Машинное обучение

Лекция 14. Нейронные сети: детали

Автор: Рустам Азимов

Санкт-Петербургский государственный университет

Санкт-Петербург

Глубокое обучение

- В прошлый раз мы разбирали backpropagation на простом одномерном примере
- На практике хотим работать сразу с батчами
- Делим набор данных на батчи примерно одинакового размера, прогоняем их через нейронную сеть и обновляем веса с помощью backpropagation
- Теперь для классификации изображений на вход не вектор X, а сразу несколько изображений в виде матрицы X размера $batch\_size \times num\_features$
- После использования каждого изображения заканчивается очередная эпоха обучения

Пример линейного слоя

$$X = \begin{pmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \end{pmatrix} \quad W = \begin{pmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \end{pmatrix}$$

$$Y = XW = \begin{pmatrix} x_{1,1}w_{1,1} + x_{1,2}w_{2,1} & x_{1,1}w_{1,2} + x_{1,2}w_{2,2} & x_{1,1}w_{1,3} + x_{1,2}w_{2,3} \\ x_{2,1}w_{1,1} + x_{2,2}w_{2,1} & x_{2,1}w_{1,2} + x_{2,2}w_{2,2} & x_{2,1}w_{1,3} + x_{2,2}w_{2,3} \end{pmatrix}$$

Chain rule

$$\frac{\partial L}{\partial X} = \frac{\partial Y}{\partial X}\frac{\partial L}{\partial Y} \qquad \frac{\partial L}{\partial W} = \frac{\partial Y}{\partial W}\frac{\partial L}{\partial Y}$$

Градиент по входу

$$
\begin{aligned}
\frac{\partial L}{\partial X} &= \begin{pmatrix} \frac{\partial L}{\partial x_{1,1}} & \frac{\partial L}{\partial x_{1,2}} \\ \frac{\partial L}{\partial x_{2,1}} & \frac{\partial L}{\partial x_{2,2}} \end{pmatrix} \\
&= \begin{pmatrix} \frac{\partial L}{\partial y_{1,1}} w_{1,1} + \frac{\partial L}{\partial y_{1,2}} w_{1,2} + \frac{\partial L}{\partial y_{1,3}} w_{1,3} & \frac{\partial L}{\partial y_{1,1}} w_{2,1} + \frac{\partial L}{\partial y_{1,2}} w_{2,2} + \frac{\partial L}{\partial y_{1,3}} w_{2,3} \\ \frac{\partial L}{\partial y_{2,1}} w_{1,1} + \frac{\partial L}{\partial y_{2,2}} w_{1,2} + \frac{\partial L}{\partial y_{2,3}} w_{1,3} & \frac{\partial L}{\partial y_{2,1}} w_{2,1} + \frac{\partial L}{\partial y_{2,2}} w_{2,2} + \frac{\partial L}{\partial y_{2,3}} w_{2,3} \end{pmatrix} \\
&= \begin{pmatrix} \frac{\partial L}{\partial y_{1,1}} & \frac{\partial L}{\partial y_{1,2}} & \frac{\partial L}{\partial y_{1,3}} \\ \frac{\partial L}{\partial y_{2,1}} & \frac{\partial L}{\partial y_{2,2}} & \frac{\partial L}{\partial y_{2,3}} \end{pmatrix} \begin{pmatrix} w_{1,1} & w_{2,1} \\ w_{1,2} & w_{2,2} \\ w_{1,3} & w_{2,3} \end{pmatrix} \\
&= \boxed{\frac{\partial L}{\partial Y} W^T}
\end{aligned}
$$

Градиент по параметрам

$$
\begin{aligned}
\frac{\partial L}{\partial W} &= \begin{pmatrix} \frac{\partial L}{\partial w_{1,1}} & \frac{\partial L}{\partial w_{1,2}} & \frac{\partial L}{\partial w_{1,3}} \\ \frac{\partial L}{\partial w_{2,1}} & \frac{\partial L}{\partial w_{2,2}} & \frac{\partial L}{\partial w_{2,3}} \end{pmatrix} \\
&= \begin{pmatrix} \frac{\partial L}{\partial y_{1,1}} x_{1,1} + \frac{\partial L}{\partial y_{2,1}} x_{2,1} & \frac{\partial L}{\partial y_{1,2}} x_{1,1} + \frac{\partial L}{\partial y_{2,2}} x_{2,1} & \frac{\partial L}{\partial y_{1,3}} x_{1,1} + \frac{\partial L}{\partial y_{2,3}} x_{2,1} \\ \frac{\partial L}{\partial y_{1,1}} x_{1,2} + \frac{\partial L}{\partial y_{2,1}} x_{2,2} & \frac{\partial L}{\partial y_{1,2}} x_{1,2} + \frac{\partial L}{\partial y_{2,2}} x_{2,2} & \frac{\partial L}{\partial y_{1,3}} x_{1,2} + \frac{\partial L}{\partial y_{2,3}} x_{2,2} \end{pmatrix} \\
&= \begin{pmatrix} x_{1,1} & x_{2,1} \\ x_{1,2} & x_{2,2} \end{pmatrix} \begin{pmatrix} \frac{\partial L}{\partial y_{1,1}} & \frac{\partial L}{\partial y_{1,2}} & \frac{\partial L}{\partial y_{1,3}} \\ \frac{\partial L}{\partial y_{2,1}} & \frac{\partial L}{\partial y_{2,2}} & \frac{\partial L}{\partial y_{2,3}} \end{pmatrix} \\
&= \boxed{X^T \frac{\partial L}{\partial Y}}
\end{aligned}
$$

Итоговые формулы для backpropagation

$$\boxed{\frac{\partial L}{\partial X} = \frac{\partial L}{\partial Y} W^T} \qquad \boxed{\frac{\partial L}{\partial W} = X^T \frac{\partial L}{\partial Y}}$$

Интерфейс слоя

```python
class Layer:
    def forward(self, X):
        result = ...  # необходимые вычисления
        self.X = X    # запись значений, которые могут понадобиться для backward
        return result

    def backward(self, d_out):
        d_input = ...        # используем формулу градиента этого слоя и сохраненные значения
        d_w = ...            # также вычисляем градиенты по параметрам слоя
        self.w.grad += d_w  # аккумулируем градиент параметров
        return d_input
```

DL frameworks

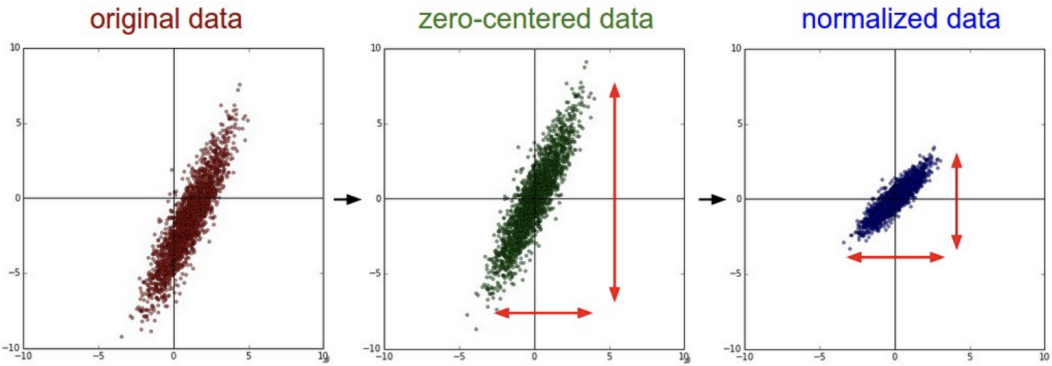
PyTorch

```
import torch.optim as optim
y = torch.LongTensor(np.random.randint(0,3,size=2))

nn = torch.nn.Sequential(
    torch.nn.Linear(3, 10),
    torch.nn.ReLU(),
    torch.nn.Linear(10, 20),
    torch.nn.ReLU(),
    torch.nn.Linear(20, 3)
)

optimizer = optim.SGD(nn.parameters(), lr=0.01, weight_decay=0.05)

for i in range(100):
    optimizer.zero_grad()
    pred = nn(X)
    criterion = torch.nn.CrossEntropyLoss()
    loss = criterion(pred, y)
    loss.backward()
    optimizer.step()
```

Основные этапы работы с сетью

- Подготовка данных
- Выбор активационной функции и архитектуры
- Инициализация весов
- Выбор оптимизатора (метод обновления весов)
- Подбор гиперпараметров (**lr**, lr decay, regularization)

Подготовка данных



Batch normalization

- Вычитаем из батча среднее по батчу и делим на отклонение
- Как нормализация данных, только после каждого полносвязного слоя (перед нелинейными)
- Такое преобразование дифференциуремо
- Позволяет избежать различной головной боли и становится не так важна инициализация весов
- Ускоряет и стабилизирует тренировку сети
- Регуляризует
- При предсказании на тестовой выборке используем среднее накопленное значение батчей

Инициализация весов для Relu (He initialization)

```
a = 1 / sqrt(in_num / 2)
W = a * random(in_num, out_num)
```

Оптимизатор (обновление параметров)

```
# Vanilla update
x += - learning_rate * dx
```

Momentum

```
# Momentum update
v = mu * v - learning_rate * dx # integrate velocity
x += v # integrate position
```

Adagrad

```
# Assume the gradient dx and parameter vector x
cache += dx**2
x += - learning_rate * dx / (np.sqrt(cache) + eps)
```

RMSprop

```
cache = decay_rate * cache + (1 - decay_rate) * dx**2
x += - learning_rate * dx / (np.sqrt(cache) + eps)
```

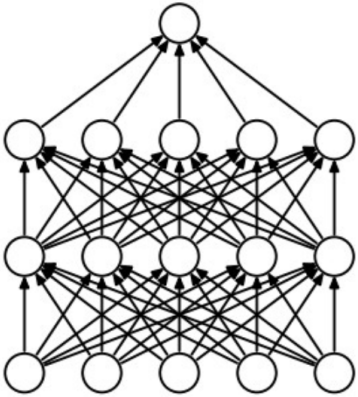
Adam (RMSprop + Momentum)

```
m = beta1*m + (1-beta1)*dx
v = beta2*v + (1-beta2)*(dx**2)
x += - learning_rate * m / (np.sqrt(v) + eps)
```

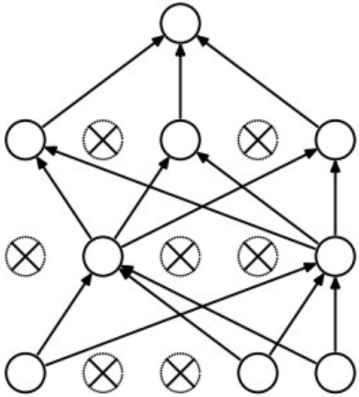
Регуляризация

- $L1$
- $L2$
- Dropout

Dropout

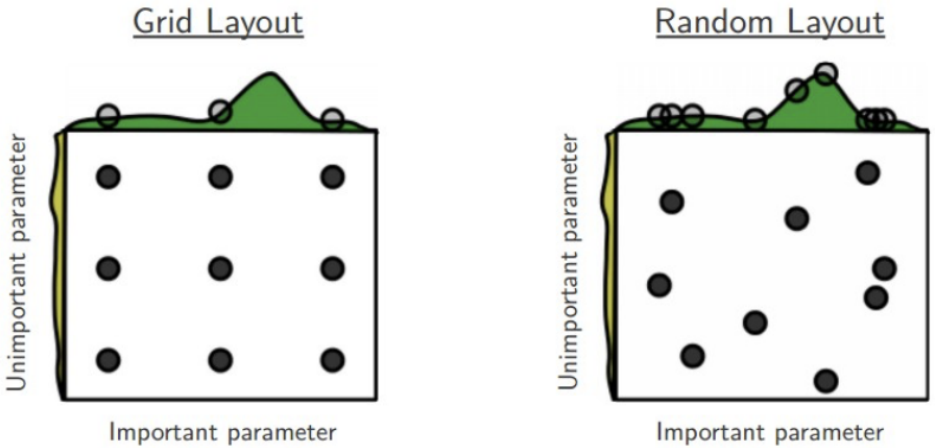


(a) Standard Neural Net

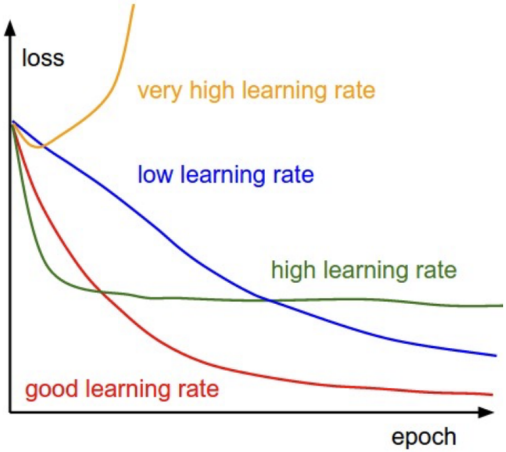


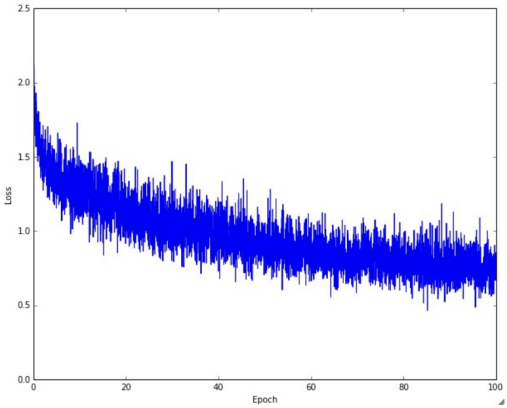
(b) After applying dropout.

Подбор параметров

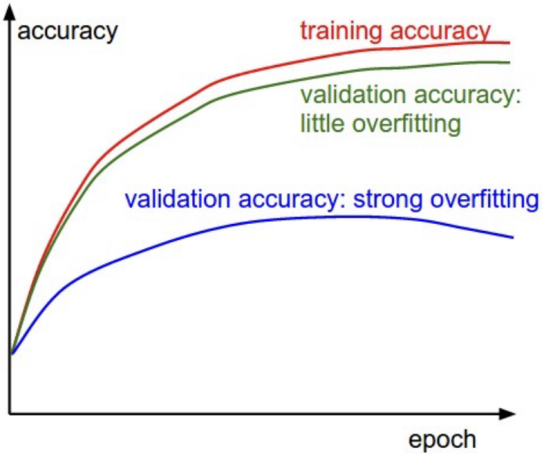


Важность подбора learning rate





Оцениваем модель на переобучение



С чего начинать?

- Вычитать среднее из данных
- Функция активации — Relu
- Оптимизатор — Adam
- Использовать Batch Normalization
- Learning rate decay на плато
- Регуляризатор — $L2$
- Подбирать гиперпараметры, особенно learning rate
- Смотреть на графики train/val

Дополнительные источники

- https://cs231n.github.io
- https://dlcourse.ai