

# Машинное обучение на примере глубокого обучения в компьютерного зрения

## Занятие 2 Функция ошибки и Оптимизация.

Дмитрий Яшунин, к.ф.-м.н  
IntelliVision

e-mail: [yashuninda@yandex.ru](mailto:yashuninda@yandex.ru)

# На прошлом занятии: Сложности распознавания

Освещение



Деформации



Заслонение



Фон

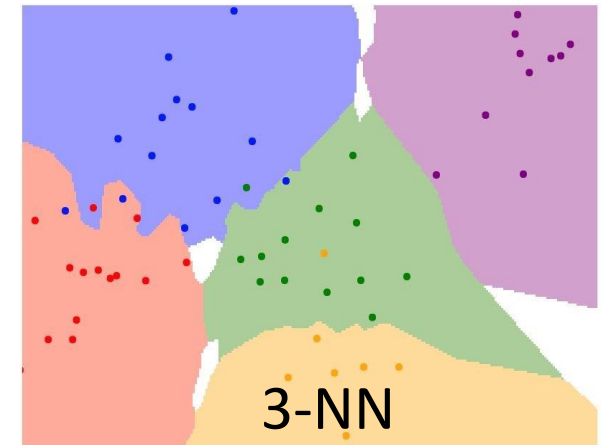
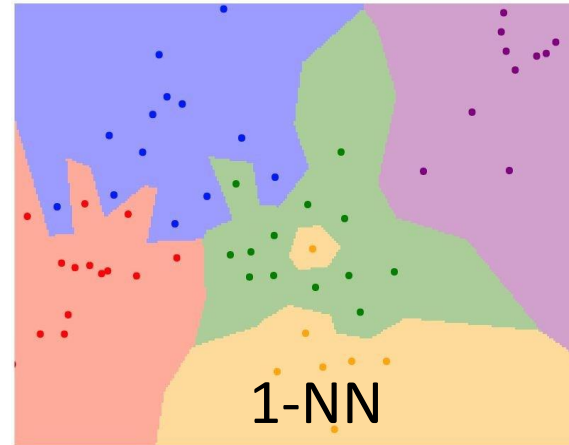
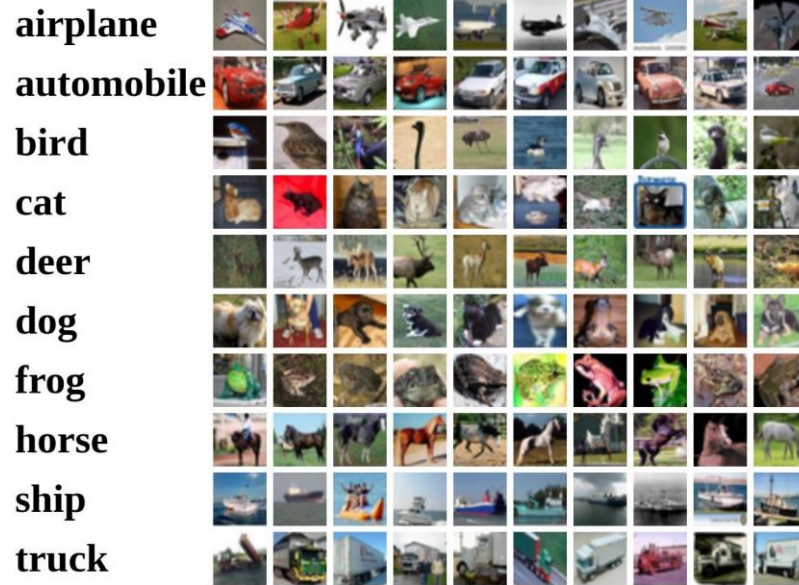


Вариативность классов

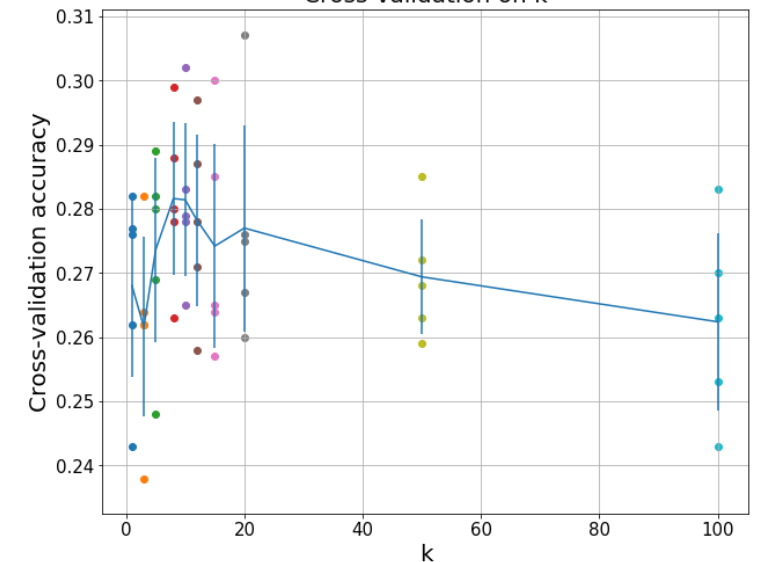


# На прошлом занятии: Классификация на основе данных, kNN

CIFAR10



Cross-validation on k



Cross-validation

fold 1	fold 2	fold 3	fold 4	test
fold 1	fold 2	fold 3	fold 4	test

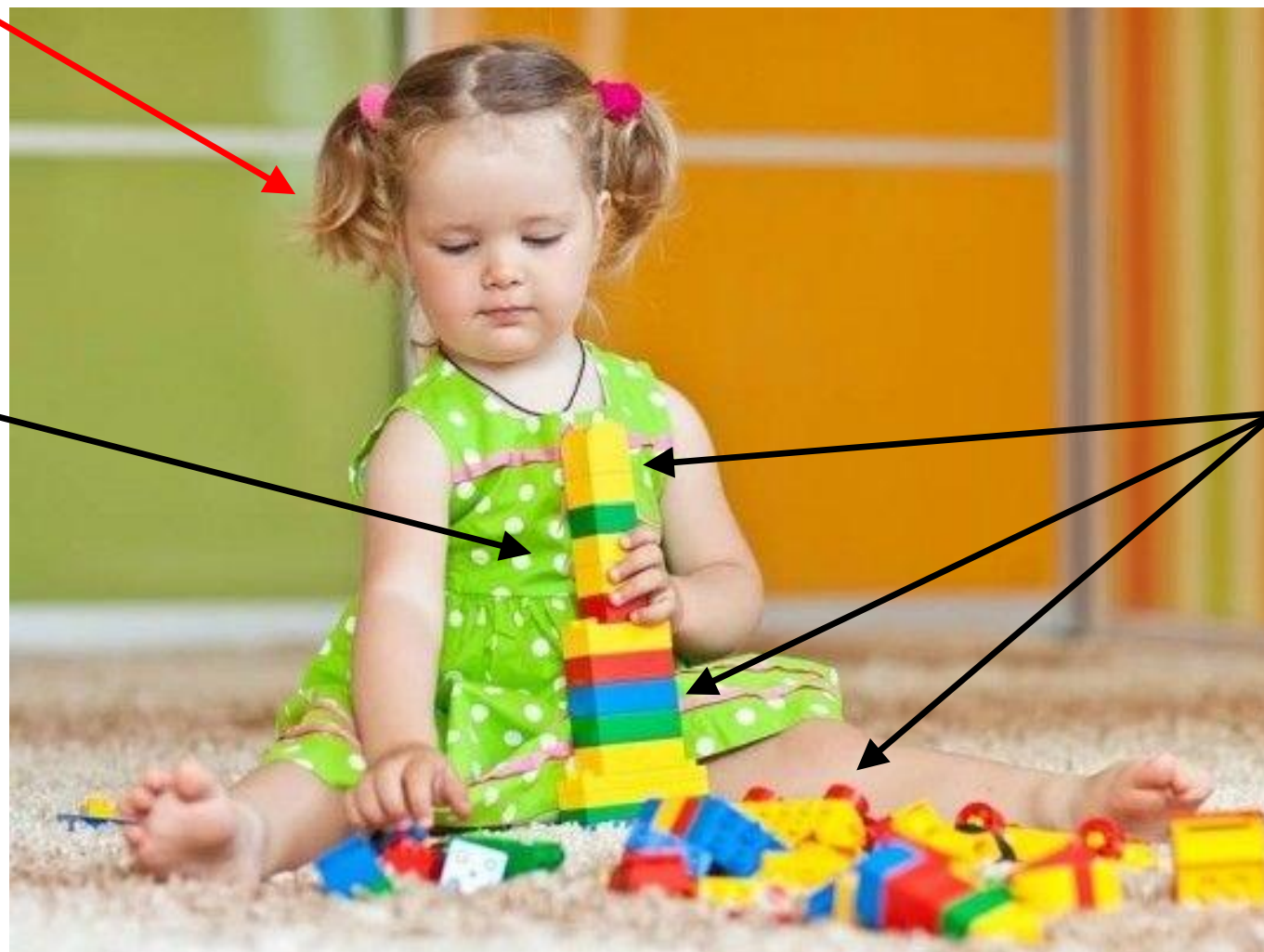


# В прошлый раз: Линейный классификатор

Neural Network practitioner

Neural Network

Linear  
classifiers



# В прошлый раз: Линейный классификатор

Image



$$s = f(x, W) = Wx + b$$

Diagram illustrating the linear classification equation  $s = f(x, W) = Wx + b$  with dimensions:

- $s$  (scores):  $10 \times 1$  (green box)
- $f(x, W)$  (function):  $10 \times 1$  (green box)
- $W$  (weights or parameters):  $10 \times 3072$  (red box)
- $x$  (image pixels):  $3072 \times 1$  (blue box)
- $b$  (bias):  $10 \times 1$  (purple box)

$s$  – scores

$W$  – weights or parameters

$x$  – image pixels

$b$  – bias

Array of **32x32x3** numbers  
(3072 numbers total)

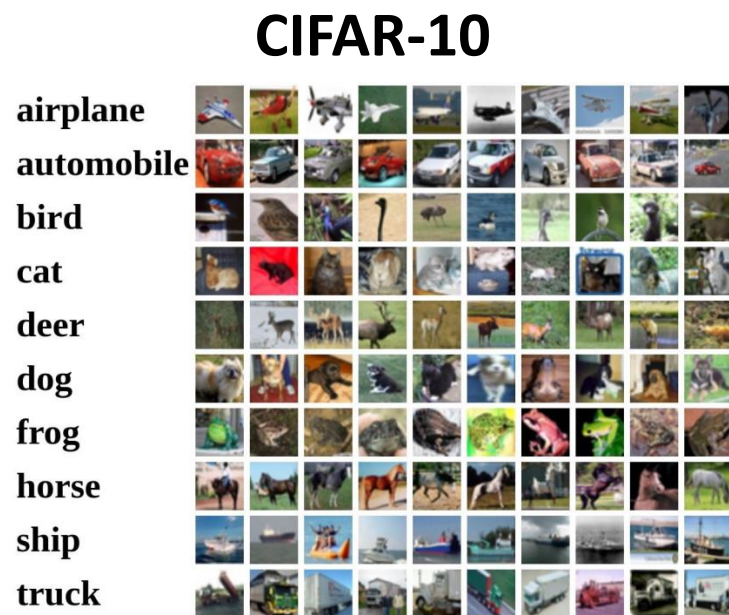
**CIFAR-10**

**50,000** training images

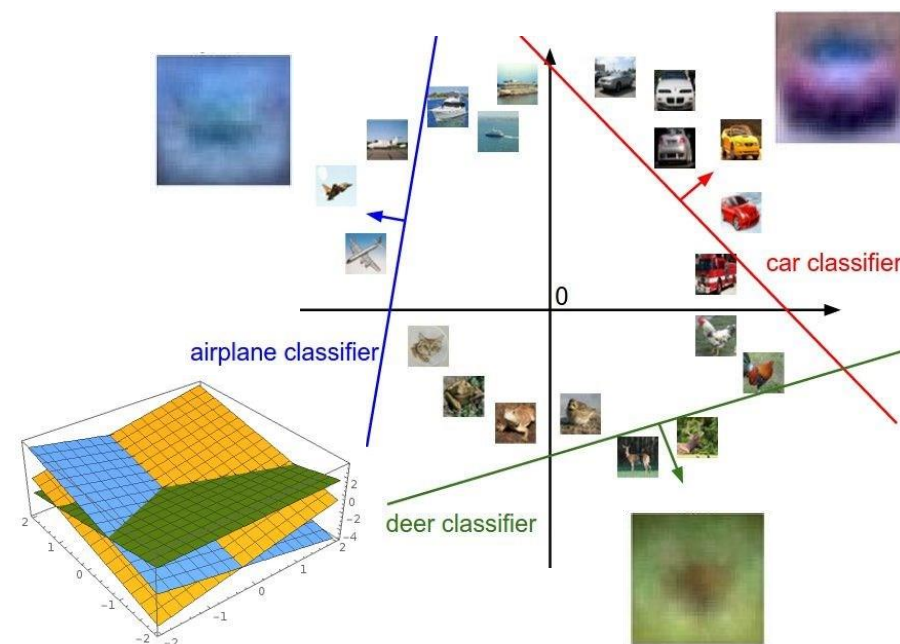
**10,000** testing images

**10** classes

# В прошлый раз: Интерпретация линейного классификатора



$$f(x, W) = Wx + b$$



Example trained weights of a linear classifier trained on CIFAR-10:



# В прошлый раз: Линейный классификатор



$$f(x, W) = Wx + b$$

1. Как ввести функцию ошибки, которая покажет насколько хорошо работает линейный классификатор?

2. Как найти параметры  $W$ , чтобы минимизировать значение функции потерь?

airplane	-3.45	-0.51	3.42
automobile	-8.87	<b>6.04</b>	4.64
bird	0.09	5.31	2.65
cat	<b>2.9</b>	-4.22	5.1
deer	4.48	-4.19	2.64
dog	8.02	3.58	5.55
frog	3.78	4.49	<b>-4.34</b>
horse	1.06	-4.37	-1.5
ship	-0.36	-2.09	-4.79
truck	-0.72	-2.93	6.14



# Toy example

Consider: 3 training examples, 3 classes.

With the scores



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>



# Функция ошибки (Loss function)

Consider: 3 training examples, 3 classes.  
With the scores



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>

A **loss function** tells how good our classifier

$x_i$  - image

$y_i$  - label, element of a set  $\{0, 1, \dots\}$

Loss over the dataset is a sum of loss over examples:

$$L = \frac{1}{N} \sum_i L_i(f(x_i, W), y_i)$$

# Multiclass SVM (hinge) loss

Consider: 3 training examples, 3 classes.

With the scores



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>

$x_i$  - image

$y_i$  - label, element of a set  $\{0, 1, \dots\}$

scores vector

$$s = f(x_i, W) = [s_0, \dots, s_{y_i}, \dots]$$

$s_{y_i}$  element corresponds  
to ground truth label  $y_i$

SVM loss

$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$
$$= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

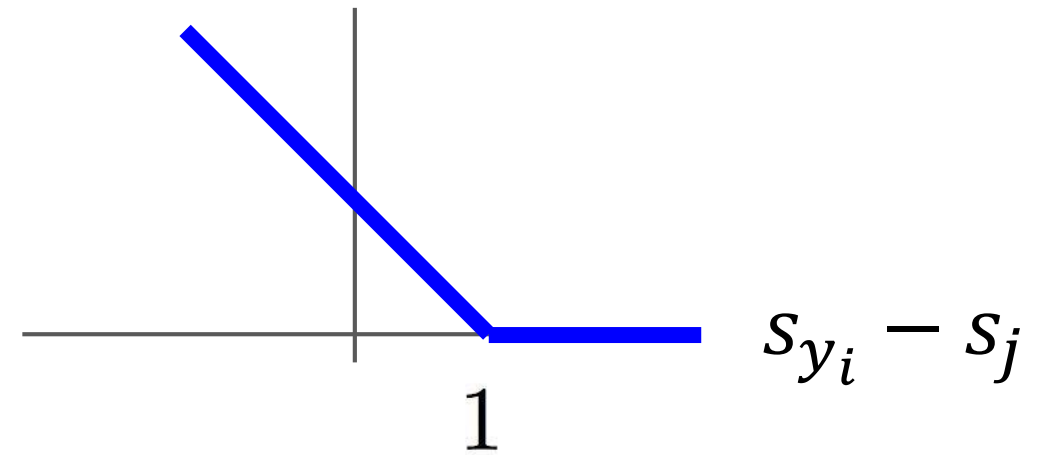
# Multiclass SVM (hinge) loss

Consider: 3 training examples, 3 classes.

With the scores



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>



SVM loss

$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$
$$= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

# Multiclass SVM (hinge) loss

Consider: 3 training examples, 3 classes.  
With the scores



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
Loss	<b>2.9</b>		

$x_i$  - image

$y_i$  - label, element of a set  $\{0, 1, \dots\}$

scores vector

$$s = f(x_i, W) = [s_0, \dots, s_{y_i}, \dots]$$

$s_{y_i}$  element corresponds  
to ground truth label  $y_i$

SVM loss

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$\begin{aligned} &= \max(0, 5.1 - 3.2 + 1) \\ &+ \max(0, -1.7 - 3.2 + 1) \\ &= \max(0, 2.9) + \max(0, -3.9) \\ &= 2.9 + 0 \\ &= 2.9 \end{aligned}$$



# Multiclass SVM (hinge) loss

Consider: 3 training examples, 3 classes.

With the scores



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
Loss	2.9	0	

$x_i$  - image

$y_i$  - label, element of a set  $\{0, 1, \dots\}$

scores vector

$$s = f(x_i, W) = [s_0, \dots, s_{y_i}, \dots]$$

$s_{y_i}$  element corresponds  
to ground truth label  $y_i$

SVM loss

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$= \max(0, 1.3 - 4.9 + 1)$$

$$+ \max(0, 2.0 - 4.9 + 1)$$

$$= \max(0, -2.6) + \max(0, -1.9)$$

$$= 0 + 0$$

$$= 0$$

# Multiclass SVM (hinge) loss

Consider: 3 training examples, 3 classes.  
With the scores



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
Loss	2.9	0	<b>12.9</b>

$x_i$  - image

$y_i$  - label, element of a set  $\{0, 1, \dots\}$

scores vector

$$s = f(x_i, W) = [s_0, \dots, s_{y_i}, \dots]$$

$s_{y_i}$  element corresponds  
to ground truth label  $y_i$

SVM loss

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$= \max(0, 2.2 - (-3.1) + 1)$$

$$+ \max(0, 2.5 - (-3.1) + 1)$$

$$= \max(0, 6.3) + \max(0, 6.6)$$

$$= 6.3 + 6.6$$

$$= 12.9$$

# Multiclass SVM (hinge) loss

Consider: 3 training examples, 3 classes.  
With the scores



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
Loss	2.9	0	12.9

$x_i$  - image

$y_i$  - label, element of a set  $\{0, 1, \dots\}$

scores vector

$$s = f(x_i, W) = [s_0, \dots, s_{y_i}, \dots]$$

$s_{y_i}$  element corresponds  
to ground truth label  $y_i$

SVM loss

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Loss over the dataset  $L = \frac{1}{N} \sum_{i=1}^N L_i$

$$L = (2.9 + 0 + 12.9)/3 \\ = 5.27$$

# Multiclass SVM (hinge) loss

Consider: 3 training examples, 3 classes.

With the scores



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
Loss	2.9	0	12.9

$x_i$  - image

$y_i$  - label, element of a set  $\{0, 1, \dots\}$

scores vector

$$s = f(x_i, W) = [s_0, \dots, s_{y_i}, \dots]$$

$s_{y_i}$  element corresponds  
to ground truth label  $y_i$

SVM loss

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Q1: What happens to  
loss if car scores  
change a bit?



# Multiclass SVM (hinge) loss

Consider: 3 training examples, 3 classes.

With the scores



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
Loss	2.9	0	12.9

$x_i$  - image

$y_i$  - label, element of a set  $\{0, 1, \dots\}$

scores vector

$$s = f(x_i, W) = [s_0, \dots, s_{y_i}, \dots]$$

$s_{y_i}$  element corresponds  
to ground truth label  $y_i$

SVM loss

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Q2: what is the  
min/max possible  
loss?

# Multiclass SVM (hinge) loss

Consider: 3 training examples, 3 classes.

With the scores



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
Loss	2.9	0	12.9

$x_i$  - image

$y_i$  - label, element of a set  $\{0, 1, \dots\}$

scores vector

$$s = f(x_i, W) = [s_0, \dots, s_{y_i}, \dots]$$

$s_{y_i}$  element corresponds  
to ground truth label  $y_i$

SVM loss

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Q3: At initialization  $W$   
is small so all  $s \approx 0$ .

What is the loss?

# Multiclass SVM (hinge) loss

Consider: 3 training examples, 3 classes.

With the scores



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
Loss	2.9	0	12.9

$x_i$  - image

$y_i$  - label, element of a set  $\{0, 1, \dots\}$

scores vector

$$s = f(x_i, W) = [s_0, \dots, s_{y_i}, \dots]$$

$s_{y_i}$  element corresponds  
to ground truth label  $y_i$

SVM loss

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Q4: What if the sum  
was over all classes?  
(including  $j = y_i$ )

# Multiclass SVM (hinge) loss

Consider: 3 training examples, 3 classes.

With the scores



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
Loss	2.9	0	12.9

$x_i$  - image

$y_i$  - label, element of a set  $\{0, 1, \dots\}$

scores vector

$$s = f(x_i, W) = [s_0, \dots, s_{y_i}, \dots]$$

$s_{y_i}$  element corresponds  
to ground truth label  $y_i$

SVM loss

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Q5: What if we used  
mean instead of  
sum?



# Multiclass SVM (hinge) loss

Consider: 3 training examples, 3 classes.

With the scores



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
Loss	2.9	0	12.9

$x_i$  - image

$y_i$  - label, element of a set  $\{0, 1, \dots\}$

scores vector

$$s = f(x_i, W) = [s_0, \dots, s_{y_i}, \dots]$$

$s_{y_i}$  element corresponds  
to ground truth label  $y_i$

SVM loss

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Q6: What if we used

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)^2$$

# Multiclass SVM (hinge) loss: Code Example

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

```
def L_i_vectorized(x, y, W):  
    scores = W.dot(x)  
    margins = np.maximum(0, scores - scores[y] + 1)  
    margins[y] = 0  
    loss_i = np.sum(margins)  
    return loss_i
```

$$f(x, W) = Wx$$

$$L = \frac{1}{N} \sum_{i=1}^N \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1)$$

E.g. Suppose that we found a  $W$  such that  $L = 0$ .  
Is this  $W$  unique?

$$f(x, W) = Wx$$

$$L = \frac{1}{N} \sum_{i=1}^N \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1)$$

E.g. Suppose that we found a  $W$  such that  $L = 0$ .  
Is this  $W$  unique?

**No!  $2W$  is also has  $L = 0$ !**



Consider: 3 training examples, 3 classes.  
With the scores



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
Loss	2.9	0	

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Before:

$$\begin{aligned} &= \max(0, 1.3 - 4.9 + 1) \\ &+ \max(0, 2.0 - 4.9 + 1) \\ &= \max(0, -2.6) + \max(0, -1.9) \\ &= 0 + 0 \\ &= 0 \end{aligned}$$

With 2W:

$$\begin{aligned} &= \max(0, 2.6 - 9.8 + 1) \\ &+ \max(0, 4.0 - 9.8 + 1) \\ &= \max(0, -6.2) + \max(0, -4.8) \\ &= 0 + 0 \end{aligned}$$

$$f(x, W) = Wx$$


$$L = \frac{1}{N} \sum_{i=1}^N \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1)$$

E.g. Suppose that we found a  $W$  such that  $L = 0$ .  
Is this  $W$  unique?

**No!  $2W$  is also has  $L = 0$ !**

**How do we choose between  $W$  and  $2W$ ?**

# Регуляризация

$$L(W) = \frac{1}{N} \sum_{i=1}^N L_i(f(x_i, W), y_i)$$


**Data loss:** Model predictions  
should match training data

# Регуляризация

$$L(W) = \underbrace{\frac{1}{N} \sum_{i=1}^N L_i(f(x_i, W), y_i)}_{\text{Data loss}} + \underbrace{\lambda R(W)}_{\text{Regularization}}$$

**Data loss:** Model predictions should match training data

**Regularization:** Prevent the model from doing *too* well on training data

# Регуляризация

$$L(W) = \underbrace{\frac{1}{N} \sum_{i=1}^N L_i(f(x_i, W), y_i)}_{\text{Data loss: Model predictions should match training data}} + \underbrace{\lambda R(W)}_{\text{Regularization: Prevent the model from doing too well on training data}}$$

$\lambda$  - regularization strength (hyperparameter)

**Data loss:** Model predictions should match training data

**Regularization:** Prevent the model from doing *too* well on training data



# Регуляризация

$$L(W) = \underbrace{\frac{1}{N} \sum_{i=1}^N L_i(f(x_i, W), y_i)}_{\text{Data loss: Model predictions should match training data}} + \underbrace{\lambda R(W)}_{\text{Regularization: Prevent the model from doing too well on training data}}$$

$\lambda$  - regularization strength (hyperparameter)

**Data loss:** Model predictions should match training data

**Regularization:** Prevent the model from doing *too* well on training data

## Примеры

L2 regularization  $R(W) = \sum_k \sum_l W_{k,l}^2$

L1 regularization  $R(W) = \sum_k \sum_l |W_{k,l}|$

Elastic net (L1 + L2)  $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$

# Регуляризация

$$L(W) = \underbrace{\frac{1}{N} \sum_{i=1}^N L_i(f(x_i, W), y_i)}_{\text{Data loss: Model predictions should match training data}} + \underbrace{\lambda R(W)}_{\text{Regularization: Prevent the model from doing too well on training data}}$$

$\lambda$  - regularization strength (hyperparameter)

**Data loss:** Model predictions should match training data

**Regularization:** Prevent the model from doing *too* well on training data

## Примеры

L2 regularization

$$R(W) = \sum_k \sum_l W_{k,l}^2$$

L1 regularization

$$R(W) = \sum_k \sum_l |W_{k,l}|$$

Elastic net (L1 + L2)

$$R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$$

Dropout

Batch normalization

(will see later)

# Регуляризация

$$L(W) = \underbrace{\frac{1}{N} \sum_{i=1}^N L_i(f(x_i, W), y_i)}_{\text{Data loss: Model predictions should match training data}} + \underbrace{\lambda R(W)}_{\text{Regularization: Prevent the model from doing too well on training data}}$$

$\lambda$  - regularization strength (hyperparameter)

**Data loss:** Model predictions should match training data

**Regularization:** Prevent the model from doing *too* well on training data

Зачем нужна регуляризация?

- управление значимостью признаков
- для увеличения обобщающей способности (простая модель лучше)

# Регуляризация: значимость признаков

L2 regularization

$$R(W) = \sum_k \sum_l W_{k,l}^2$$

$$x = [1, 1, 1, 1]$$

$$w_1 = [1, 0, 0, 0]$$

$$w_2 = [0.25, 0.25, 0.25, 0.25]$$

$$w_1^T x = w_2^T x = 1$$

# Регуляризация: значимость признаков

L2 regularization

$$R(W) = \sum_k \sum_l W_{k,l}^2$$

$$x = [1, 1, 1, 1]$$

$$w_1 = [1, 0, 0, 0]$$

$$w_2 = [0.25, 0.25, 0.25, 0.25]$$

$$w_1^T x = w_2^T x = 1$$

L2 регуляризация  
«размазывает» веса



# Регуляризация: значимость признаков

L2 regularization

$$R(W) = \sum_k \sum_l W_{k,l}^2$$

$$x = [1, 1, 1, 1]$$

$$w_1 = [1, 0, 0, 0]$$

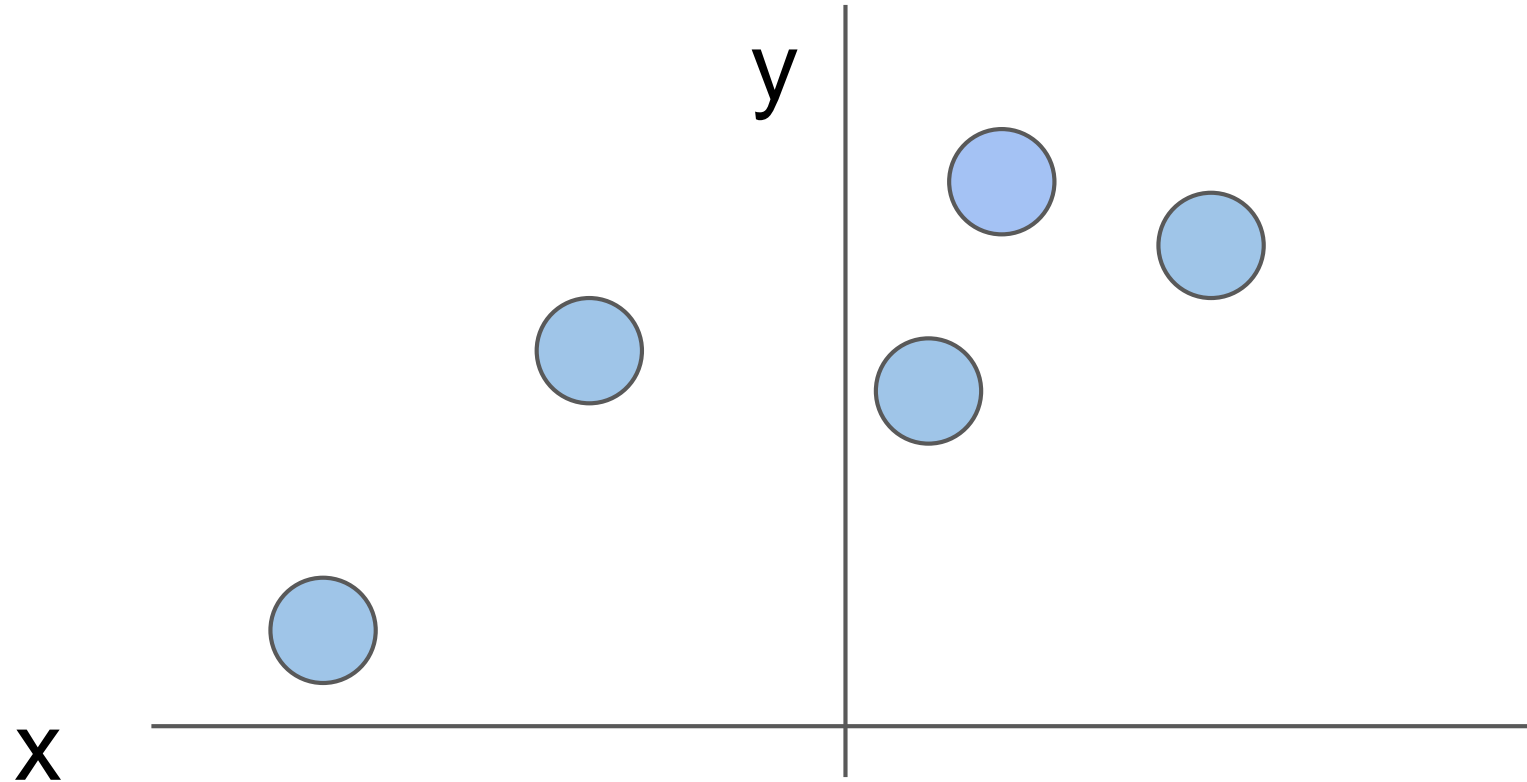
$$w_2 = [0.25, 0.25, 0.25, 0.25]$$

$$w_1^T x = w_2^T x = 1$$

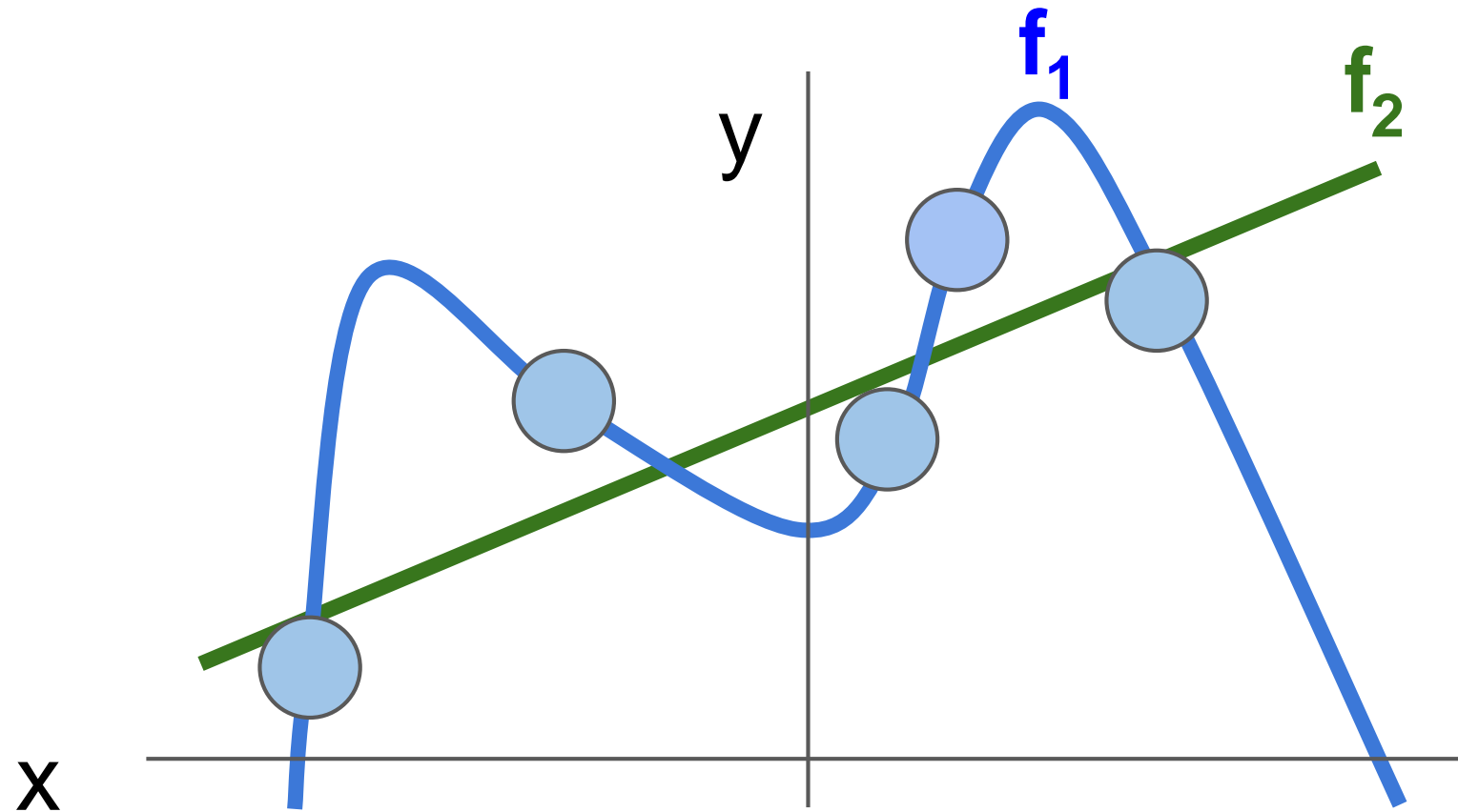
L2 регуляризация  
«размазывает» веса

L1 регуляризация оставляет  
только самые значимые веса  
(можно делать селекцию  
признаков)

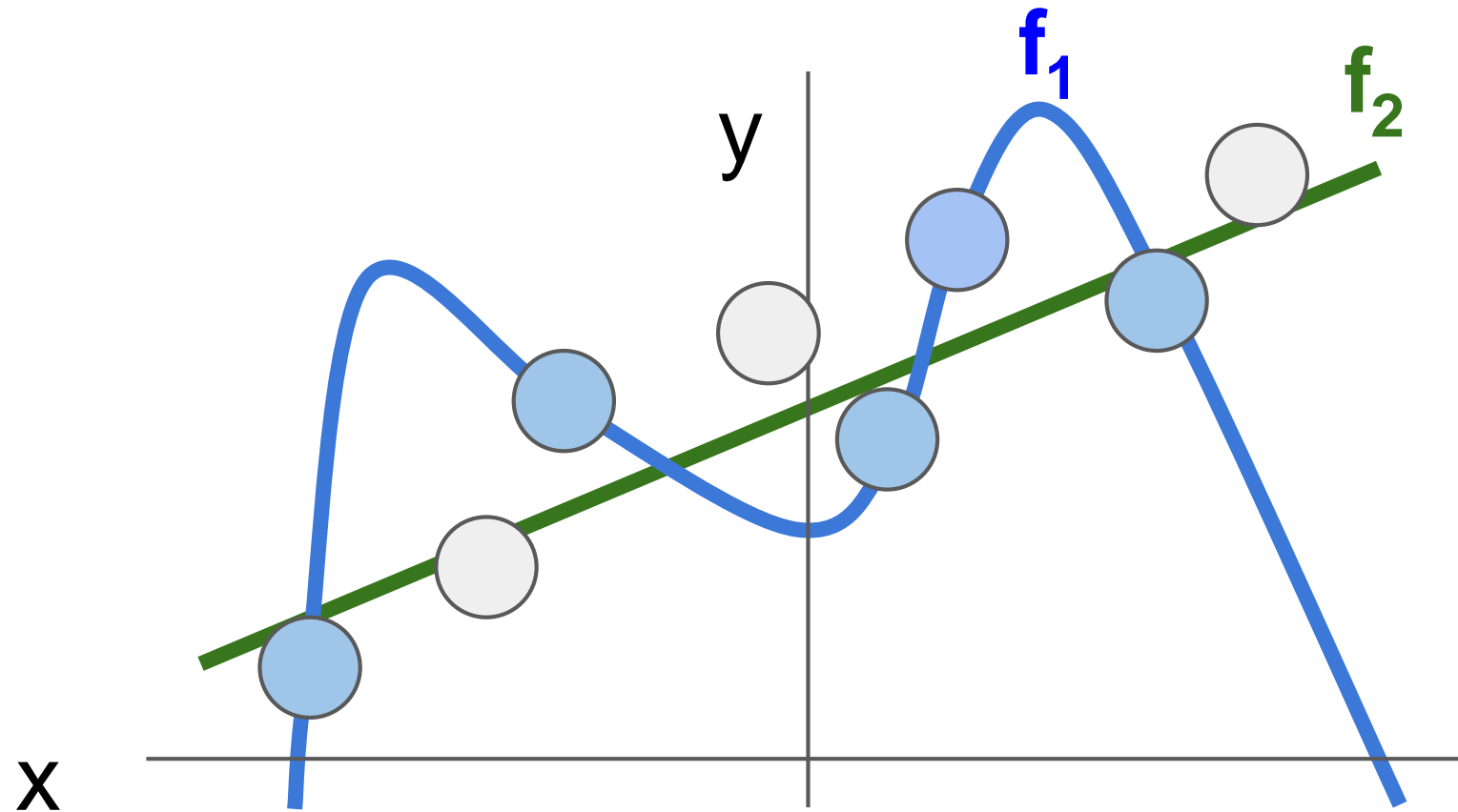
# Регуляризация: уменьшение переобучения



# Регуляризация: уменьшение переобучения



# Регуляризация: уменьшение переобучения



Регуляризация мешает модели слишком хорошо работать на тренировочном наборе, мы не переобучаемся на шумы

# Machine learning principles

- Maximum Likelihood Estimate (MLE) - метод максимального правдоподобия
- Maximum A Posteriori probability estimate (MAP) - оценка апостериорного максимума



# Maximum Likelihood Estimate (MLE)

Метод максимального правдоподобия

$x_i$  - данные из  $X$

$y_i$  - ответы из  $Y$  (например, классы)

$P(Y|X = x_i, W)$  - модель дающая распределение вероятностей ответов

likelihood  $\prod_i P(Y = y_i | X = x_i, W)$

# Maximum Likelihood Estimate (MLE)

Метод максимального правдоподобия

$x_i$  - данные из  $X$

$y_i$  - ответы из  $Y$  (например, классы)

$P(Y|X = x_i, W)$  - модель дающая распределение вероятностей ответов

likelihood  $\prod_i P(Y = y_i | X = x_i, W)$

maximum likelihood  $\max_W \prod_i P(Y = y_i | X = x_i, W) \Rightarrow \max_W \sum_i \log P(Y = y_i | X = x_i, W)$

Задача найти такие параметры  $W$ , чтобы вероятность правильного ответа на тренировочном наборе была наибольшей

# Maximum A Posteriori probability estimate (MAP)

Оценка апостериорного максимума

$x_i$  - данные из  $X$

$y_i$  - ответы из  $Y$  (например, классы)

$P(Y|X = x_i, W)$  - модель дающая распределение вероятностей ответов

Теорема Байеса

$$P(W|Y, X) = \frac{P(Y|X, W)P(W)}{P(Y, X)}$$

$$\textit{posterior} = \frac{\textit{likelihood} \cdot \textit{prior}}{\textit{evidence}}$$

# Maximum A Posteriori probability estimate (MAP)

Оценка апостериорного максимума

$x_i$  - данные из  $X$

$y_i$  - ответы из  $Y$  (например, классы)

$P(Y|X = x_i, W)$  - модель дающая распределение вероятностей ответов

$$\begin{aligned}\hat{W}_{MAP} &= \arg \max_W P(W|Y, X) \\ &= \arg \max_W \frac{P(Y|X, W)P(W)}{P(Y, X)} \\ &= \arg \max_W P(Y|X, W)P(W) \\ &= \arg \max_W \log P(Y|X, W)P(W) \\ &= \arg \max_W \log P(Y|X, W) + \log P(W)\end{aligned}$$

Задача найти наиболее вероятные параметры  $W$ , исходя из тренировочных данных

# Regularization: Maximum A Posteriori probability estimate (MAP)

$$\hat{W}_{MAP} = \arg \max_W \log P(Y|X, W) + \log P(W)$$

Выбор априорных распределений параметров  $W$

- Распределение Гаусса

$$P(W|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(W-\mu)^2}{2\sigma^2}}$$

- Распределение Лапласа

$$P(W|\mu, b) = \frac{1}{2b} \exp\left(-\frac{|W - \mu|}{b}\right)$$



# Regularization: Maximum A Posteriori probability estimate (MAP)

$$\hat{W}_{MAP} = \arg \max_W \log P(Y|X, W) + \log P(W)$$

Выбор априорных распределений параметров  $W$

- Распределение Гаусса

$$P(W|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(W-\mu)^2}{2\sigma^2}}$$

Обычно берут распределения с нулевым средним  $\mu = 0$

- Распределение Лапласа

$$P(W|\mu, b) = \frac{1}{2b} \exp\left(-\frac{|W - \mu|}{b}\right)$$

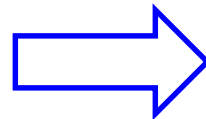
# Regularization: Maximum A Posteriori probability estimate (MAP)

$$\hat{W}_{MAP} = \arg \max_W \log P(Y|X, W) + \log P(W)$$

Выбор априорных распределений параметров  $W$

- Распределение Гаусса

$$P(W|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(W-\mu)^2}{2\sigma^2}}$$

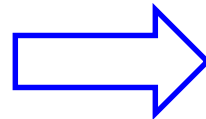


L2 regularization

$$R(W) = \sum_j W_j^2$$

- Распределение Лапласа

$$P(W|\mu, b) = \frac{1}{2b} \exp\left(-\frac{|W - \mu|}{b}\right)$$



L1 regularization

$$R(W) = \sum_j |W_j|$$

# Softmax Classifier (Multinomial Logistic Regression)



Want to interpret raw classifier scores as **probabilities**

cat	<b>3.2</b>
car	5.1
frog	-1.7

# Softmax Classifier (Multinomial Logistic Regression)



Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i, W) \quad P(Y = y_i | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \quad \text{probability of } x_i \text{ image has } y_i \text{ label}$$

cat	<b>3.2</b>
car	5.1
frog	-1.7

# Softmax Classifier (Multinomial Logistic Regression)



Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i, W) \quad P(Y = y_i | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \quad \text{probability of } x_i \text{ image has } y_i \text{ label}$$

Softmax function

cat	<b>3.2</b>
car	5.1
frog	-1.7

# Softmax Classifier (Multinomial Logistic Regression)



Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i, W) \quad P(Y = y_i | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \quad \text{probability of } x_i \text{ image has } y_i \text{ label}$$

Probabilities  
must be  $\geq 0$

cat	3.2
car	5.1
frog	-1.7

exp →

24.5
164.0
0.18

unnormalized  
probabilities

# Softmax Classifier (Multinomial Logistic Regression)



Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i, W) \quad P(Y = y_i | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \quad \text{probability of } x_i \text{ image has } y_i \text{ label}$$

Probabilities  
must be  $\geq 0$

Probabilities  
must sum to 1

cat	3.2
car	5.1
frog	-1.7

exp

24.5
164.0
0.18

normalize

0.13
0.87
0.00

unnormalized  
probabilities

probabilities



# Softmax Classifier (Multinomial Logistic Regression)



Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i, W) \quad P(Y = y_i | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \quad \text{probability of } x_i \text{ image has } y_i \text{ label}$$

Probabilities  
must be  $\geq 0$

Probabilities  
must sum to 1

cat  
car  
frog

3.2  
5.1  
-1.7

exp

24.5  
164.0  
0.18

normalize

0.13  
0.87  
0.00

unnormalized  
log probabilities  
or logits

unnormalized  
probabilities

probabilities

# Softmax Classifier (Multinomial Logistic Regression)

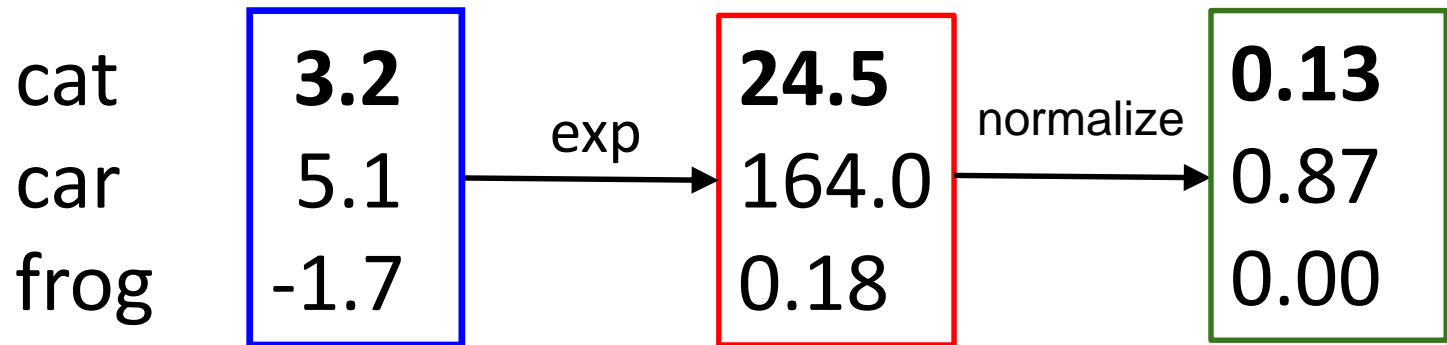


Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i, W) \quad P(Y = y_i | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \quad \text{probability of } x_i \text{ image has } y_i \text{ label}$$

Probabilities  
must be  $\geq 0$

Probabilities  
must sum to 1



unnormalized  
log probabilities  
or logits

unnormalized  
probabilities

probabilities

How to calculate loss?

# Softmax Classifier: Maximum likelihood estimation



Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i, W) \quad P(Y = y_i | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \quad \text{probability of } x_i \text{ image has } y_i \text{ label}$$

Probabilities  
must be  $\geq 0$

Probabilities  
must sum to 1

Want to maximize the likelihood

$$\max_W \prod_i P(Y = y_i | X = x_i)$$

or log likelihood

$$\max_W \sum_i \log P(Y = y_i | X = x_i)$$

or minimize negative log likelihood

$$L_i = -\log P(Y = y_i | X = x_i)$$

cat  
car  
frog

3.2  
5.1  
-1.7

exp

24.5  
164.0  
0.18

normalize

0.13  
0.87  
0.00

unnormalized  
log probabilities  
or logits

unnormalized  
probabilities

probabilities

# Softmax Classifier: Maximum likelihood estimation



Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i, W) \quad P(Y = y_i | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \quad \text{probability of } x_i \text{ image has } y_i \text{ label}$$

Probabilities  
must be  $\geq 0$

Probabilities  
must sum to 1

$$L_i = -\log P(Y = y_i | X = x_i)$$

cat  
car  
frog

3.2  
5.1  
-1.7

exp

24.5  
164.0  
0.18

normalize

0.13  
0.87  
0.00

unnormalized  
log probabilities  
or logits

unnormalized  
probabilities

probabilities

$$L_i = -\log(0.13) = 0.89$$

# Softmax Classifier: Cross-entropy loss

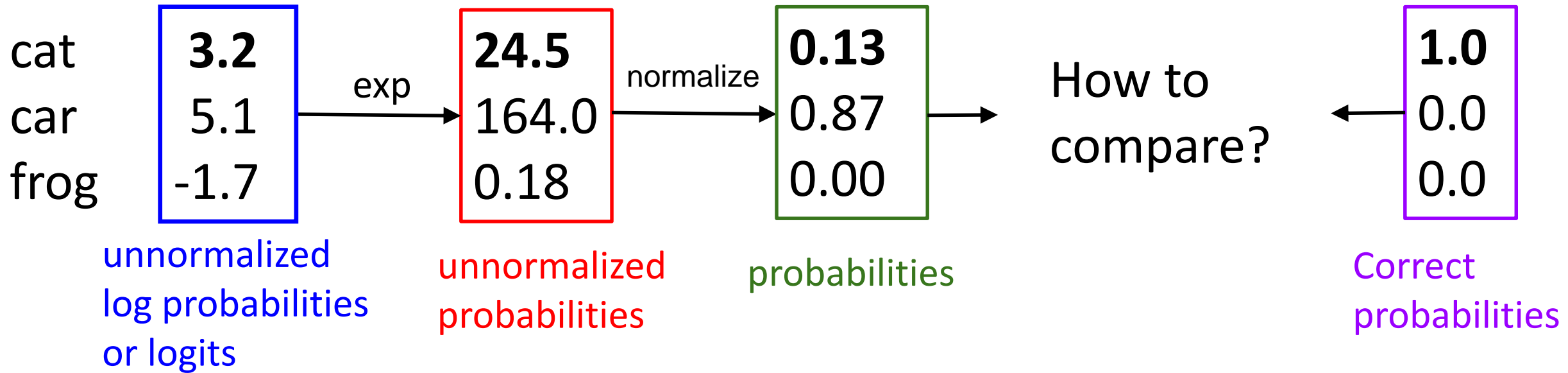


Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i, W) \quad P(Y = y_i | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \quad \text{probability of } x_i \text{ image has } y_i \text{ label}$$

Probabilities  
must be  $\geq 0$

Probabilities  
must sum to 1



# Softmax Classifier: Cross-entropy loss

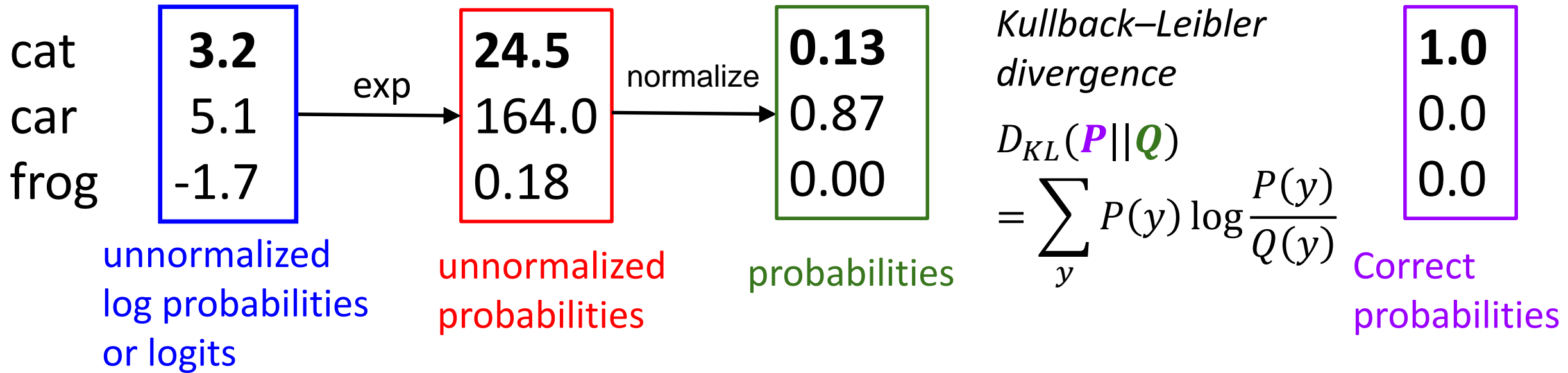


Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i, W) \quad P(Y = y_i | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \quad \text{probability of } x_i \text{ image has } y_i \text{ label}$$

Probabilities  
must be  $\geq 0$

Probabilities  
must sum to 1





# Softmax Classifier: Cross-entropy loss



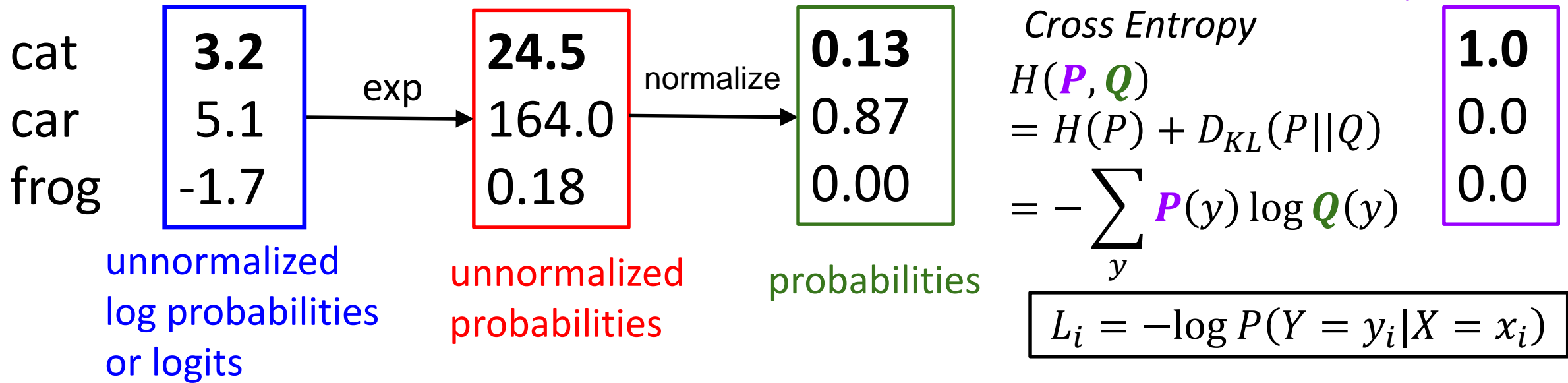
Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i, W) \quad P(Y = y_i | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \quad \text{probability of } x_i \text{ image has } y_i \text{ label}$$

Probabilities  
must be  $\geq 0$

Probabilities  
must sum to 1

Correct  
probabilities





# Softmax Classifier (Multinomial Logistic Regression)



cat	<b>3.2</b>
car	5.1
frog	-1.7

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i, W) \quad P(Y = y_i | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \quad \begin{array}{l} \text{probability of } x_i \\ \text{image has } y_i \text{ label} \end{array}$$

Maximize probability of correct class

Putting it all together

$$L_i = -\log P(Y = y_i | X = x_i)$$

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

# Softmax Classifier (Multinomial Logistic Regression)



cat	<b>3.2</b>
car	5.1
frog	-1.7

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i, W) \quad P(Y = y_i | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \quad \text{probability of } x_i \text{ image has } y_i \text{ label}$$

Maximize probability of correct class

Putting it all together

$$L_i = -\log P(Y = y_i | X = x_i)$$

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

Q1: What is the possible min/max loss  $L_i$ ?

# Softmax Classifier (Multinomial Logistic Regression)



cat	<b>3.2</b>
car	5.1
frog	-1.7

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i, W) \quad P(Y = y_i | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \quad \begin{array}{l} \text{probability of } x_i \\ \text{image has } y_i \text{ label} \end{array}$$

Maximize probability of correct class

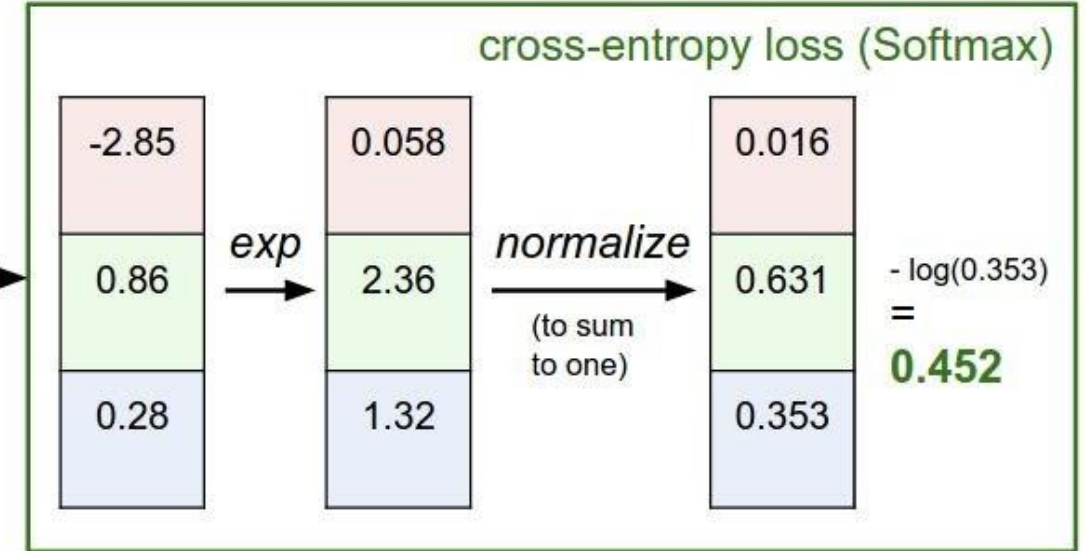
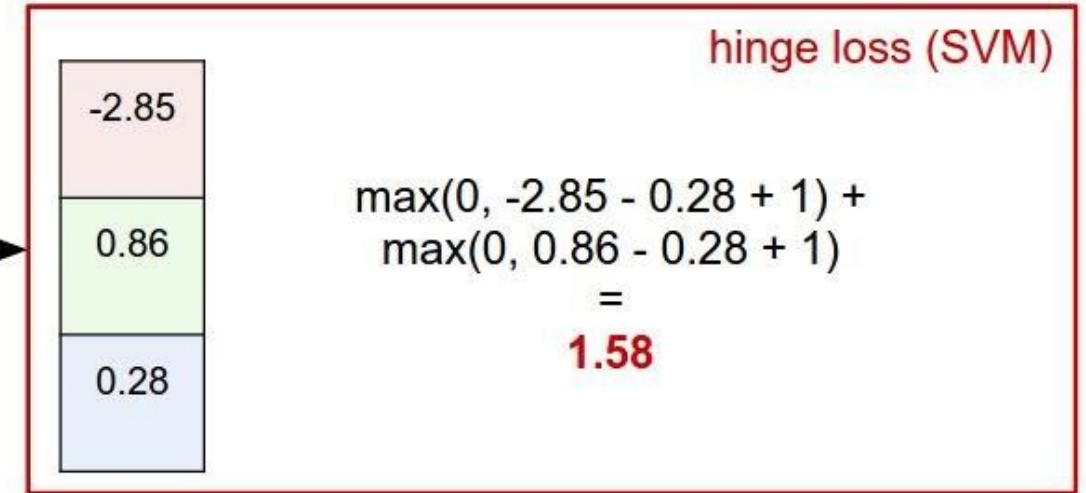
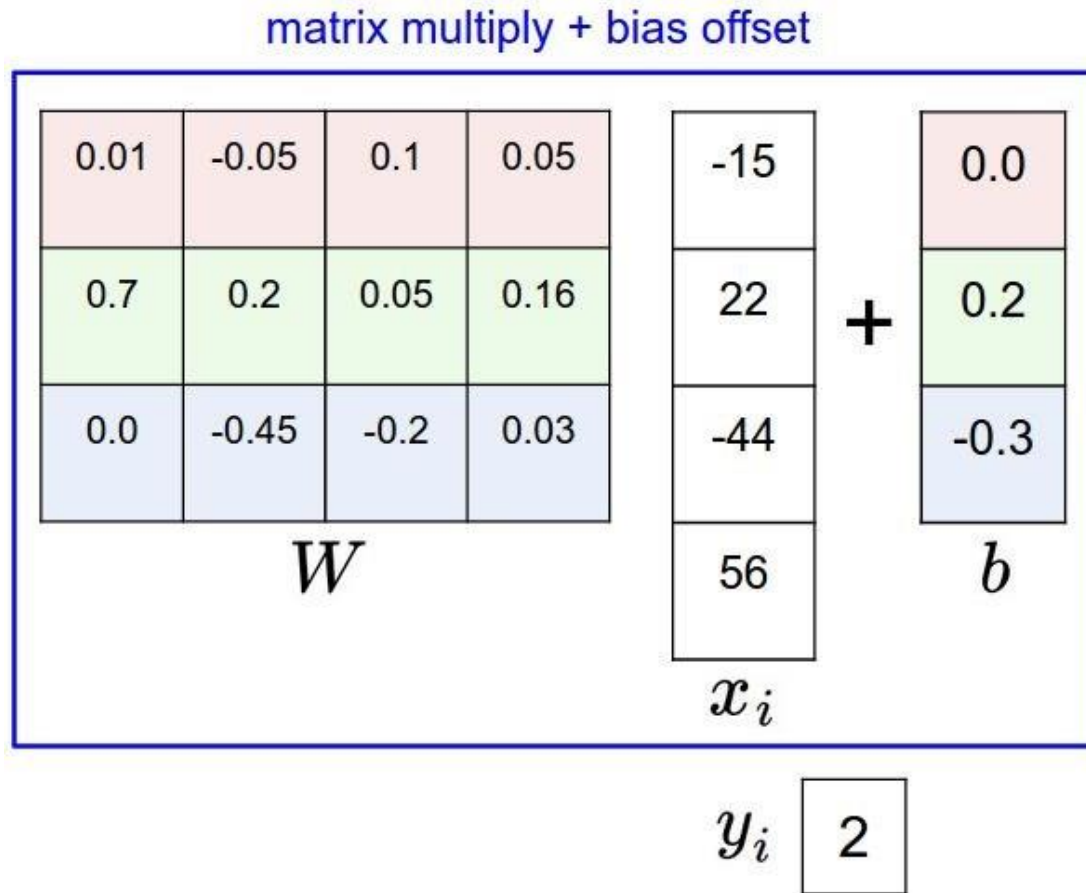
Putting it all together

$$L_i = -\log P(Y = y_i | X = x_i)$$

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

Q2: Usually at  
initialization  $W$  is small  
so all  $s \approx 0$ .  
What is the loss?

# Softmax vs SVM



# Softmax vs SVM

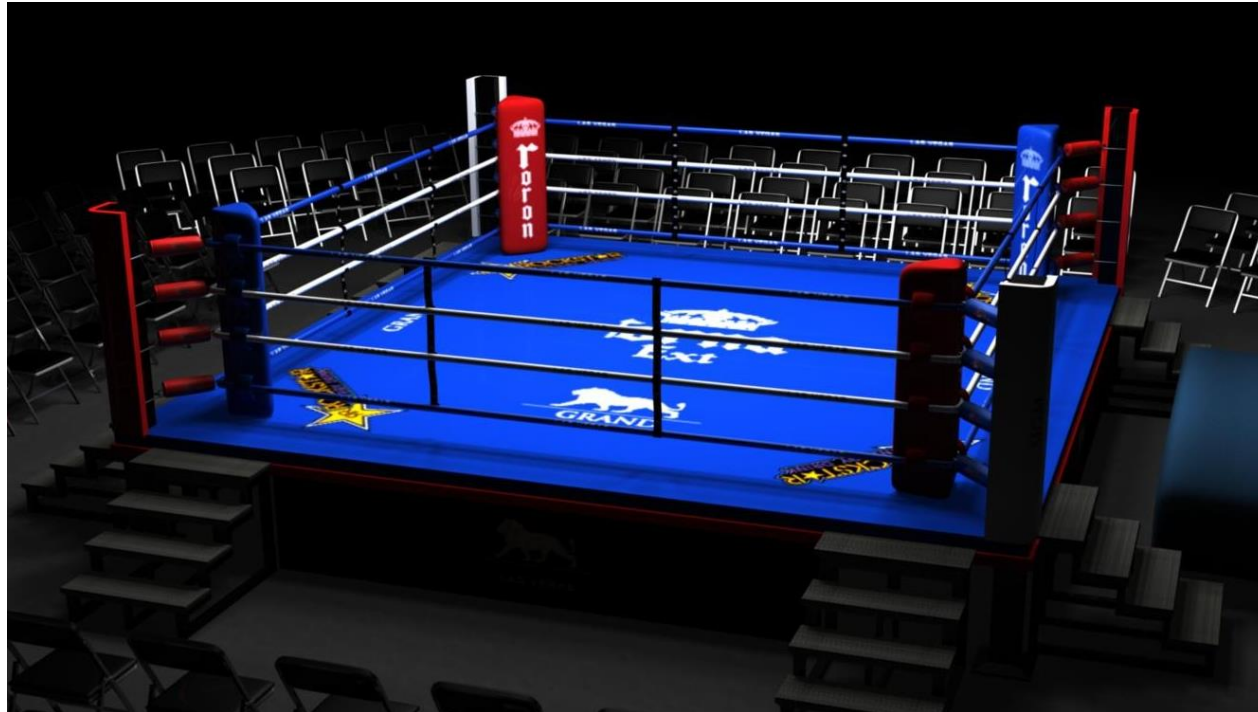
Softmax

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

VS

SVM

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$



# Softmax vs SVM

Softmax

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

vs

SVM

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

assume scores:

[10, -2, 3]

[10, 9, 9]

[10, -100, -100]

and  $y_i = 0$

Q: Suppose I take a datapoint and I jiggle a bit (changing its score slightly). What happens to the loss in both cases?

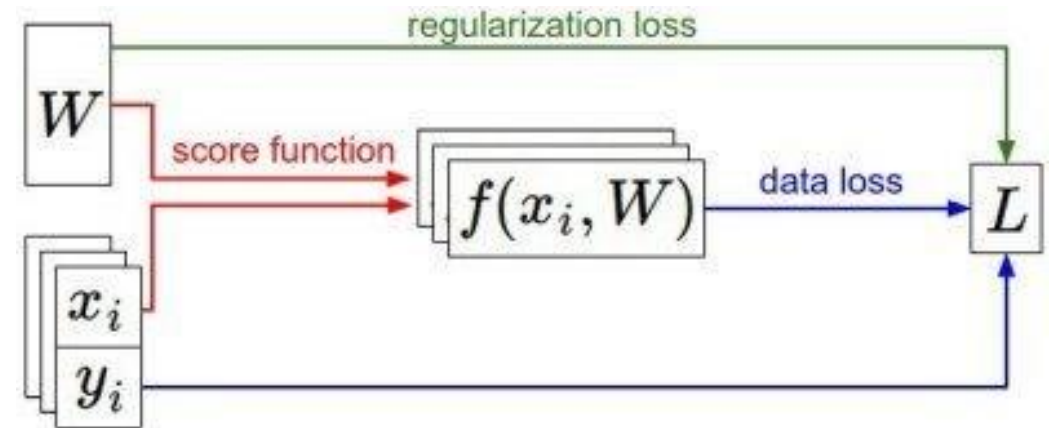
# Recap

- We have some dataset of  $(x_i, y_i)$
- We have a **score function**:  $s = f(x_i, W)$
- We have a **loss function**:

$$L_i = -\log \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \quad \text{Softmax}$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \quad \text{SVM}$$

$$L = \frac{1}{N} \sum_{i=1}^N L_i + \lambda R(W) \quad \text{Full loss}$$





# Recap

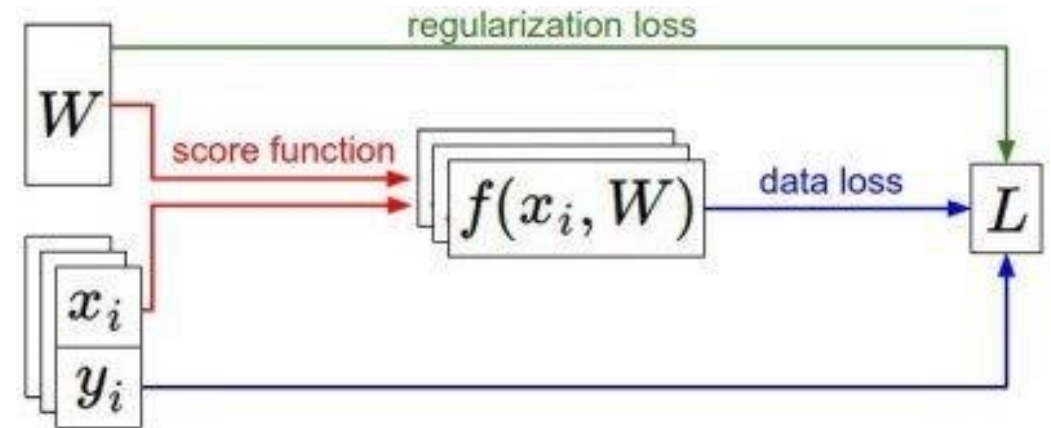
How do we find the best  $W$ ?

- We have some dataset of  $(x_i, y_i)$
- We have a **score function**:  $s = f(x_i, W)$
- We have a **loss function**:

$$L_i = -\log \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \quad \text{Softmax}$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \quad \text{SVM}$$

$$L = \frac{1}{N} \sum_{i=1}^N L_i + \lambda R(W) \quad \text{Full loss}$$



Оптимизация

# Оптимизация





# Оптимизация: случайный поиск



# Оптимизация: случайный поиск

```
# assume X_train is the data where each column is an example (e.g. 3073 x 50,000)  
# assume Y_train are the labels (e.g. 1D array of 50,000)  
# assume the function L evaluates the loss function  
  
bestloss = float("inf") # Python assigns the highest possible float value  
for num in xrange(1000):  
    W = np.random.randn(10, 3073) * 0.0001 # generate random parameters  
    loss = L(X_train, Y_train, W) # get the loss over the entire training set  
    if loss < bestloss: # keep track of the best solution  
        bestloss = loss  
        bestW = W  
    print 'in attempt %d the loss was %f, best %f' % (num, loss, bestloss)  
  
# prints:  
# in attempt 0 the loss was 9.401632, best 9.401632  
# in attempt 1 the loss was 8.959668, best 8.959668  
# in attempt 2 the loss was 9.044034, best 8.959668  
# in attempt 3 the loss was 9.278948, best 8.959668  
# in attempt 4 the loss was 8.857370, best 8.857370  
# in attempt 5 the loss was 8.943151, best 8.857370  
# in attempt 6 the loss was 8.605604, best 8.605604  
# ... (truncated: continues for 1000 lines)
```



# Оптимизация: случайный поиск

```
# Assume X_test is [3073 x 10000], Y_test [10000 x 1]  
scores = Wbest.dot(Xte_cols) # 10 x 10000, the class scores for all test examples  
# find the index with max score in each column (the predicted class)  
Yte_predict = np.argmax(scores, axis = 0)  
# and calculate accuracy (fraction of predictions that are correct)  
np.mean(Yte_predict == Yte)  
# returns 0.1555
```

Для линейного классификатора на CIFAR-10 можно достичь точности 15.5%  
(state of the art is  $\approx 96\%$ )

# Оптимизация: следуем по градиенту





# Оптимизация: следуем по градиенту

В одномерном случае производная скаляр

$$\frac{dL(w)}{dw} = \lim_{h \rightarrow 0} \frac{L(w + h) - L(w)}{h}$$

В многомерном случае градиент – вектор частных производных

Скорость убывания функции в произвольном направлении – скалярное произведение вектора направления и градиента

Наиболее быстро функция убывает при движении в направлении **обратного градиента**

**current W:**

[0.34,  
-1.11,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**loss 1.25347**

**gradient dL/dW:**

[?,  
?,  
?,  
?,  
?,  
?,  
?,  
?,  
?,...]

**current W:**

[0.34,  
-1.11,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**loss 1.25347**

**W + h (first dim):**

[0.34 + **0.0001**,  
-1.11,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**loss 1.25322**

**gradient dL/dW:**

[?,  
?,  
?,  
?,  
?,  
?,  
?,  
?,  
?,...]

**current W:**

[0.34,  
-1.11,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**loss 1.25347**

**W + h (first dim):**

[0.34 + **0.0001**,  
-1.11,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**loss 1.25322**

**gradient dL/dW:**

[-2.5,  
?,  
?,

$$(1.25322 - 1.25347)/0.0001 = -2.5$$

$$\frac{dL(W)}{dW} = \lim_{h \rightarrow 0} \frac{L(W + h) - L(W)}{h}$$

∴,  
?,...]

**current W:**

[0.34,  
-1.11,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**loss 1.25347**

**W + h (second dim):**

[0.34,  
-1.11 + **0.0001**,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**loss 1.25322**

**gradient dL/dW:**

[-2.5,  
?,  
?,  
?,  
?,  
?,  
?,  
?,  
?,...]

**current W:**

[0.34,  
-1.11,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**loss 1.25347**

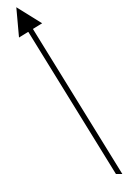
**W + h (second dim):**

[0.34,  
-1.11 + **0.0001**,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**loss 1.25353**

**gradient dL/dW:**

[-2.5,  
**0.6**,  
?,  
?,  
..., ...]


$$(1.25353 - 1.25347) / 0.0001 = 0.6$$

$$\frac{dL(W)}{dW} = \lim_{h \rightarrow 0} \frac{L(W + h) - L(W)}{h}$$

**current W:**

[0.34,  
-1.11,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**loss 1.25347**

**W + h (third dim):**

[0.34,  
-1.11,  
0.78 + **0.0001**,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**loss 1.25347**

**gradient dL/dW:**

[-2.5,  
0.6,  
?,  
?,  
?,  
?,  
?,  
?,  
?,...]



**current W:**

[0.34,  
-1.11,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**loss 1.25347**

**W + h (third dim):**

[0.34,  
-1.11,  
0.78 + **0.0001**,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**loss 1.25347**

**gradient dL/dW:**

[-2.5,  
0.6,  
**0**,  
?,  
?


$$(1.25347 - 1.25347)/0.0001 = 0$$

$$\frac{dL(W)}{dW} = \lim_{h \rightarrow 0} \frac{L(W + h) - L(W)}{h}$$

# Can we do better?

The loss is just a function of  $W$ :

$$L = \frac{1}{N} \sum_{i=1}^N L_i + \lambda \sum_k W_k^2$$

$$L_i = -\log \frac{e^{s_{yi}}}{\sum_j e^{s_j}}$$

$$s = f(x, W) = Wx + b$$

want  $\nabla_W L$

Вычисление  
градиента!



**current W:**

[0.34,  
-1.11,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**loss 1.25347**

$dL/dW = \dots$   
(some function  
data and W)

**gradient  $dL/dW$ :**

[-2.5,  
0.6,  
0,  
0.2,  
0.7,  
-0.5,  
1.1,  
1.3,  
-2.1,...]

# Резюме

- Numerical gradient: approximate, slow, easy to write
- Analytic gradient: exact, fast, error-prone

In practice: Always use analytic gradient, but check implementation with numerical gradient. This is called a **gradient check**.

# Градиентный спуск: Gradient Descent

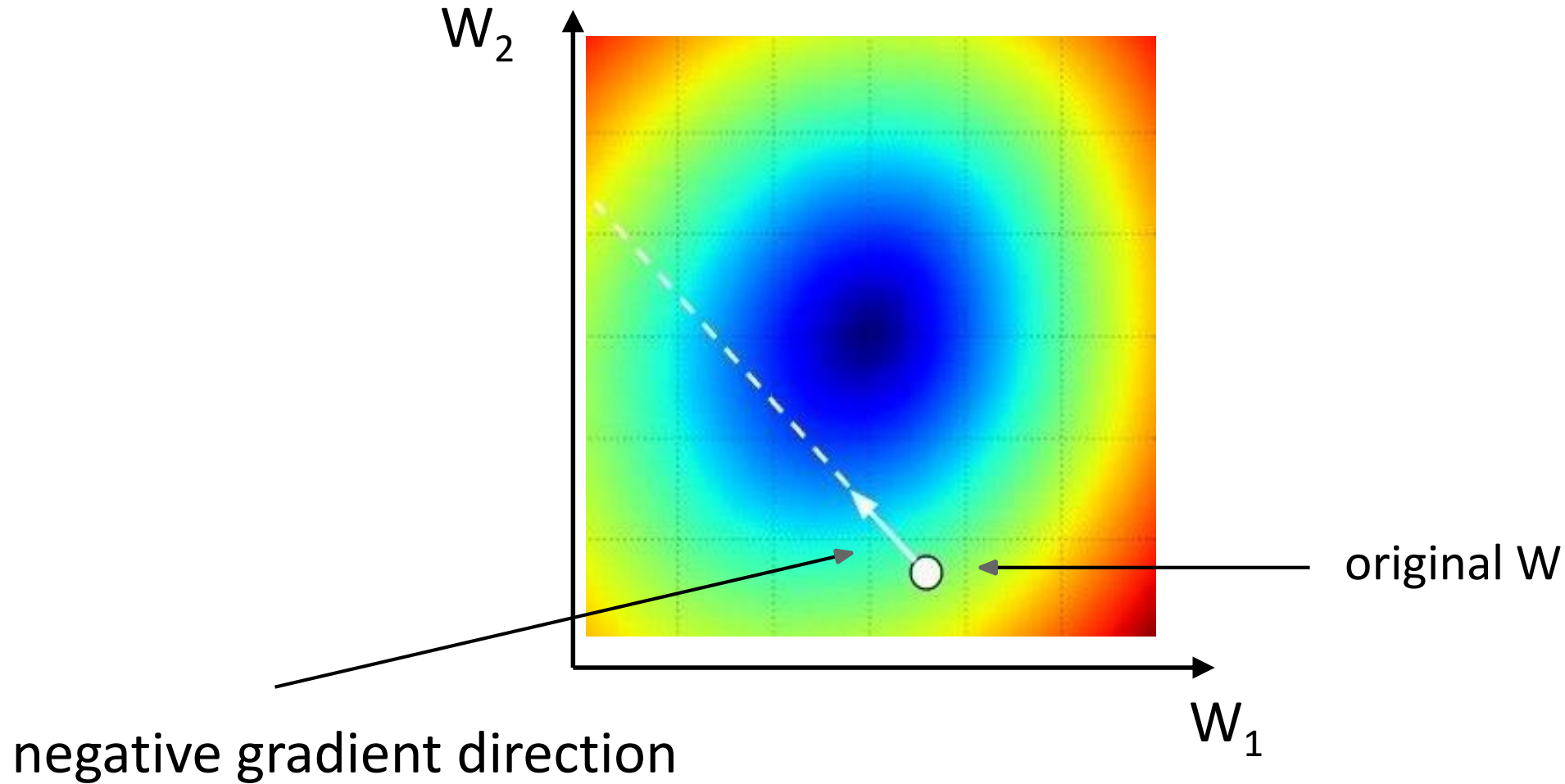
```
# Vanilla Gradient Descent
```

```
while True:
```

```
    weights_grad = evaluate_gradient(loss_fun, data, weights)
```

```
    weights += - step_size * weights_grad # perform parameter update
```

# Пример функции потерь от 2-х переменных



# Стохастический градиентный спуск: Stochastic Gradient Descent (SGD)

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

Full sum expensive when  
N is large!

$$\nabla_W L(W) = \frac{1}{N} \sum_{i=1}^N \nabla_W L_i(x_i, y_i, W) + \lambda \nabla_W R(W)$$

Approximate sum using a  
**minibatch** of examples  
32 / 64 / 128 common

```
# Vanilla Minibatch Gradient Descent
```

```
while True:
```

```
    data_batch = sample_training_data(data, 256) # sample 256 examples
```

```
    weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
```

```
    weights += - step_size * weights_grad # perform parameter update
```



# Стохастический градиентный спуск: Stochastic Gradient Descent (SGD)

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

Full sum expensive when  
N is large!

$$\nabla_W L(W) = \frac{1}{N} \sum_{i=1}^N \nabla_W L_i(x_i, y_i, W) + \lambda \nabla_W R(W)$$

Approximate sum using a  
**minibatch** of examples  
32 / 64 / 128 common

```
# Vanilla Minibatch Gradient Descent
```

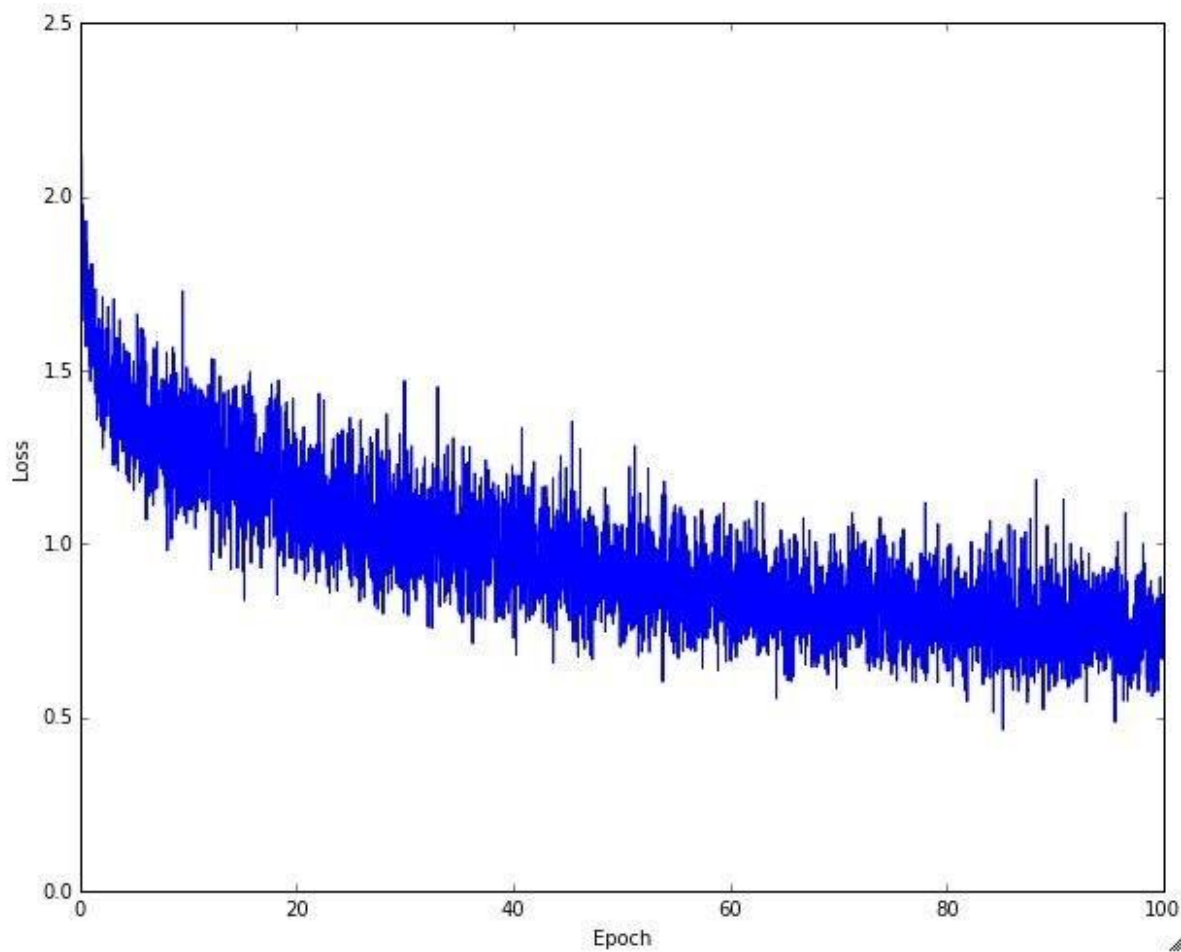
```
while True:
```

```
    data_batch = sample_training_data(data, 256) # sample 256 examples
```

```
    weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
```

```
    weights += - step_size * weights_grad # perform parameter update
```

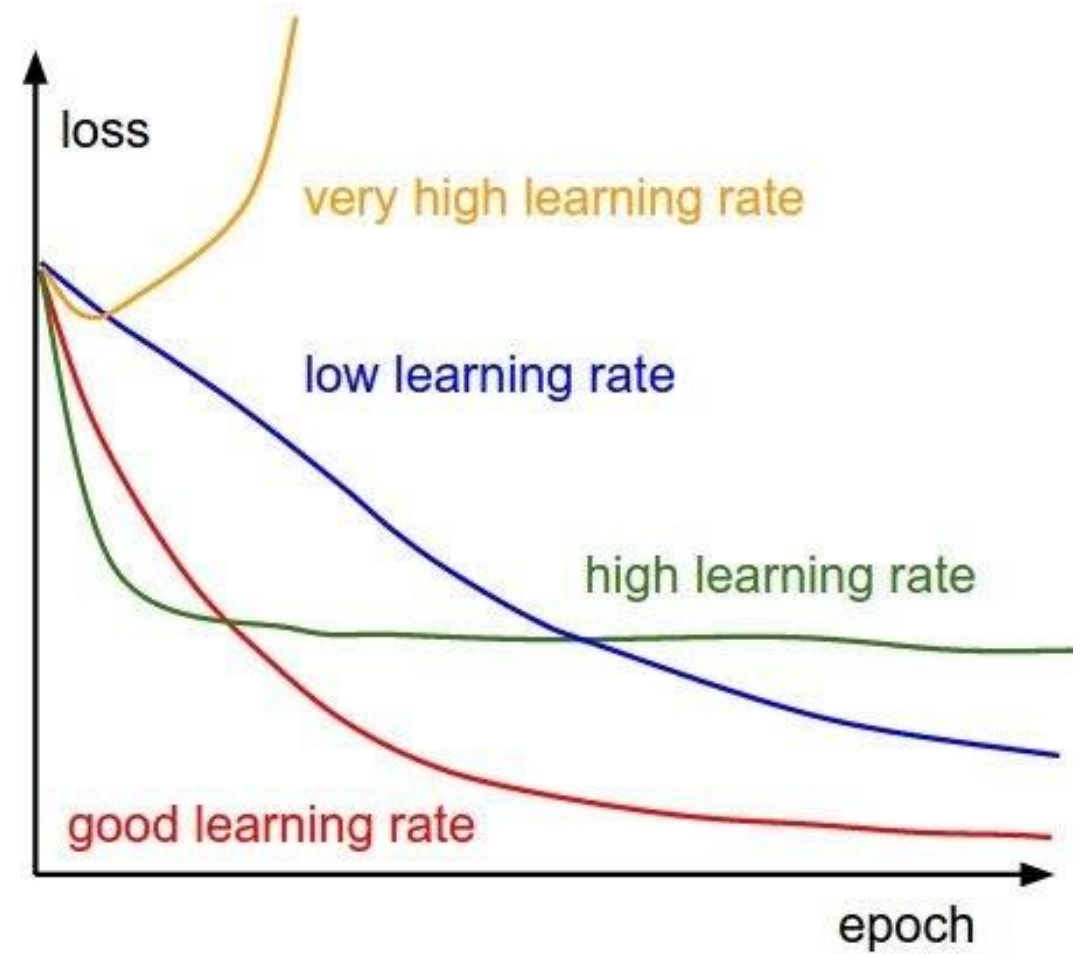
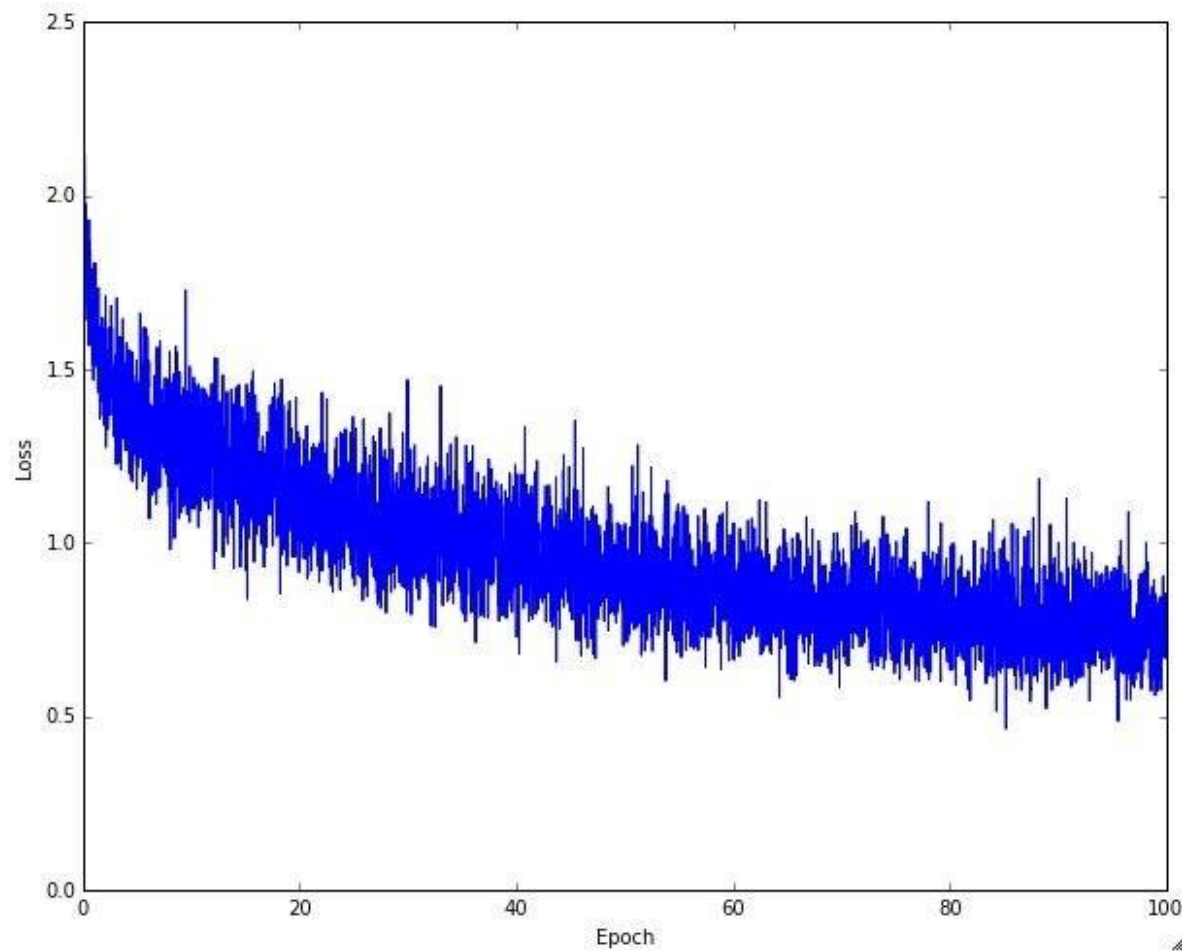
we will look at more  
fancy update formulas  
(momentum, Adagrad,  
RMSProp, Adam, ...)



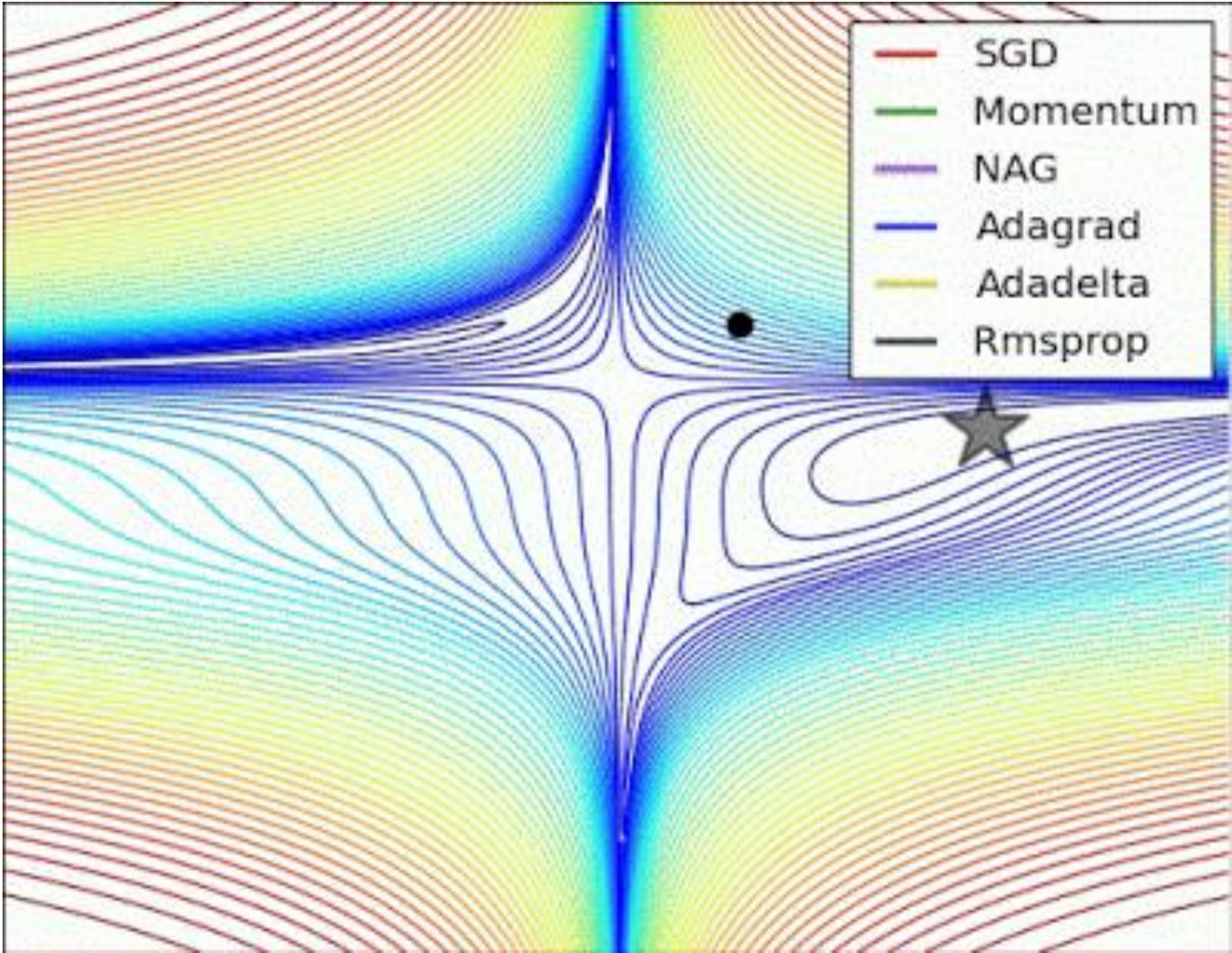
Example of optimization progress while training a neural network.

(Loss over mini-batches goes down over time.)

## The effects of step size (or “learning rate”)

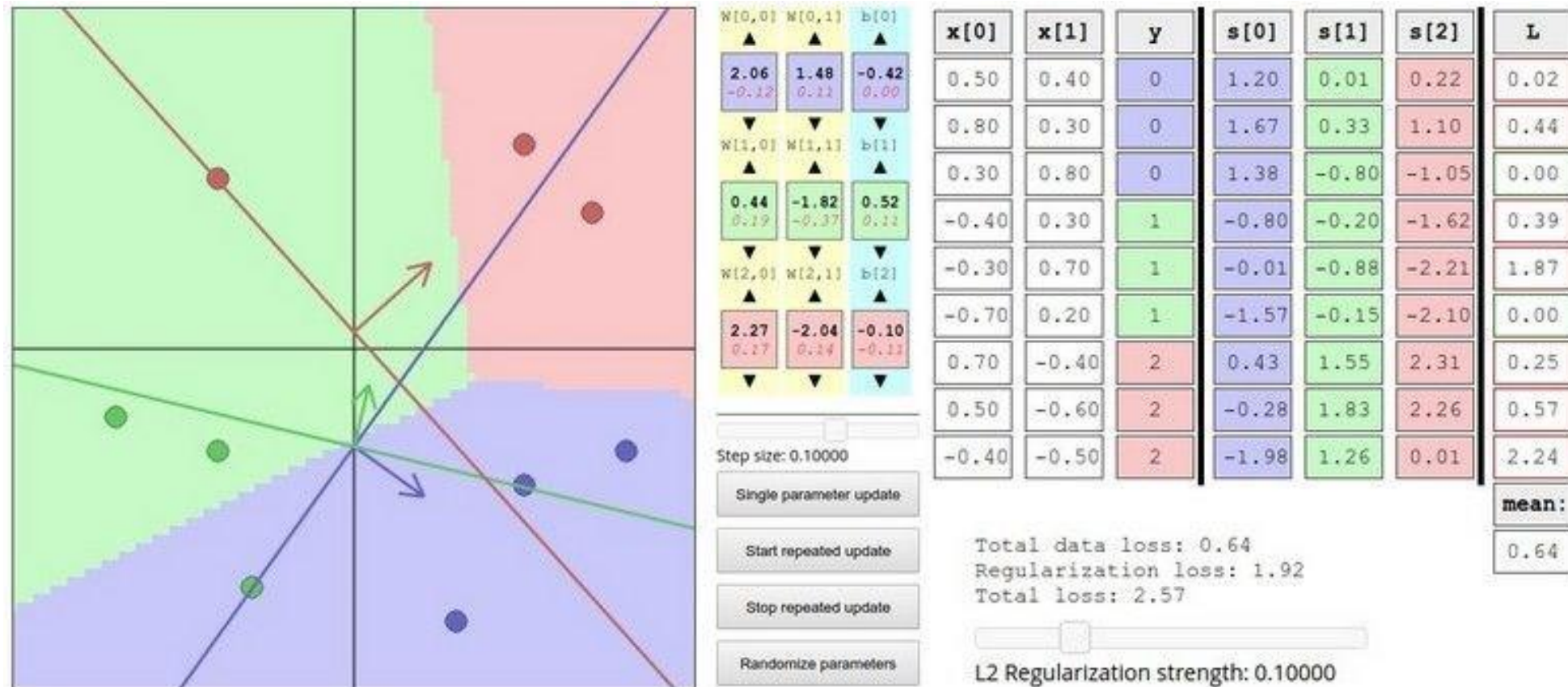


The effects of  
different  
update form  
formulas





# Interactive Web Demo time....



<http://vision.stanford.edu/teaching/cs231n-demos/linear-classify/>

# Image Features

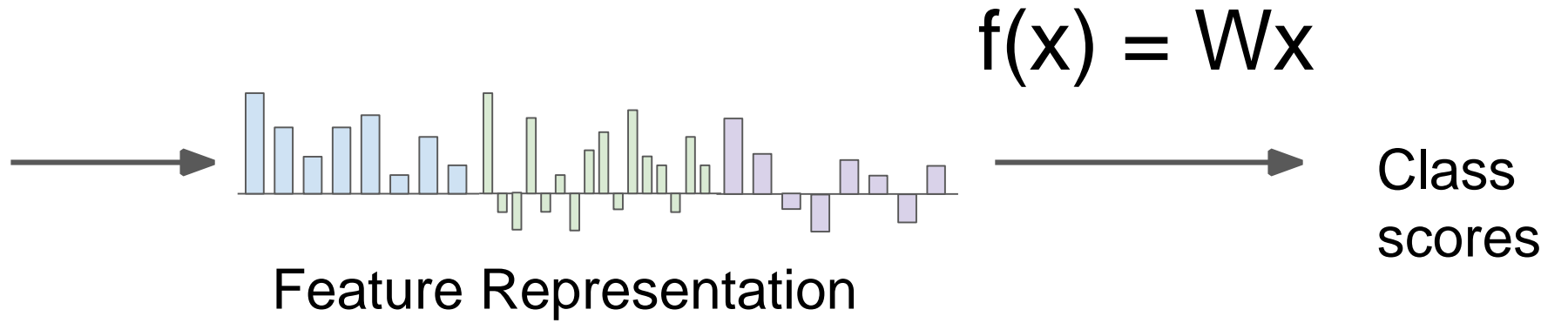


$$f(x) = Wx$$

Class  
scores

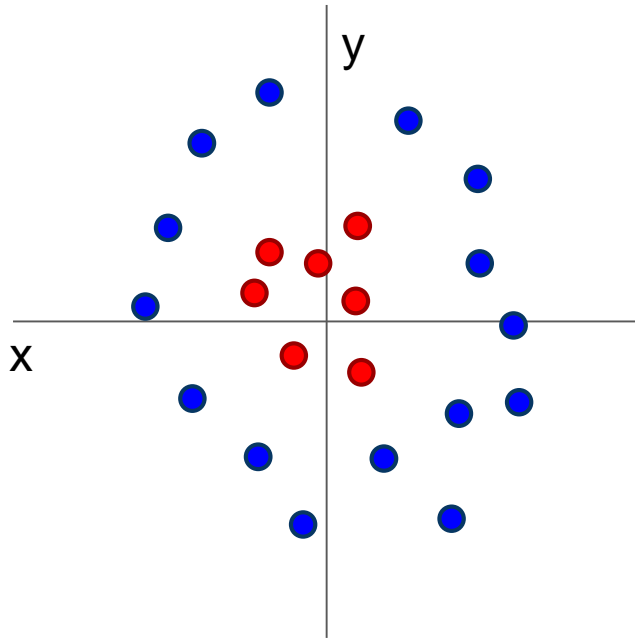


# Image Features



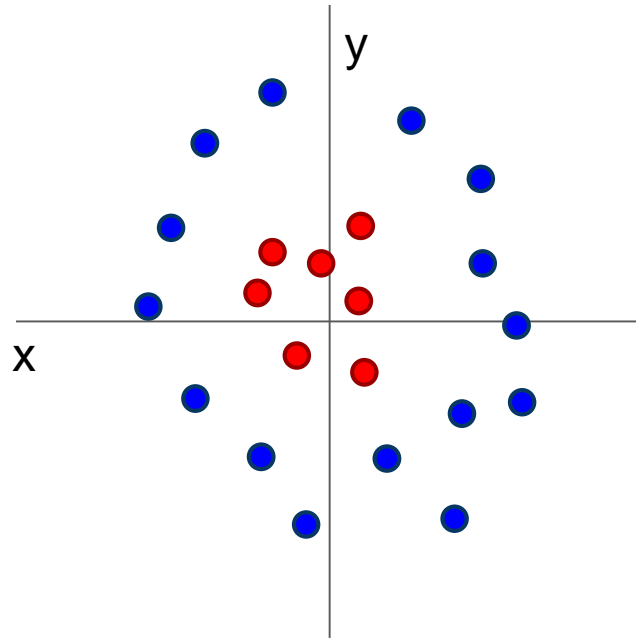


# Image Features: Motivation




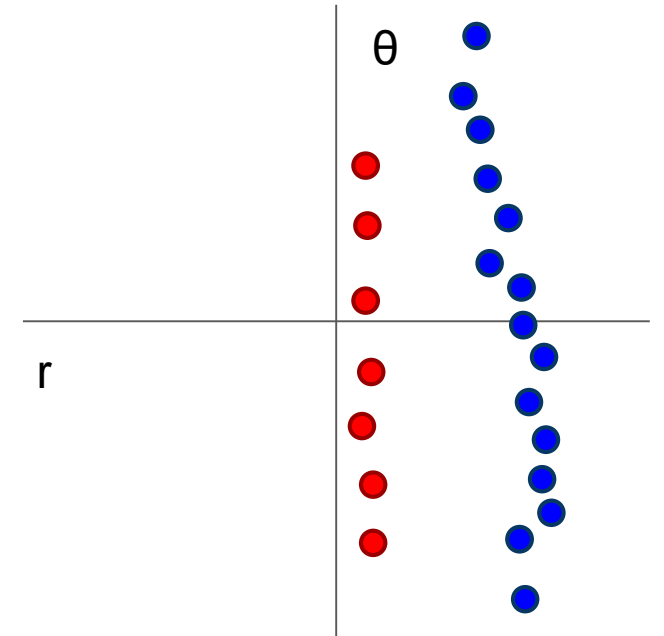
Cannot separate red  
and blue points with  
linear classifier

# Image Features: Motivation



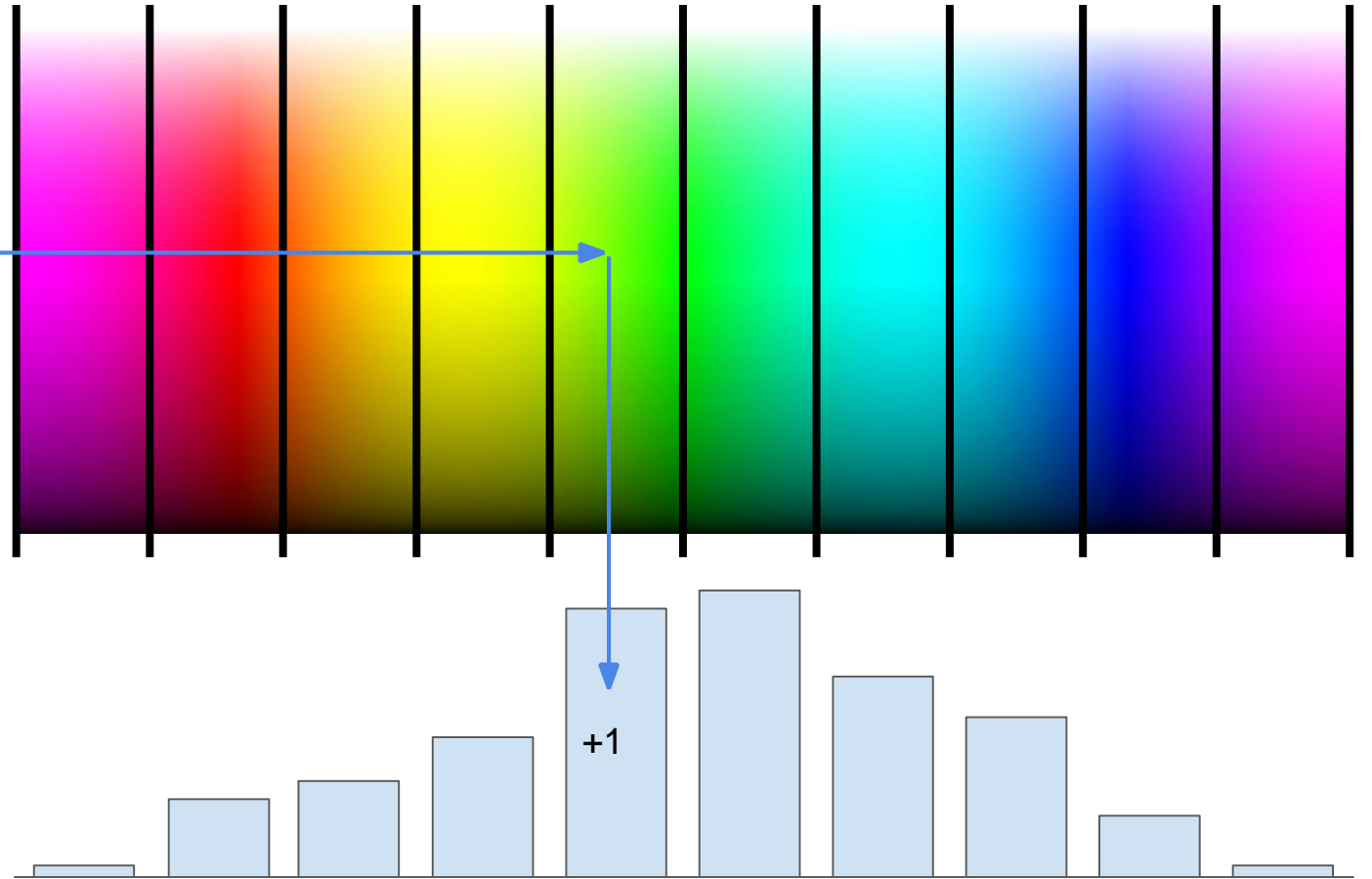
Cannot separate red  
and blue points with  
linear classifier

$$f(x, y) = (r(x, y), \theta(x, y))$$




After applying feature  
transform, points can  
be separated by linear  
classifier

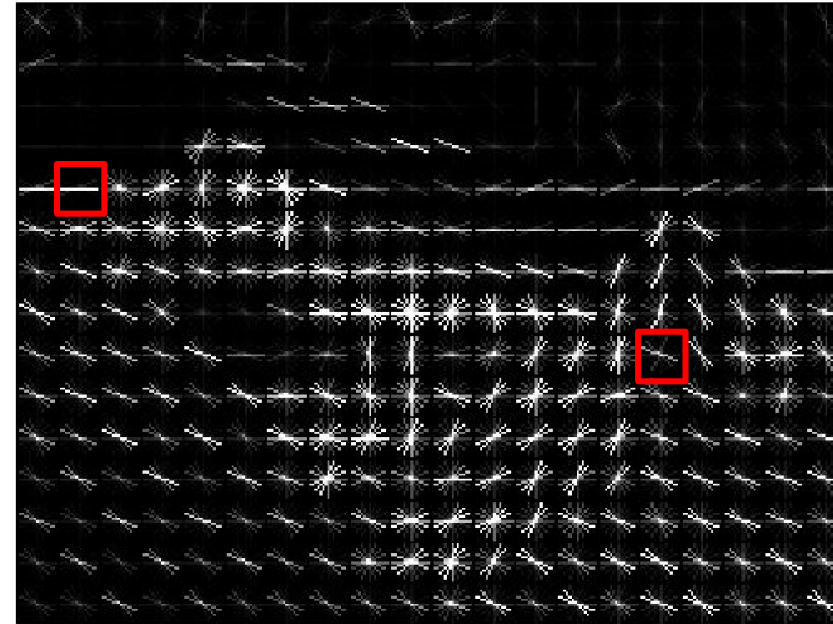
# Example: Color Histogram



# Example: Histogram of Oriented Gradients (HoG)



Divide image into 8x8 pixel regions  
Within each region quantize edge  
direction into 9 bins



Example: 320x240 image gets divided  
into 40x30 cells; in each cell there are  
9 numbers so feature vector has  
 $30 \times 40 \times 9 = 10,800$  numbers

Lowe, "Object recognition from local scale-invariant features", ICCV 1999

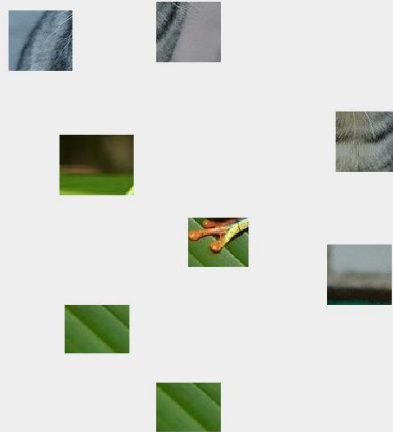
Dalal and Triggs, "Histograms of oriented gradients for human detection," CVPR 2005

# Example: Bag of Words

## Step 1: Build codebook



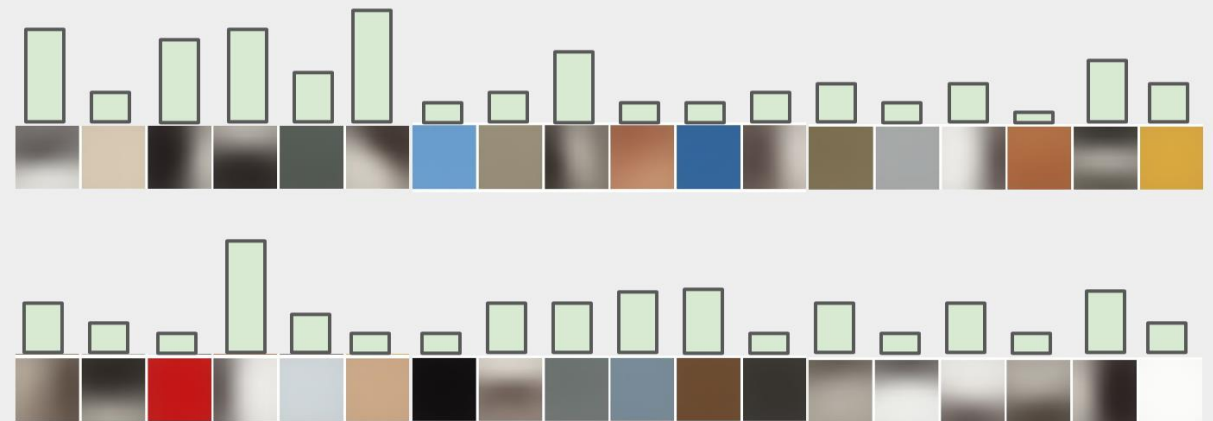
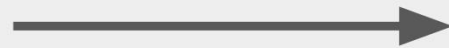
Extract random patches



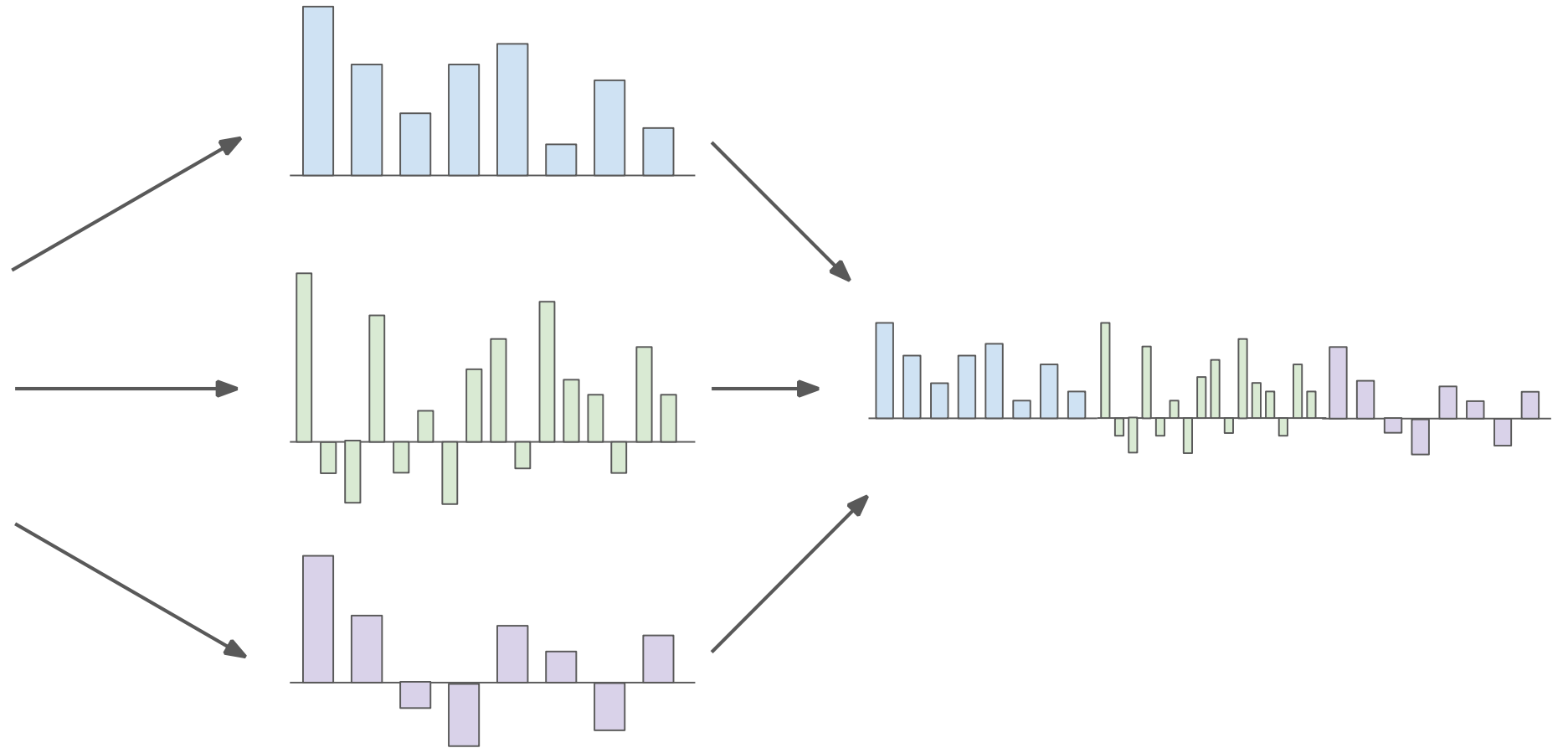
Cluster patches to form "codebook" of "visual words"



## Step 2: Encode images

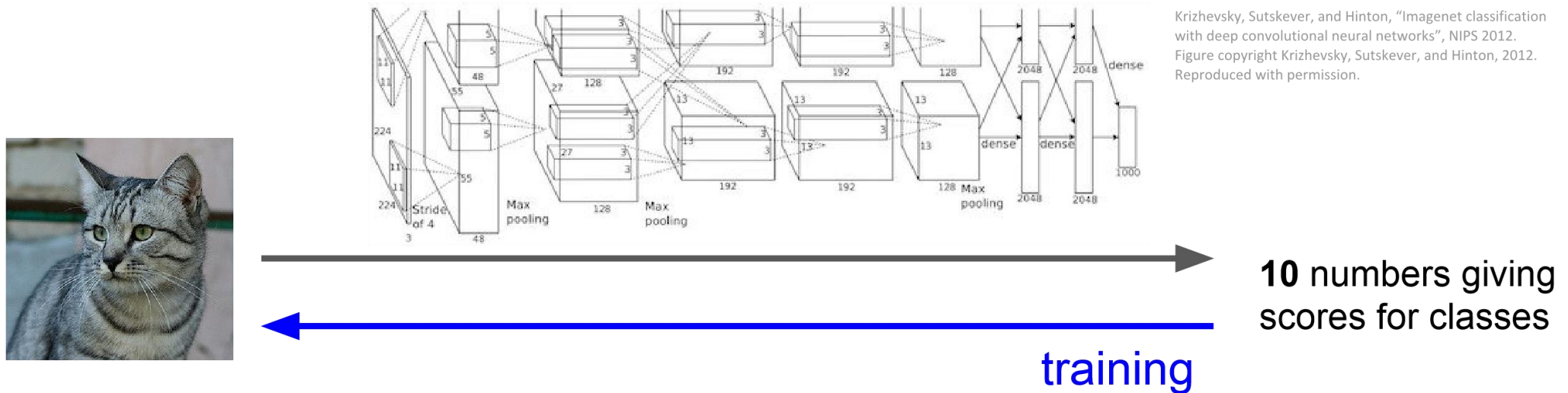
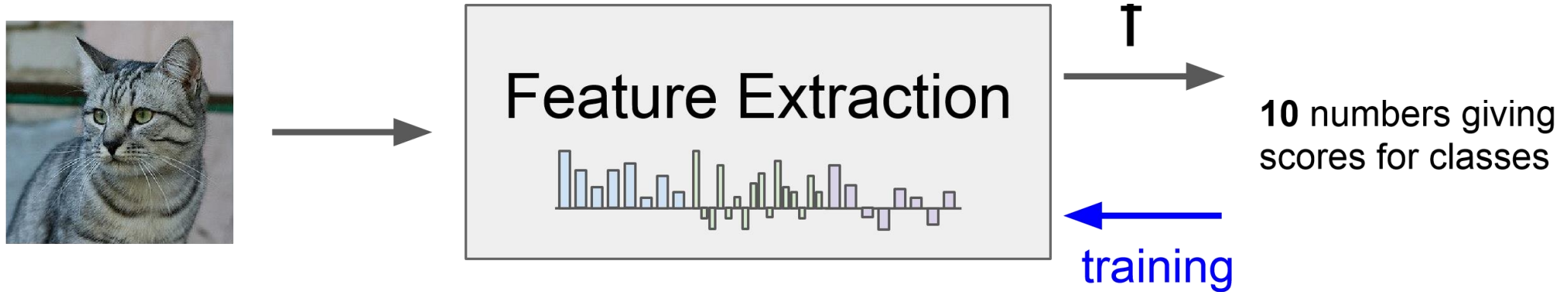


# Image Features





# Image features vs ConvNets





# В следующий раз

- Введение в нейронные сети
- Backpropagation