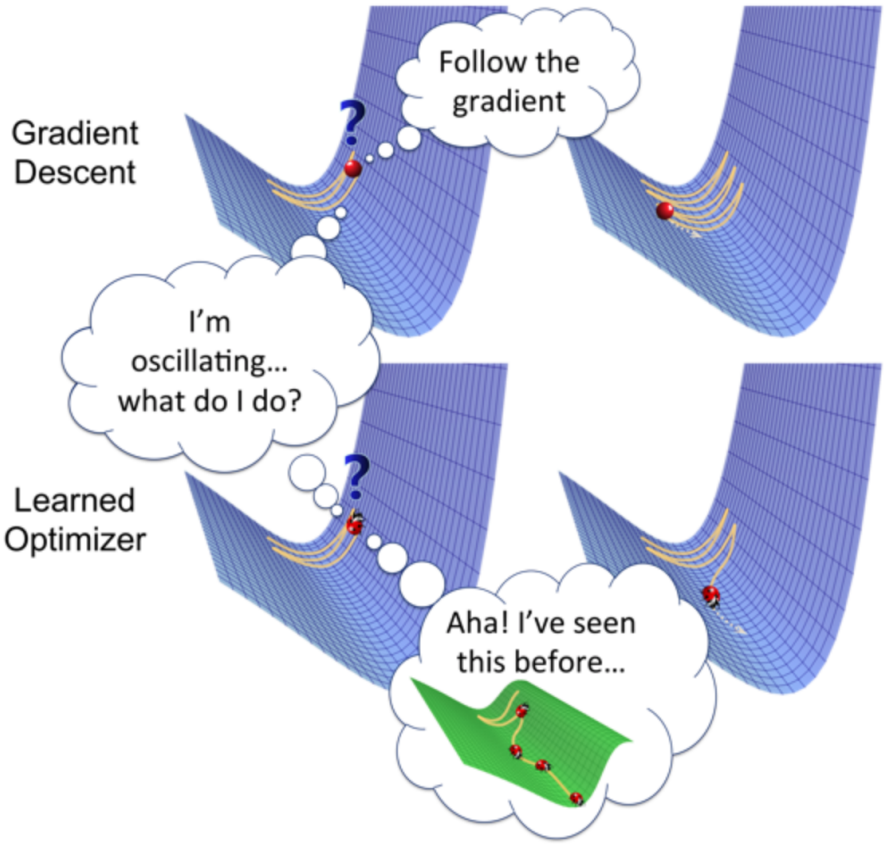
Cross Entropy Loss: $\sum_{i \in I} -(y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i))$

Обратное распространение ошибки (backpropagation)

$$\frac{dz}{dx^0} = \frac{dx^1}{dx^0}^T \frac{dz}{dx^1}$$
$$\frac{dz}{dx^1} = \frac{dx^2}{dx^1}^T \frac{dz}{dx^2}$$



Оптимизатор (optimizer)



https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6

Батч (batch)

- Градиенты для нескольких входов суммируются
- Уменьшает количество модификаций весов
- Делает обучение менее чувствительным к выбросам

Эпоха (epoch)

- Полный проход по датасету
- Данные можно перемешивать

Learning rate

- Ограничение на максимальный шаг
- Баланс между скоростью и стабильностью сходимости

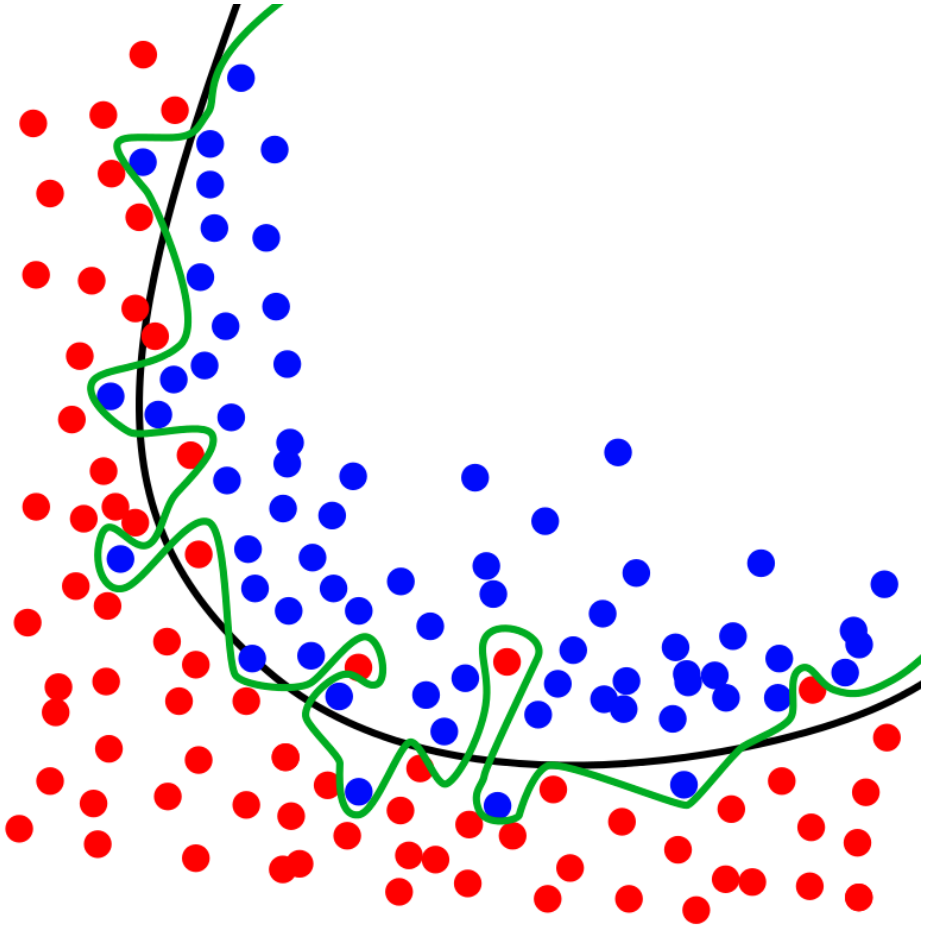
$$\delta w_i = -\alpha \frac{\partial Loss}{\partial w_i}$$

Train loop целиком

- Формирование пары $x, \hat{y}$
- Получение ответа модели от x
- Вычисление лосса
- Нахождение градиентов
- Шаг изменения весов

```
for idx in range(0, EPOCHS_TO_TRAIN):
    for input, target in zip(inputs, targets):
        optimizer.zero_grad()   # zero the gradient buffers
        output = net(input)
        loss = criterion(output, target)
        loss.backward()
        optimizer.step()    # Does the update
```

Переобучение(overfitting)



Train/val/test split