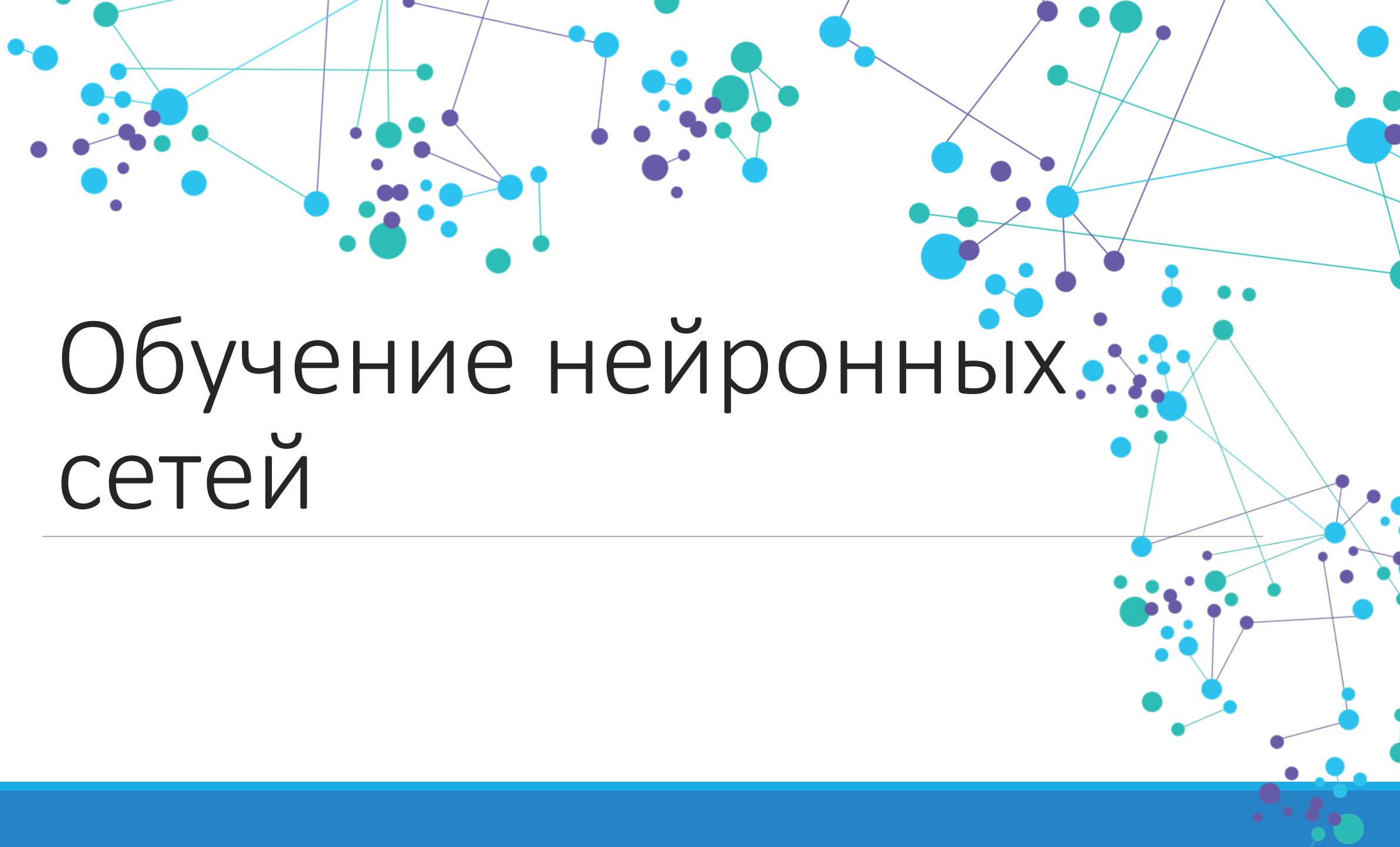


Обучение нейронных сетей

- Как обучаются нейронные сети
- Разбиение выборки для обучения
- Мониторинг процесса обучения
- Задание 4

Как обучаются нейронные сети

Функция потерь

Функция потерь — функция, которая в теории статистических решений характеризует потери при неправильном принятии решений на основе наблюдаемых данных.

Softmax

Cat	**3.2**
Car	5.1
Frog	-1.7

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i + R(W)$$

s – Score Function (функция оценки)

$s = f(x, W)$ – сверточная нейронная сеть с параметрами W

$R(W)$ – регуляризатор

N – количество изображений в выборке

Чем больше сеть ошибается, тем большее значение имеет функция потерь

Softmax

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

Cat	**3.2**			**24.5**			**0.13**→	$\mathbf{-log(0.13) = 0.89}$
Car	5.1	→	экспонента →	164.0	→	нормализация	→	0.87
Frog	-1.7			0.18				0.00

Softmax

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

Cat	3.2			24.5			0.13	
Car	**5.1**	**→**	**экспонента**	**→**	**164.0**	**→**	**нормализация**	**→**	**0.87**	**→**	$-\log(0.87) = 0.14$
Frog	-1.7			0.18			0.00	

Softmax

$$L_i = -\log(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}})$$

Cat	3.2			24.5			0.13
Car	5.1	→	экспонента →	164.0	→	нормализация →	0.87
Frog	-1.7			0.18			0.00→ $-\log(0) = \infty$

Обучение сети – минимизация функции потерь

$$L_i = -\log\left(\frac{e^{\boldsymbol{s_{y_i}}}}{\sum_j e^{\boldsymbol{s_j}}}\right)$$

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i + R(W)$$

s – Score Function (функция оценки)

$\boldsymbol{s = f(x, W)}$ – сверточная нейронная сеть с параметрами $\boldsymbol{W}$

Необходимо подобрать параметры сети (фильтры сверточных слоев) так, чтобы функция потерь принимала наименьшее значение

$$\underset{W}{\operatorname{argmin}}\, L$$

Метод градиентного спуска

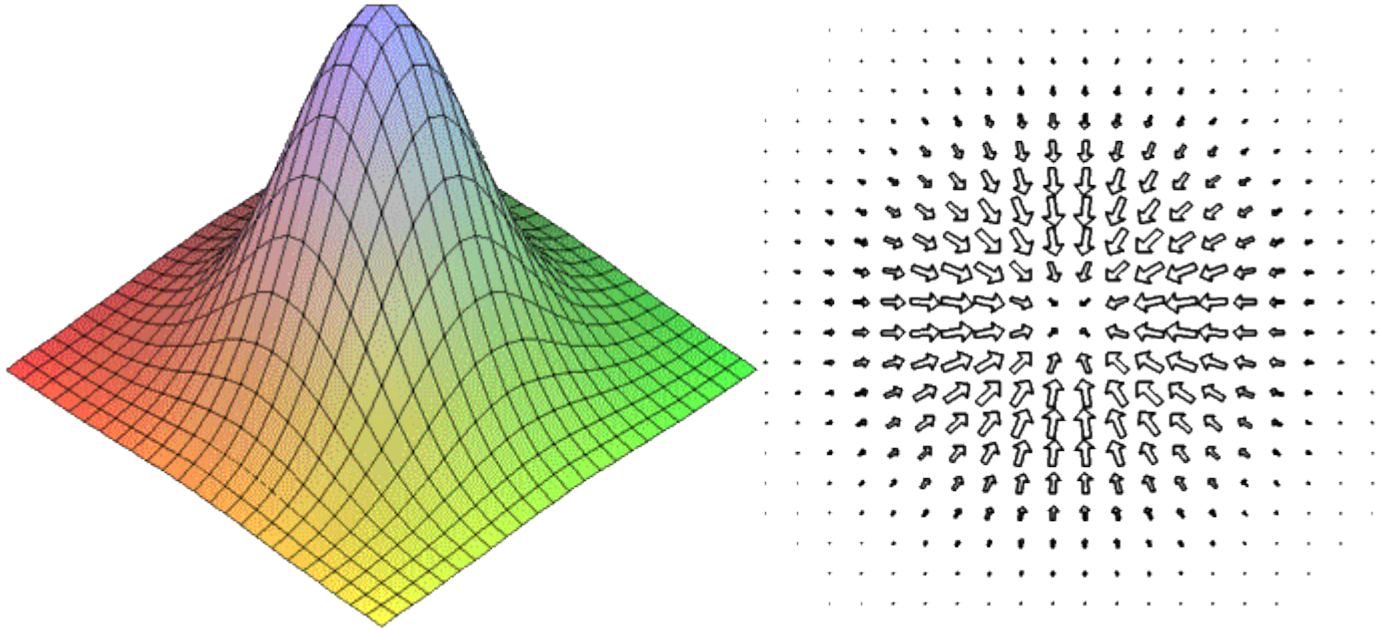
Градиент

$f = f(x, y, z)$

$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right)$ **- градиент**

Градиент указывает направление возрастания функции

В случае минимизации необходимо брать отрицательный градиент



Нас интересует $-\nabla_W L$

https://www.youtube.com/watch?v=kJgx2RcJKZY

Метод обратного распространения ошибки

$s = f(x, W)$ **– нейронная сеть**

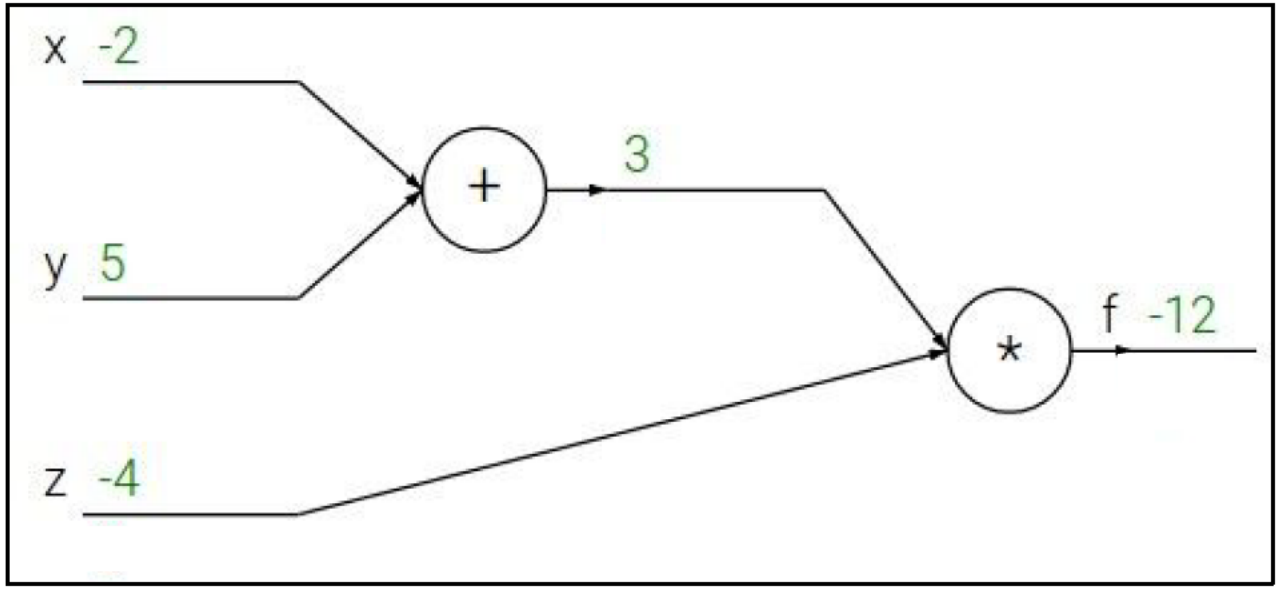
Необходимо найти градиент $\rightarrow$ **частные производные для каждого элемента из** W

Нейронная сеть – функция, которую можно представить в виде графа и посчитать локальную производную для каждого узла

Backpropagation: a simple example

$$f(x, y, z) = (x + y)z$$

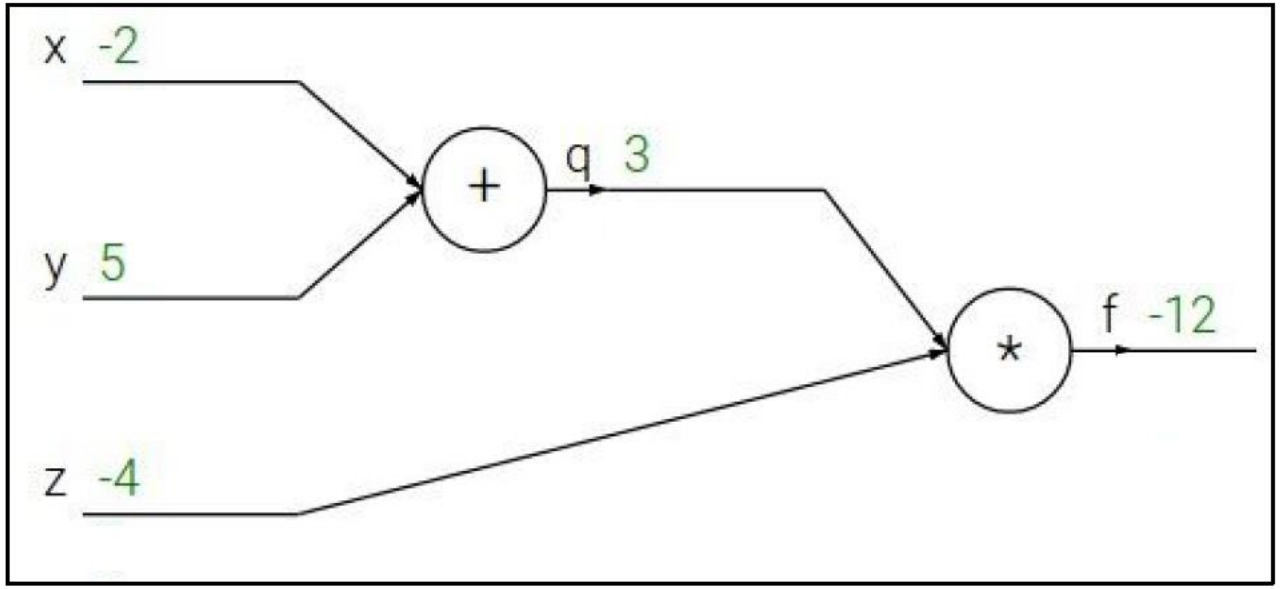
e.g. x = -2, y = 5, z = -4



Backpropagation: a simple example

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Backpropagation: a simple example

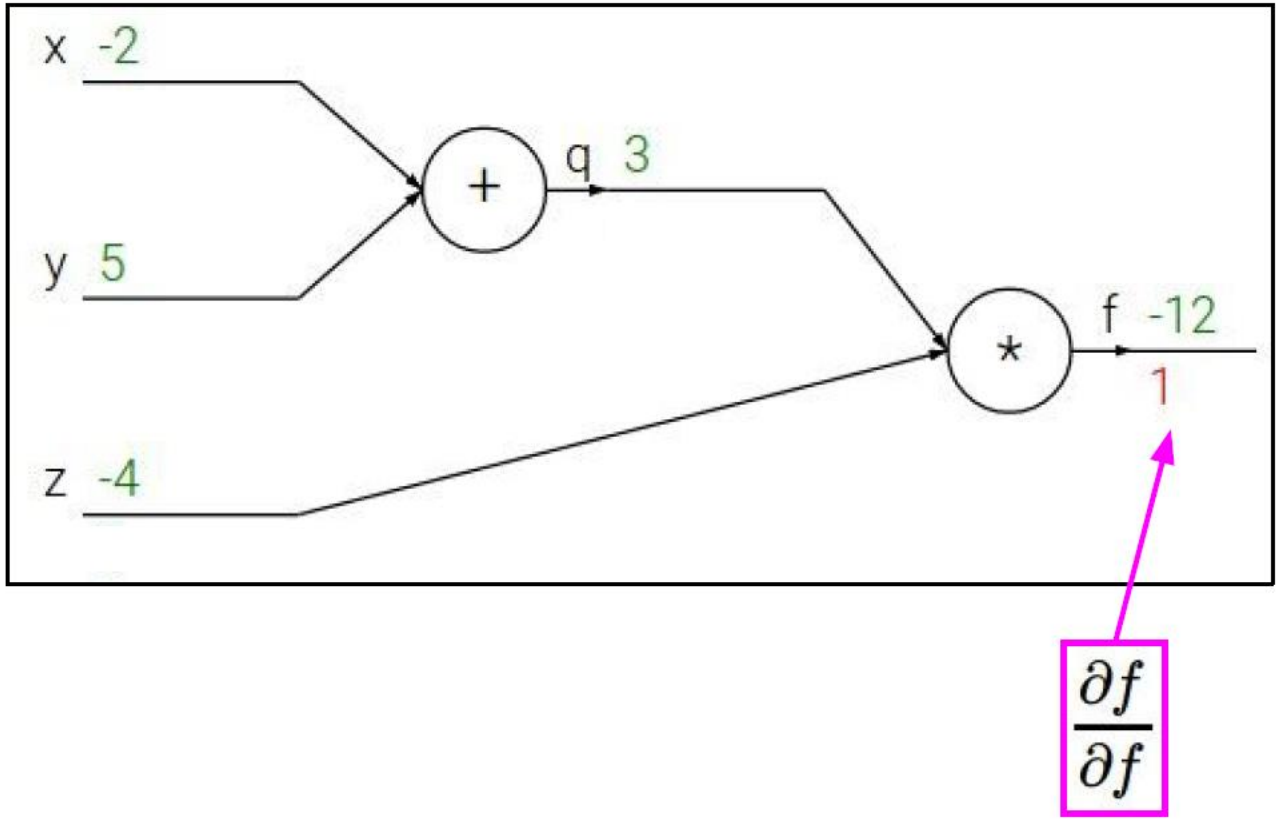
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Backpropagation: a simple example

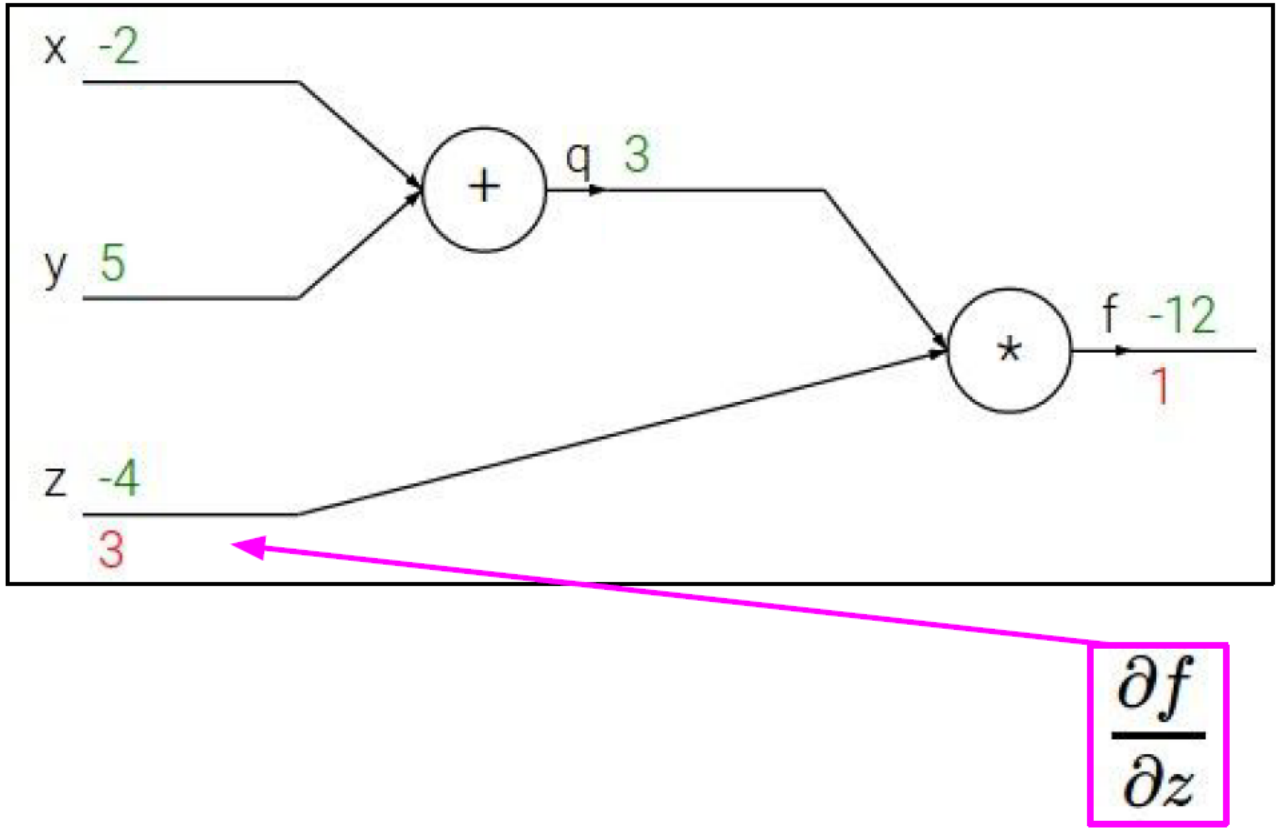
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Backpropagation: a simple example

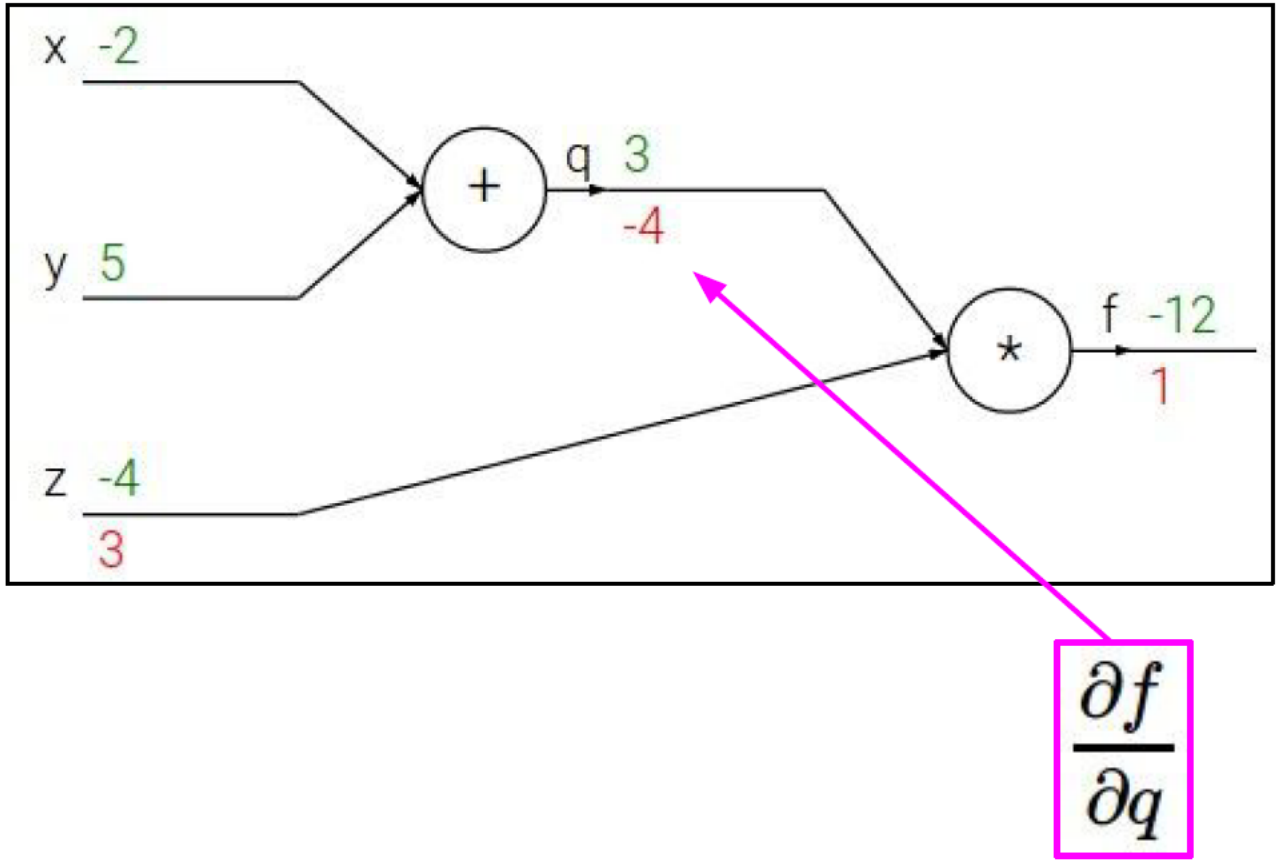
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Backpropagation: a simple example

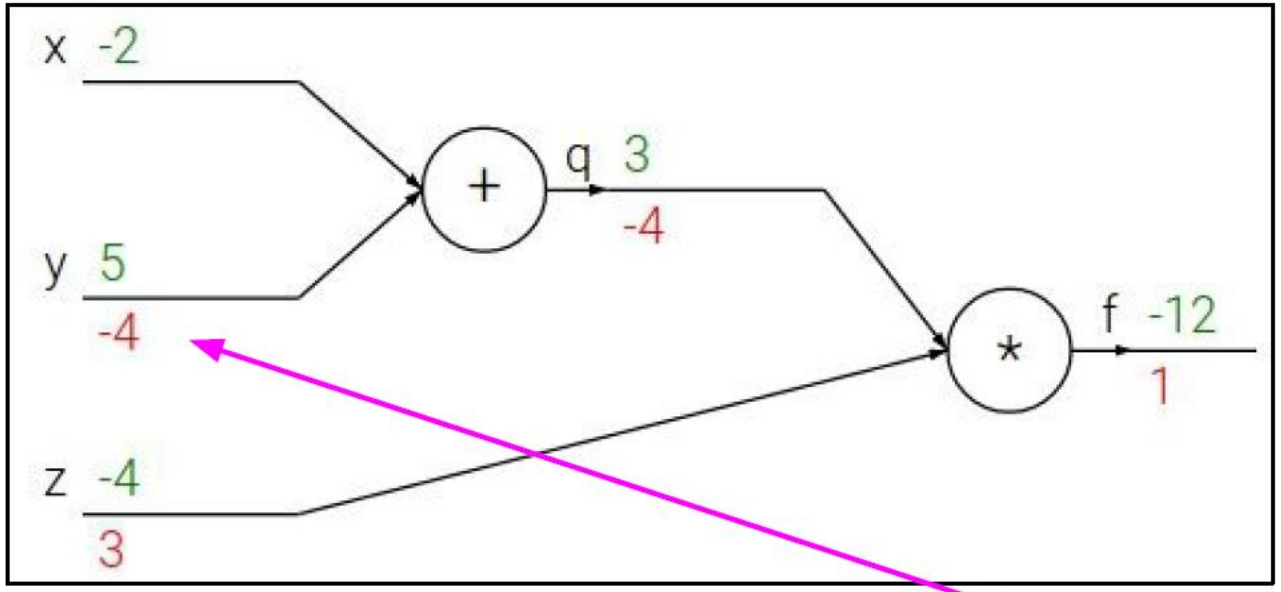
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Chain rule:

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y}$$

Backpropagation: a simple example

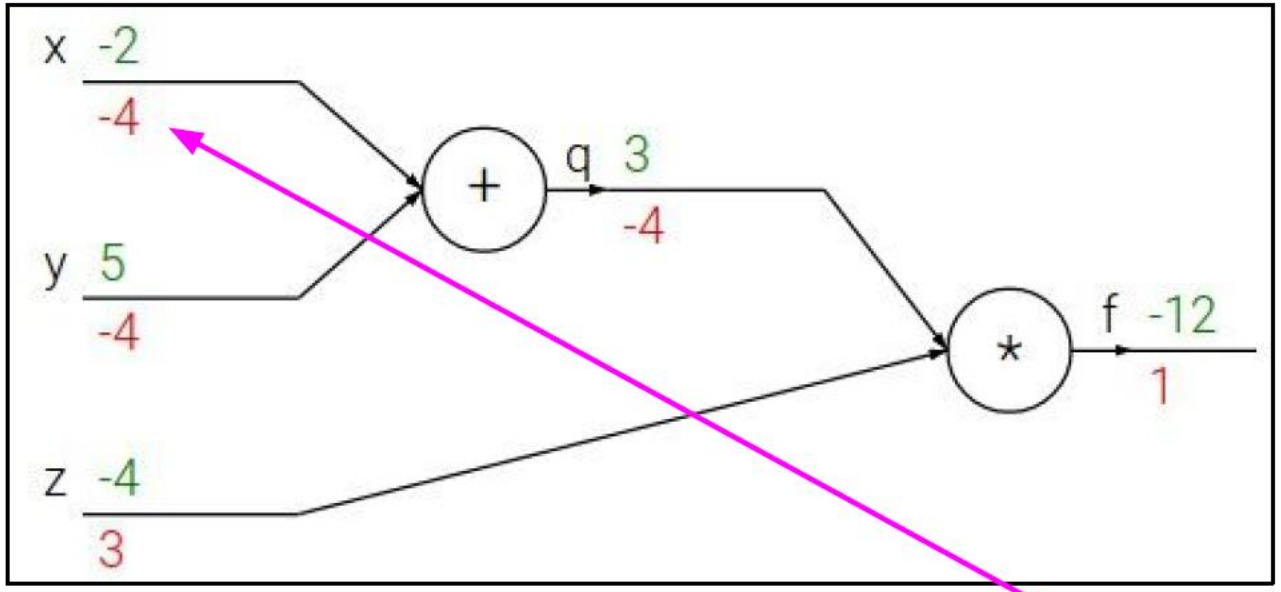
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$\frac{\partial f}{\partial x}$

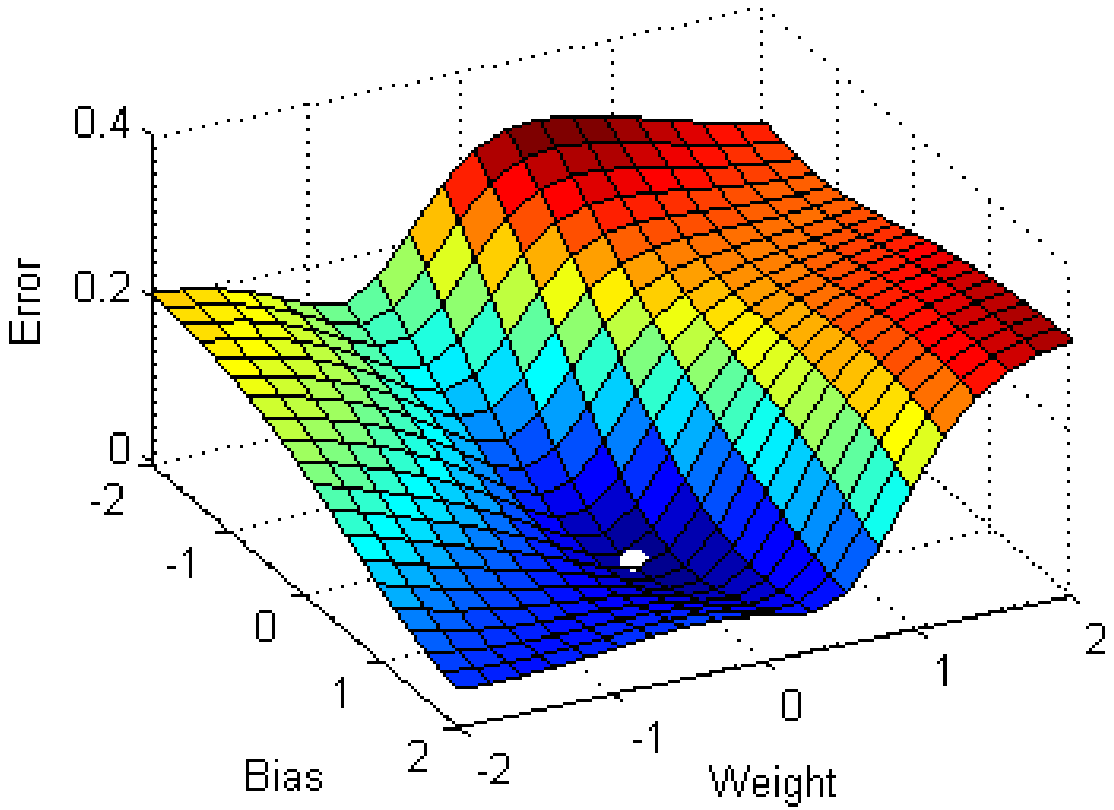
Chain rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

Обновление весов

$$w_j^{i+1} = w_j^i + v\left(-\frac{\partial l}{\partial w_i}\right)$$

v – скорость обучения

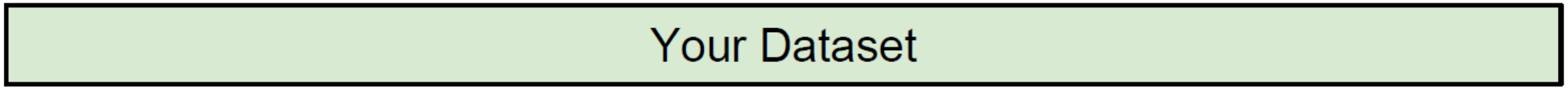


Разбиение выборки

«Запоминание» ответа

Обученная нейронная сеть всегда дает верный ответ для тех элементов выборки, на которых она обучалась

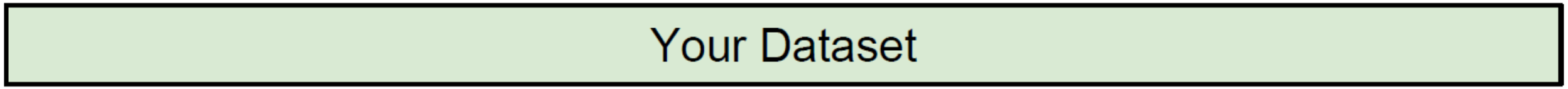
Обучающая-тестовая



Выборка разделяется на обучающую и тестовую
На тестовой оценивают точность сети

Используется в крайнем случае, так как не дает объективной информации о точности сети

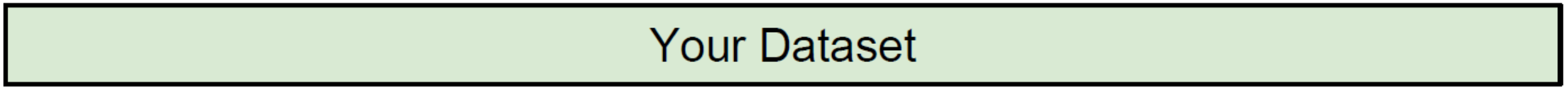
Обучающая-проверочная-тестовая



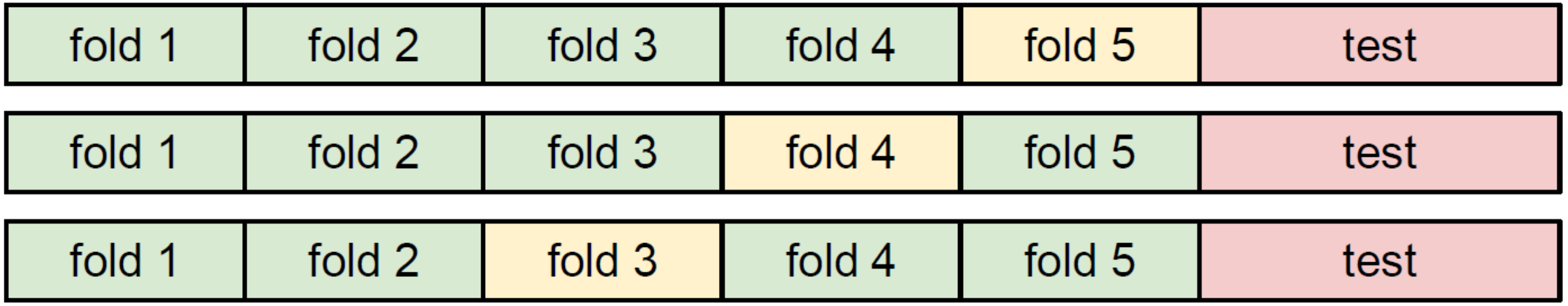
Выборка разделяется на обучающую, проверочную и тестовую
Проверочная используется для оценки точности на этапе обучения
Тестовая используется для оценки точности после обучения

Как правило используется в компьютерном зрении, если данных достаточно

Кросс-проверка



Выборка разделяется на несколько частей
Оценка точности проводится в несколько этапов, на каждом этапе проверочная выборка новая
Цель – получить наибольшую среднюю точность



В компьютерном зрении используется редко, только если выборка небольшая

Мониторинг процесса обучения

HTTPS://CS231N.GITHUB.IO/NEURAL-NETWORKS-3/#SANITYCHECK

Предварительно: корректность функции потерь

```
def init_two_layer_model(input_size, hidden_size, output_size):
  # initialize a model
  model = {}
  model['W1'] = 0.0001 * np.random.randn(input_size, hidden_size)
  model['b1'] = np.zeros(hidden_size)
  model['W2'] = 0.0001 * np.random.randn(hidden_size, output_size)
  model['b2'] = np.zeros(output_size)
  return model
```

```
model = init_two_layer_model(32*32*3, 50, 10) # input size, hidden size, number of classes
loss, grad = two_layer_net(X_train, model, y_train, 0.0)
print loss
```

disable regularization

```
2.30261216167
```

loss ~2.3.
“correct “ for
10 classes

returns the loss and the
gradient for all parameters

Предварительно: корректность функции потерь

```
def init_two_layer_model(input_size, hidden_size, output_size):
  # initialize a model
  model = {}
  model['W1'] = 0.0001 * np.random.randn(input_size, hidden_size)
  model['b1'] = np.zeros(hidden_size)
  model['W2'] = 0.0001 * np.random.randn(hidden_size, output_size)
  model['b2'] = np.zeros(output_size)
  return model
```

```
model = init_two_layer_model(32*32*3, 50, 10) # input size, hidden size, number of classes
loss, grad = two_layer_net(X_train, model, y_train, 1e3)
print loss
```

crank up regularization

```
3.06859716482
```

loss went up, good. (sanity check)

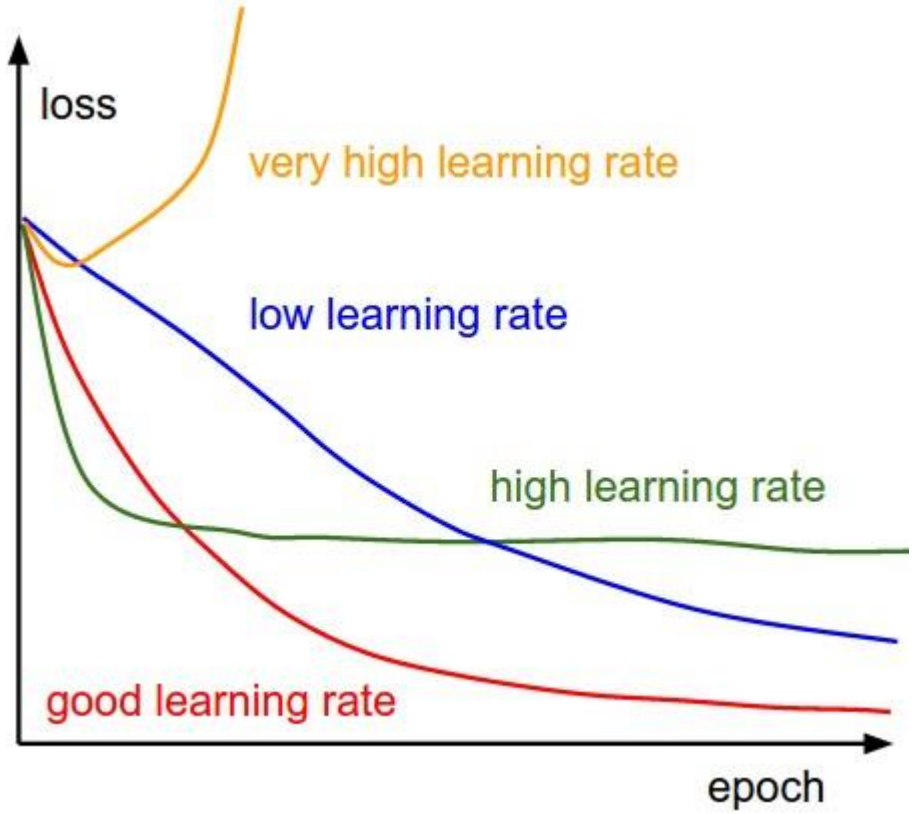
Предварительно: переобучение на крошечной выборке

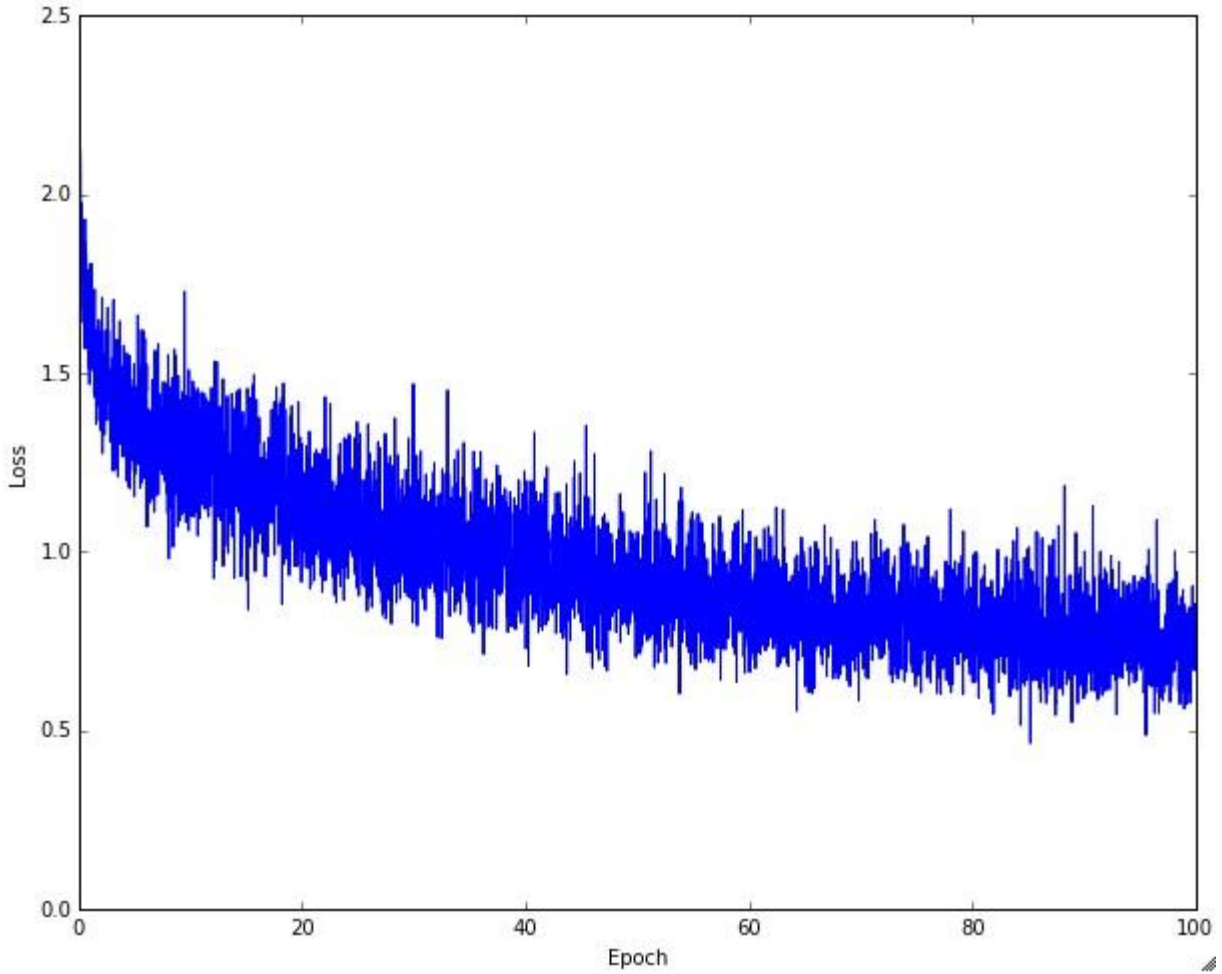
```
model = init_two_layer_model(32*32*3, 50, 10) # input size, hidden size, number of classes
trainer = ClassifierTrainer()
X_tiny = X_train[:20] # take 20 examples
y_tiny = y_train[:20]
best_model, stats = trainer.train(X_tiny, y_tiny, X_tiny, y_tiny,
                                  model, two_layer_net,
                                  num_epochs=200, reg=0.0,
                                  update='sgd', learning_rate_decay=1,
                                  sample_batches = False,
                                  learning_rate=1e-3, verbose=True)
```

```
Finished epoch 1 / 200: cost 2.302603, train: 0.400000, val 0.400000, lr 1.000000e-03
Finished epoch 2 / 200: cost 2.302258, train: 0.450000, val 0.450000, lr 1.000000e-03
Finished epoch 3 / 200: cost 2.301849, train: 0.600000, val 0.600000, lr 1.000000e-03
Finished epoch 4 / 200: cost 2.301196, train: 0.650000, val 0.650000, lr 1.000000e-03
Finished epoch 5 / 200: cost 2.300044, train: 0.650000, val 0.650000, lr 1.000000e-03
Finished epoch 6 / 200: cost 2.297864, train: 0.550000, val 0.550000, lr 1.000000e-03
Finished epoch 7 / 200: cost 2.293595, train: 0.600000, val 0.600000, lr 1.000000e-03
Finished epoch 8 / 200: cost 2.285096, train: 0.550000, val 0.550000, lr 1.000000e-03
Finished epoch 9 / 200: cost 2.268094, train: 0.550000, val 0.550000, lr 1.000000e-03
Finished epoch 10 / 200: cost 2.234787, train: 0.500000, val 0.500000, lr 1.000000e-03
Finished epoch 11 / 200: cost 2.173187, train: 0.500000, val 0.500000, lr 1.000000e-03
Finished epoch 12 / 200: cost 2.076862, train: 0.500000, val 0.500000, lr 1.000000e-03
Finished epoch 13 / 200: cost 1.974090, train: 0.400000, val 0.400000, lr 1.000000e-03
Finished epoch 14 / 200: cost 1.895885, train: 0.400000, val 0.400000, lr 1.000000e-03
Finished epoch 15 / 200: cost 1.820876, train: 0.450000, val 0.450000, lr 1.000000e-03
Finished epoch 16 / 200: cost 1.737430, train: 0.450000, val 0.450000, lr 1.000000e-03
Finished epoch 17 / 200: cost 1.642356, train: 0.500000, val 0.500000, lr 1.000000e-03
Finished epoch 18 / 200: cost 1.535239, train: 0.600000, val 0.600000, lr 1.000000e-03
Finished epoch 19 / 200: cost 1.421527, train: 0.600000, val 0.600000, lr 1.000000e-03
```

```
Finished epoch 195 / 200: cost 0.002694, train: 1.000000, val 1.000000, lr 1.000000e-03
Finished epoch 196 / 200: cost 0.002674, train: 1.000000, val 1.000000, lr 1.000000e-03
Finished epoch 197 / 200: cost 0.002655, train: 1.000000, val 1.000000, lr 1.000000e-03
Finished epoch 198 / 200: cost 0.002635, train: 1.000000, val 1.000000, lr 1.000000e-03
Finished epoch 199 / 200: cost 0.002617, train: 1.000000, val 1.000000, lr 1.000000e-03
Finished epoch 200 / 200: cost 0.002597, train: 1.000000, val 1.000000, lr 1.000000e-03
finished optimization. best validation accuracy: 1.000000
```

Мониторинг: скорость обучения

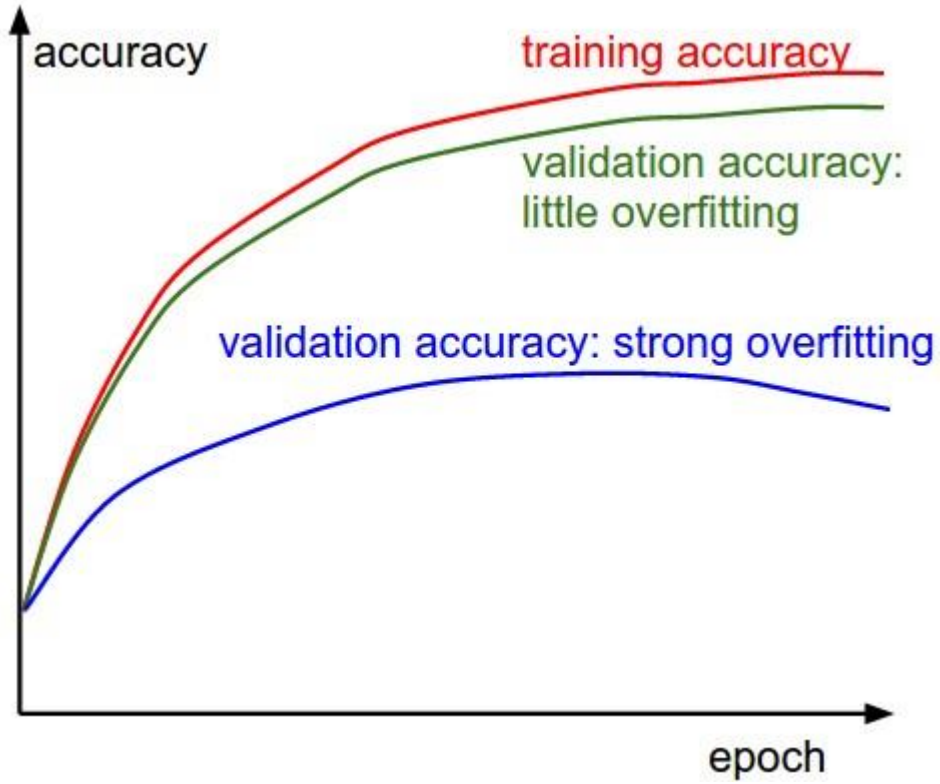




$$w_j^{i+1} = w_j^i + v\left(-\frac{\partial l}{\partial w_i}\right)$$

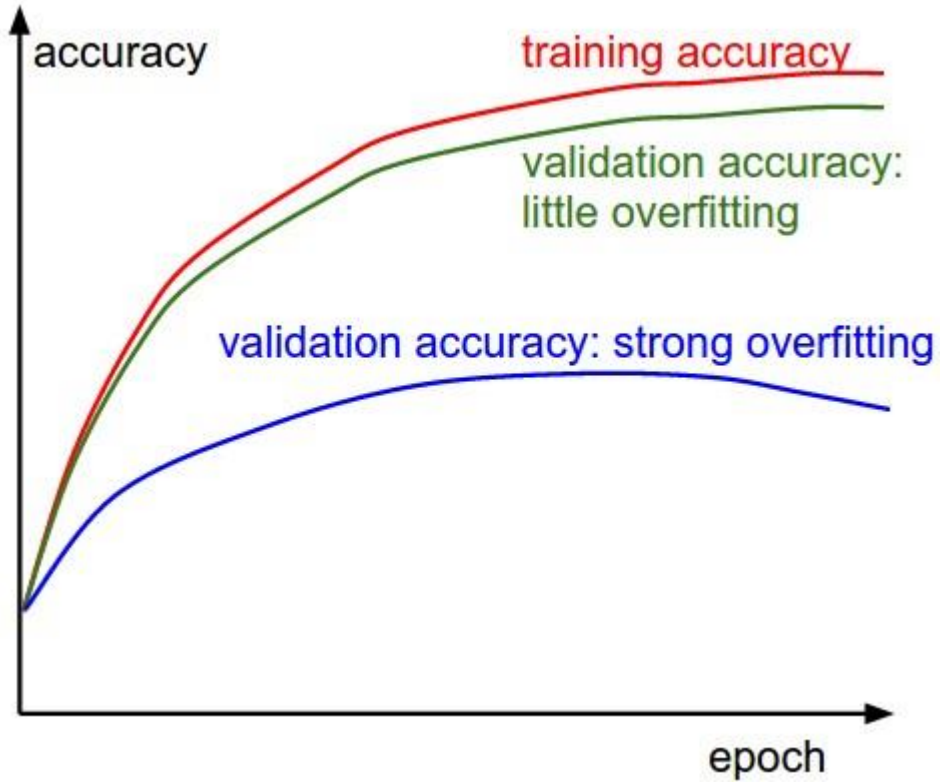
v – скорость обучения

Мониторинг: регуляризация



Разница между точностью во время обучения и точностью во время теста должна быть минимальна

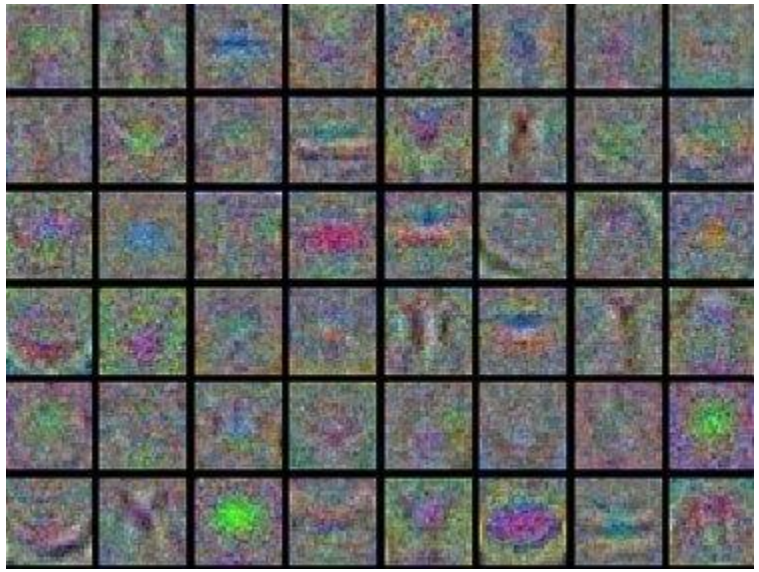
Мониторинг: регуляризация



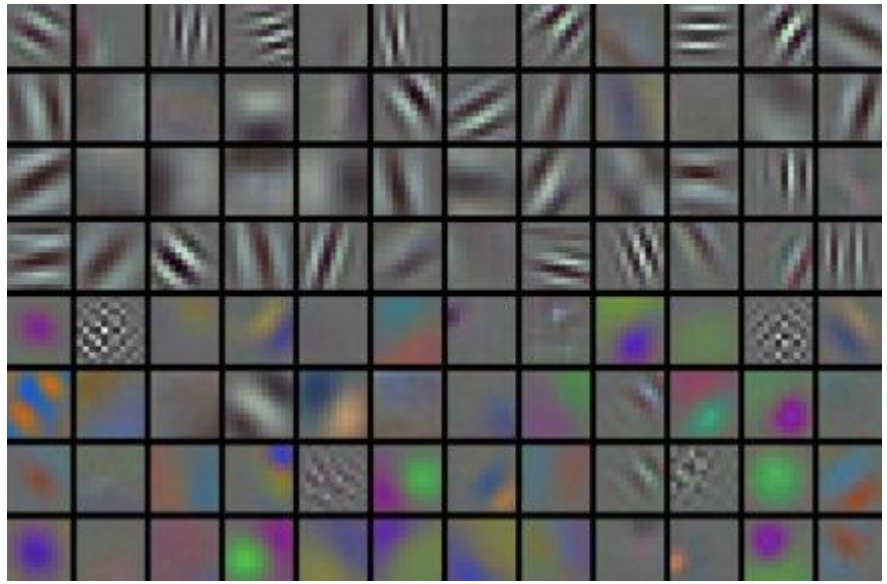
Разница между точностью во время обучения и точностью во время теста должна быть минимальна

Начинайте с небольших значений параметра регуляризации

Мониторинг: визуализация первых слоев



Низкая скорость обучения – зашумленные фильтры



Фильтры здоровой сети