Глубокое обучение
в компьютерном зрении

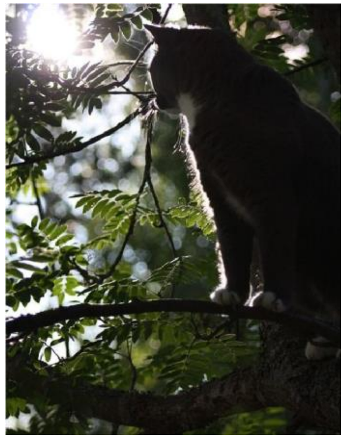
Занятие 2
Функция ошибки и Оптимизация.

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

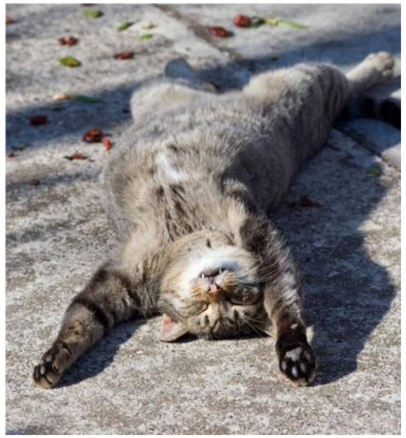
e-mail: yashuninda@yandex.ru

На прошлом занятии: **Сложности распознавания**

Освещение



Деформации



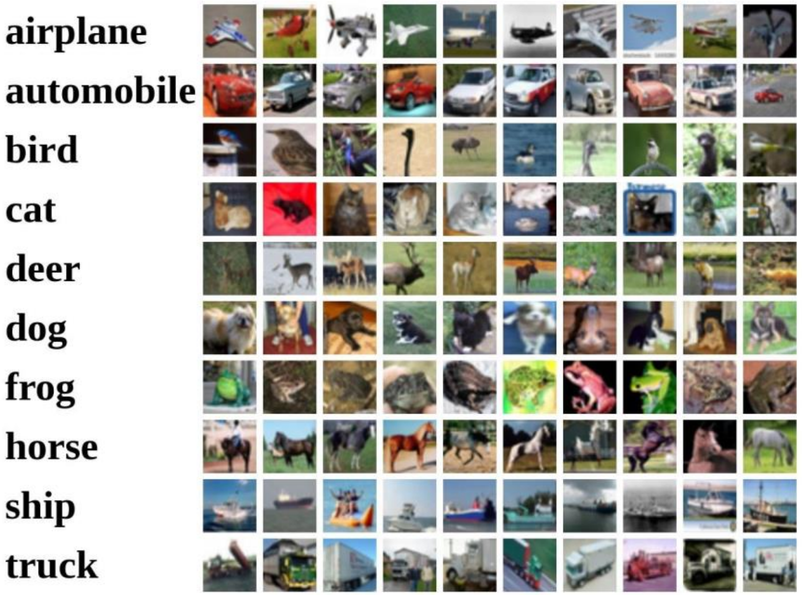
Заслонение

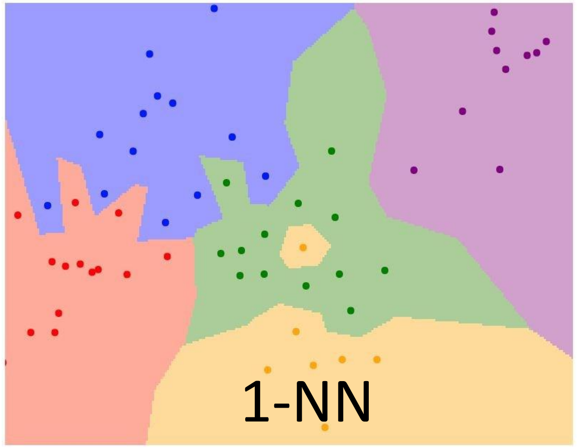
Фон

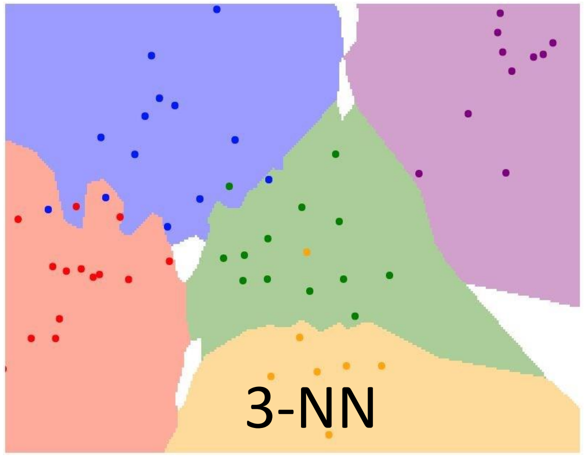
Вариативность классов

На прошлом занятии: **Классификация на основе данных, kNN**

CIFAR10

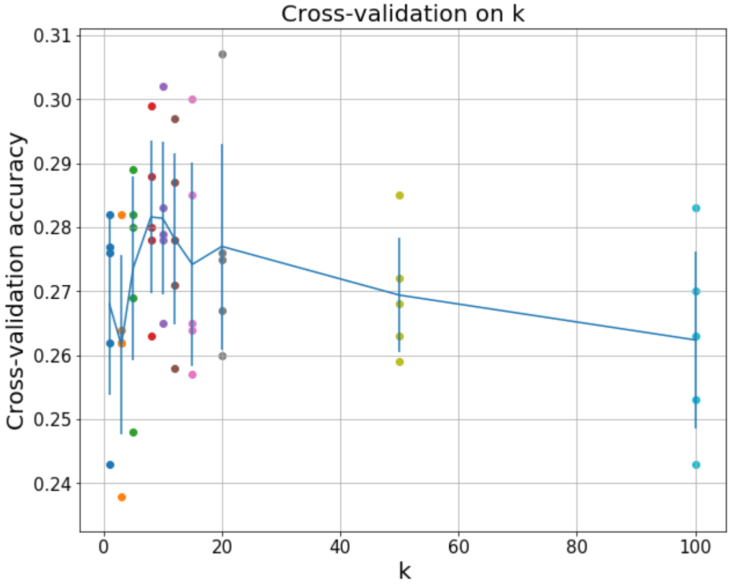






Cross-validation

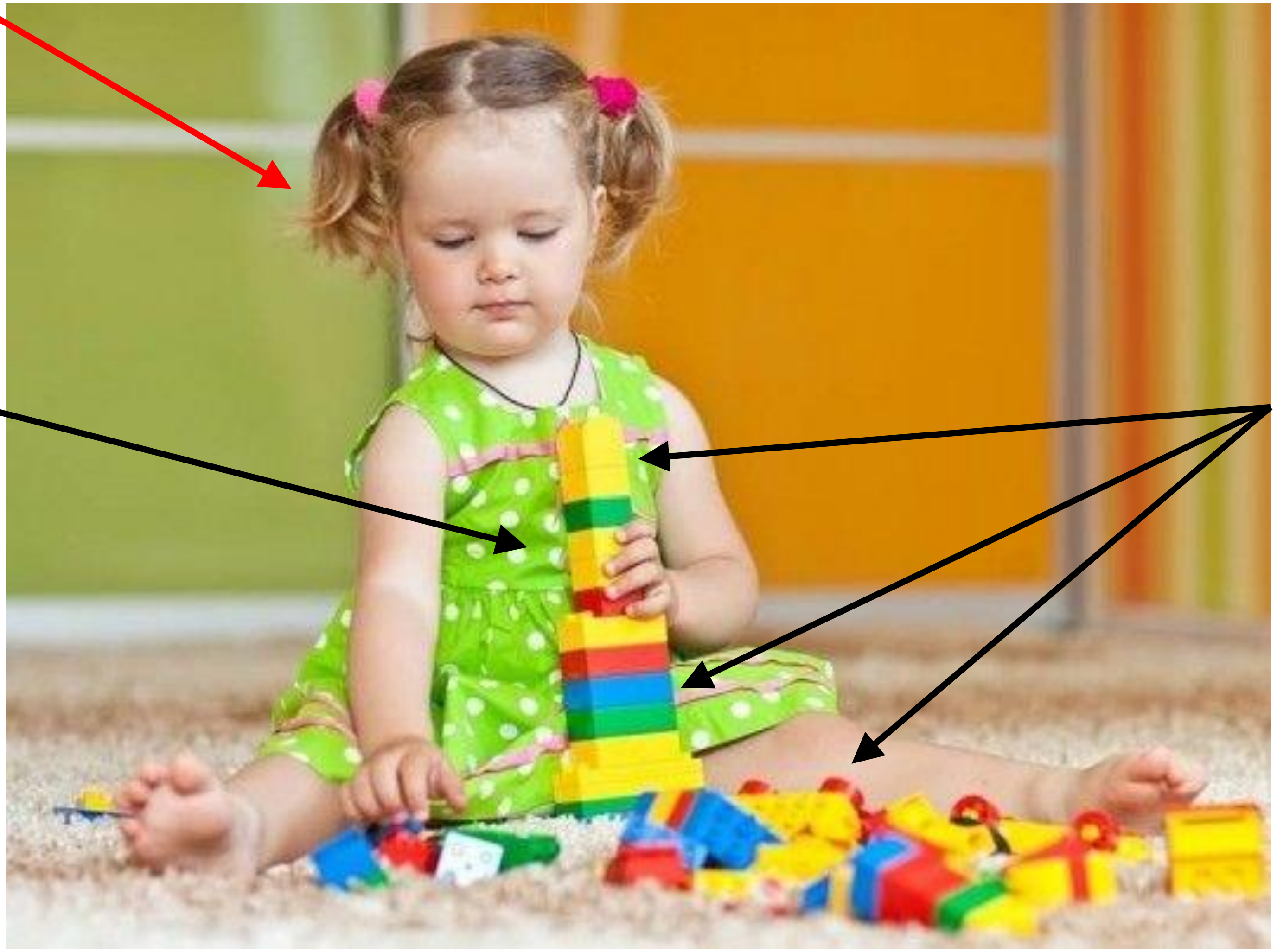
fold 1	fold 2	fold 3	fold 4	test
fold 1	fold 2	fold 3	fold 4	test



В прошлый раз: **Линейный классификатор**

Neural Network practitioner

Neural Network



Linear
classifiers

В прошлый раз: **Линейный классификатор**

Image

$$s = \boxed{f(x,W)}_{10\times1} = \boxed{W}_{10\times3072}\boxed{x}^{3072\times1} + \boxed{b}_{10\times1}$$

s – scores
W – weights or parameters
x – image pixels
b – bias

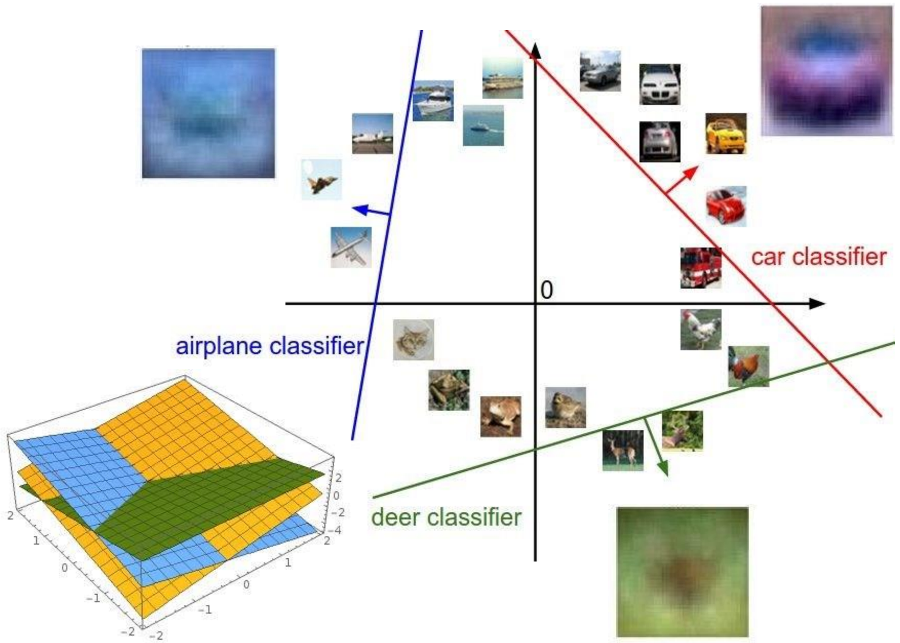
Array of **32x32x3** numbers
(3072 numbers total)

CIFAR-10
50,000 training images
10,000 testing images
10 classes

В прошлый раз: **Интерпретация линейного классификатора**

CIFAR-10

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



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

В прошлый раз: **Линейный классификатор**



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

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

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

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

Toy example

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

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	-**3.1**

Функция ошибки (Loss function)

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



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	-**3.1**

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

x_i - image
y_i - label, element of a set {0, 1, …}

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	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

x_i - image
y_i - label, element of a set {0, 1, …}

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

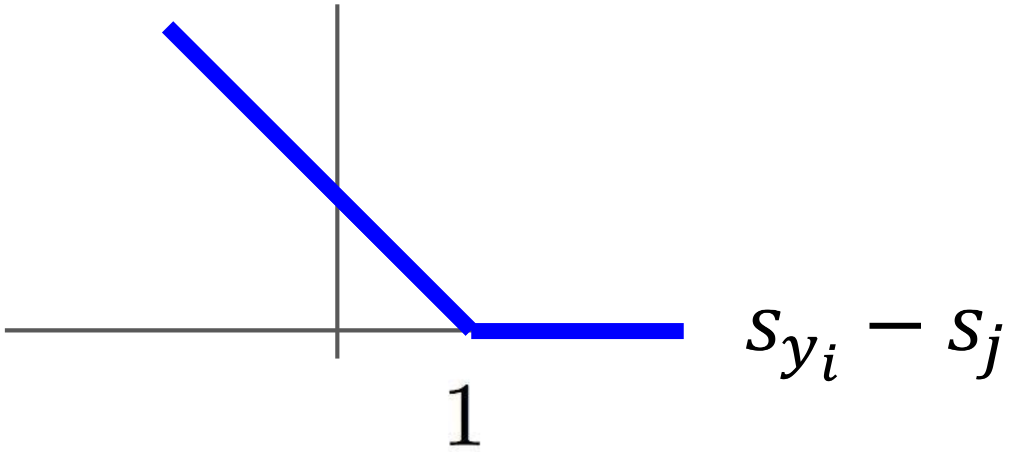
$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$

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

Multiclass SVM (hinge) loss

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

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**



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	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9		

x_i - image

y_i - label, element of a set {0, 1, …}

scores vector

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

s_{y_i} element corresponds to ground truth label y_i

SVM loss

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

= 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

Multiclass SVM (hinge) loss

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



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	

x_i - image
y_i - label, element of a set {0, 1, ...}

scores vector

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

s_{y_i} element corresponds
to ground truth label y_i

SVM loss

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

= 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	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	12.9

x_i - image
y_i - label, element of a set {0, 1, …}

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

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

= 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	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	12.9

x_i - image
y_i - label, element of a set {0, 1, …}

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

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

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	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	12.9

x_i - image
y_i - label, element of a set {0, 1, ...}

scores vector

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

s_{y_i} element corresponds
to ground truth label y_i

SVM loss

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

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	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	12.9

x_i - image
y_i - label, element of a set {0, 1, …}

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

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

Q2: what is the
min/max possible
loss?

Multiclass SVM (hinge) loss

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

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	-**3.1**
Loss	2.9	0	12.9

x_i - image
y_i - label, element of a set {0, 1, …}

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

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

Q3: At initialization W
is small so all s ≈ 0.
What is the loss?

Multiclass SVM (hinge) loss

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

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	12.9

x_i - image
y_i - label, element of a set {0, 1, …}

scores vector

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

s_{y_i} element corresponds
to ground truth label y_i

SVM loss

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

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	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	12.9

x_i - image
y_i - label, element of a set {0, 1, ...}

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

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

Q5: What if we used mean instead of sum?

Multiclass SVM (hinge) loss

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



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	12.9

x_i - image
y_i - label, element of a set {0, 1, …}

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

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

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	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	

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

Before:

= 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

With 2W:

= max(0, 2.6 - 9.8 + 1)
+max(0, 4.0 - 9.8 + 1)
= max(0, -6.2) + max(0, -4.8)
= 0 + 0

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

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

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

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

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

λ - regularization strength (hyperparameter)

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

Data loss: Model predictions should match training data

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

Регуляризация

λ - regularization strength (hyperparameter)

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

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

λ - regularization strength (hyperparameter)

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

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

λ - regularization strength (hyperparameter)

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

Data loss: Model predictions should match training data

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

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

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

$$w_1 = [1, 0, 0, 0]$$
$$w_2 = [0.25, 0.25, 0.25, 0.25]$$

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

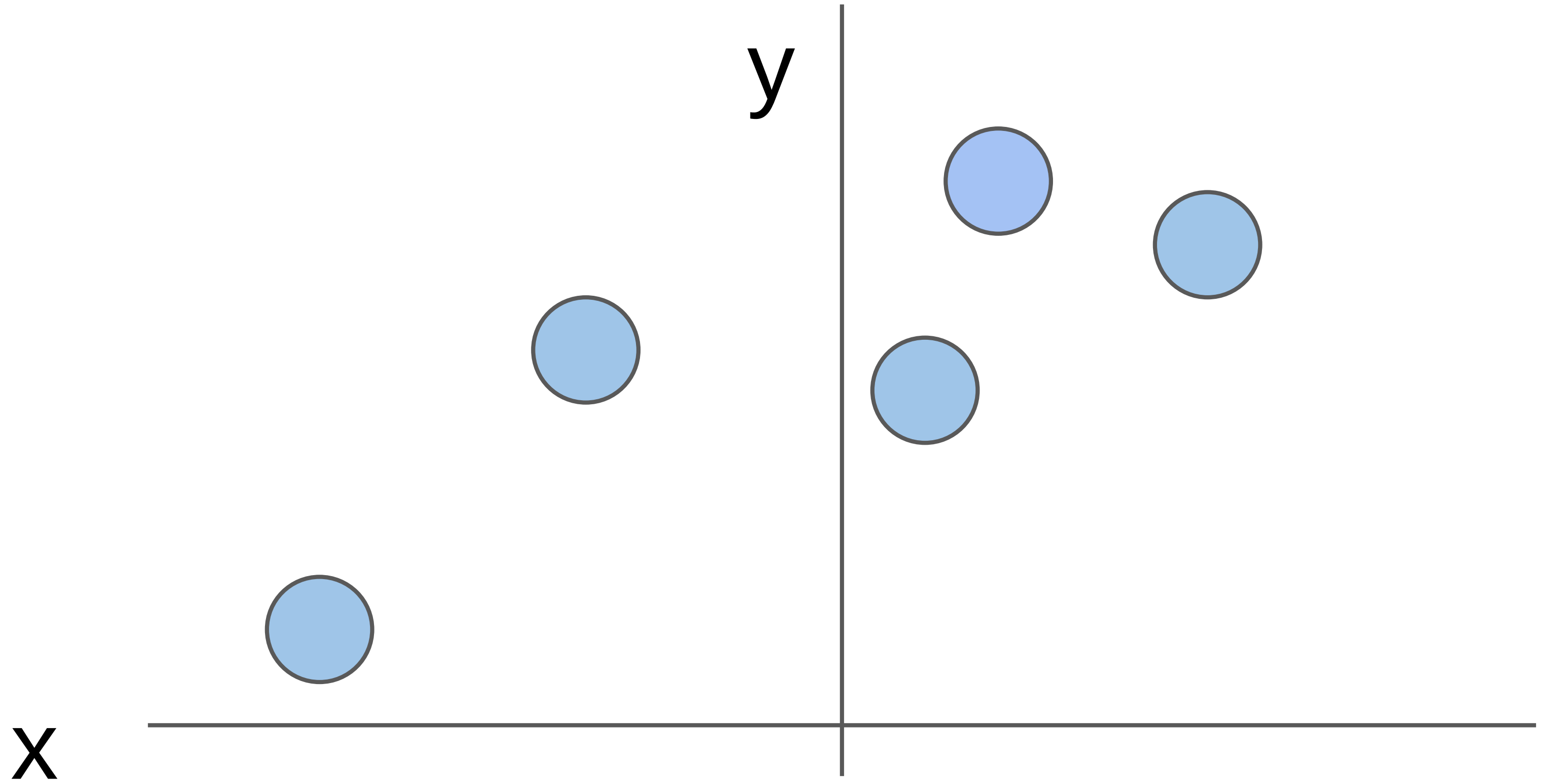
L2 regularization

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

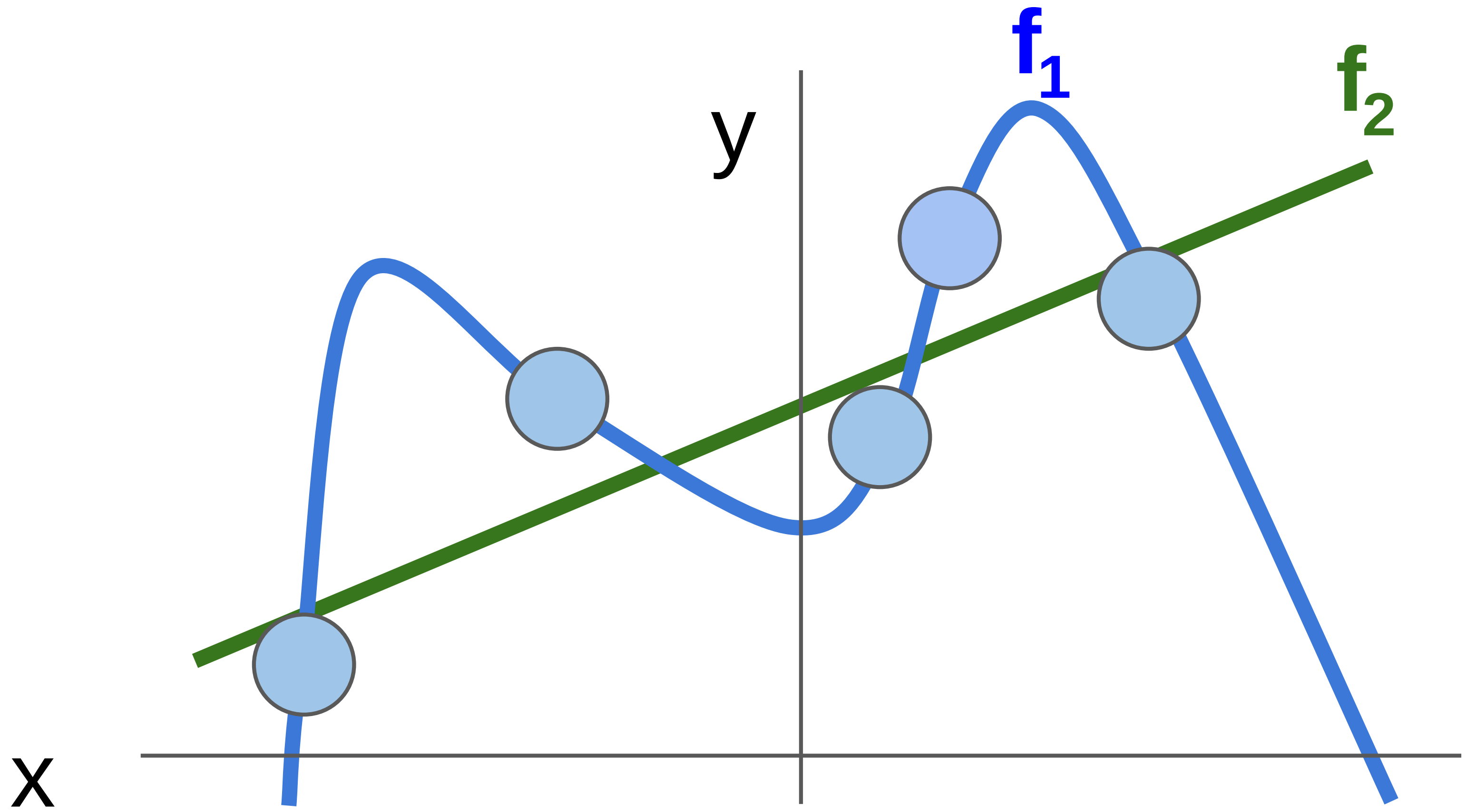
L2 регуляризация «размазывает» веса

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

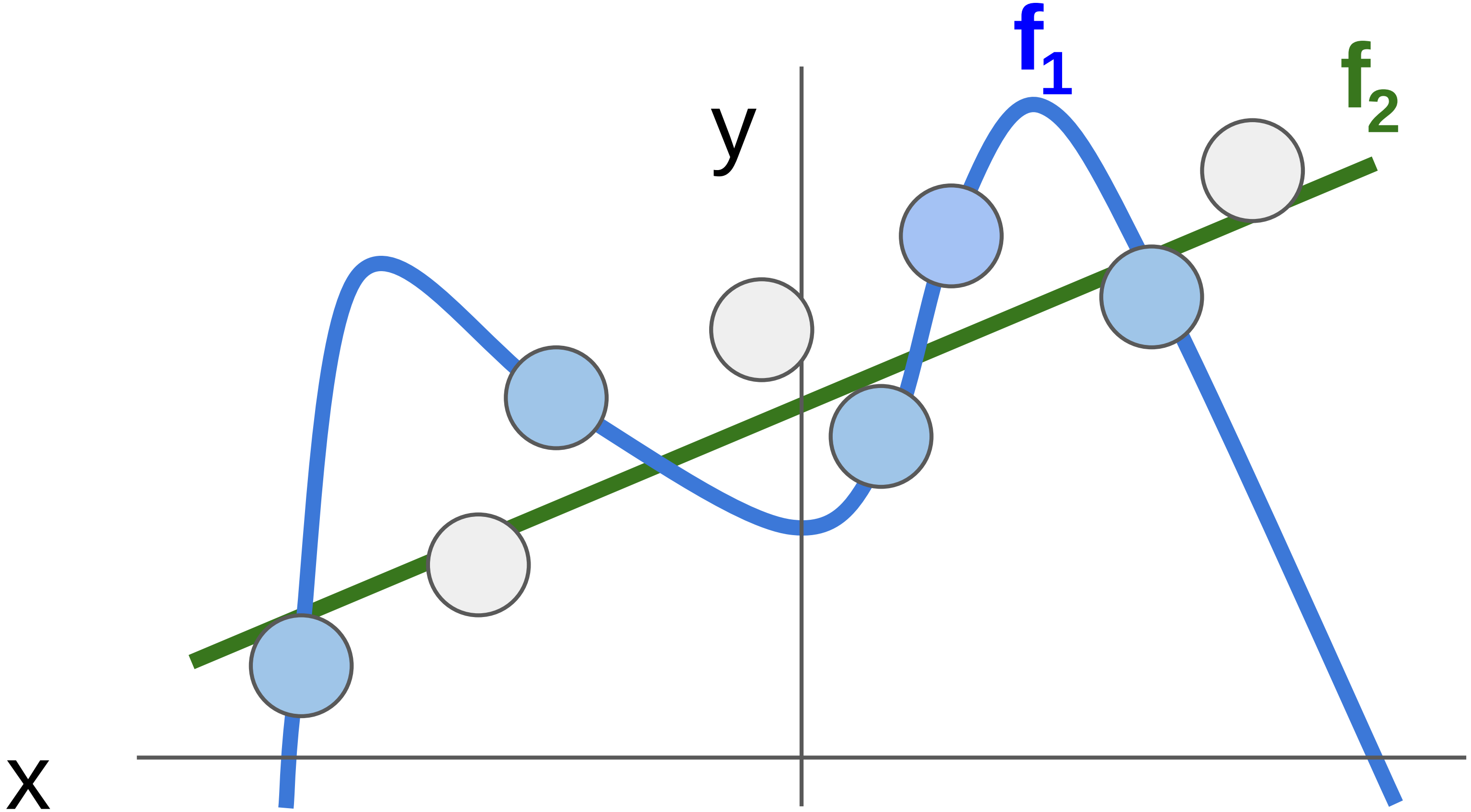
Регуляризация: уменьшение переобучения



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 $$\boxed{\max_W \prod_i P(Y = y_i|X = x_i, W) \Rightarrow \max_W \sum_i \log P(Y = y_i|X = x_i, W)}$$

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

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

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

http://bjlkeng.github.io/posts/probabilistic-interpretation-of-regularization/

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

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

Задача найти наиболее вероятные параметры W, исходя из тренировочных данных

http://bjlkeng.github.io/posts/probabilistic-interpretation-of-regularization/

Regularization: **Maximum A Posteriori probability estimate (MAP)**

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

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

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

http://bjlkeng.github.io/posts/probabilistic-interpretation-of-regularization/

Regularization: **Maximum A Posteriori probability estimate (MAP)**

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

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

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

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

http://bjlkeng.github.io/posts/probabilistic-interpretation-of-regularization/

Regularization: **Maximum A Posteriori probability estimate (MAP)**

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

Выбор априорных распределений параметров W

- Распределение Гаусса

$$P(W|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(W-\mu)^2}{2\sigma^2}} \Rightarrow \text{L2 regularization} \quad R(W) = \sum_j W_j^2$$

- Распределение Лапласа

$$P(W|\mu,b) = \frac{1}{2b} \exp\left(-\frac{|W-\mu|}{b}\right) \Rightarrow \text{L1 regularization} \quad R(W) = \sum_j |W_j|$$

http://bjlkeng.github.io/posts/probabilistic-interpretation-of-regularization/

Softmax Classifier (Multinomial Logistic Regression)

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

cat	**3.2**
car	5.1
frog	-1.7

Softmax Classifier (Multinomial Logistic Regression)

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

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

cat	**3.2**
car	5.1
frog	-1.7

Softmax Classifier (Multinomial Logistic Regression)

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

$$s = f(x_i, W) \qquad P(Y = y_i | X = x_i) = \boxed{\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}}$$

probability of x_i image has y_i label

Softmax function

cat	**3.2**
car	5.1
frog	-1.7

Softmax Classifier (Multinomial Logistic Regression)

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

$s = f(x_i, W) \qquad P(Y = y_i | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$

probability of x_i image has y_i label

Probabilities must be >= 0

cat	**3.2**	$\xrightarrow{\text{exp}}$	**24.5**
car	5.1		164.0
frog	-1.7		0.18

unnormalized probabilities

Softmax Classifier (Multinomial Logistic Regression)

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

$s = f(x_i, W) \qquad P(Y = y_i | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$

probability of x_i image has y_i label

Probabilities must be >= 0

Probabilities must sum to 1

cat	**3.2**	**24.5**	**0.13**
car	5.1	164.0	0.87
frog	-1.7	0.18	0.00

exp → normalize →

unnormalized probabilities

probabilities

Softmax Classifier (Multinomial Logistic Regression)

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

$$s = f(x_i, W) \qquad P(Y = y_i | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$$

probability of x_i image has y_i label

Probabilities must be >= 0

Probabilities must sum to 1

		exp		normalize	
cat	**3.2**		**24.5**		**0.13**
car	5.1		164.0		0.87
frog	-1.7		0.18		0.00

unnormalized log probabilities or logits

unnormalized probabilities

probabilities

Softmax Classifier (Multinomial Logistic Regression)

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

$$s = f(x_i, W) \qquad P(Y = y_i | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$$

probability of x_i image has y_i label

Probabilities must be >= 0

Probabilities must sum to 1

	unnormalized log probabilities or logits		unnormalized probabilities		probabilities
cat	**3.2**	exp →	**24.5**	normalize →	**0.13**
car	5.1		164.0		0.87
frog	-1.7		0.18		0.00

How to calculate loss?

Softmax Classifier: **Maximum likelihood estimation**

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

$s = f(x_i, W)$ $P(Y = y_i | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$

probability of x_i image has y_i label

Probabilities must be >= 0

Probabilities must sum to 1

	unnormalized log probabilities or logits	exp →	unnormalized probabilities	normalize →	probabilities
cat	**3.2**		**24.5**		**0.13**
car	5.1		164.0		0.87
frog	-1.7		0.18		0.00

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

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

Softmax Classifier: **Maximum likelihood estimation**

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

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

Probabilities must be >= 0

Probabilities must sum to 1

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

	unnormalized log probabilities or logits		unnormalized probabilities		probabilities
cat	**3.2**	exp	**24.5**	normalize	**0.13**
car	5.1		164.0		0.87
frog	-1.7		0.18		0.00

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

Softmax Classifier: **Cross-entropy loss**

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

$$s = f(x_i, W) \qquad P(Y = y_i | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$$

probability of x_i image has y_i label

Probabilities must be >= 0

Probabilities must sum to 1

	unnormalized log probabilities or logits	exp →	unnormalized probabilities	normalize →	probabilities	→ How to compare? ←	Correct probabilities
cat	**3.2**		**24.5**		**0.13**		**1.0**
car	5.1		164.0		0.87		0.0
frog	-1.7		0.18		0.00		0.0

Softmax Classifier: **Cross-entropy loss**

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

$s = f(x_i, W) \qquad P(Y = y_i | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$

probability of x_i image has y_i label

Probabilities must be >= 0

Probabilities must sum to 1

	unnormalized log probabilities or logits		unnormalized probabilities		probabilities	Correct probabilities
cat	**3.2**	exp →	**24.5**	normalize →	**0.13**	**1.0**
car	5.1		164.0		0.87	0.0
frog	-1.7		0.18		0.00	0.0

Kullback–Leibler divergence

$D_{KL}(\boldsymbol{P}||\boldsymbol{Q}) = \sum_y P(y) \log \frac{P(y)}{Q(y)}$

Softmax Classifier: **Cross-entropy loss**

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

$$s = f(x_i, W) \qquad P(Y = y_i | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$$

probability of x_i image has y_i label

	unnormalized log probabilities or logits		unnormalized probabilities (Probabilities must be >= 0)		probabilities (Probabilities must sum to 1)	Correct probabilities
cat	**3.2**	exp →	**24.5**	normalize →	**0.13**	**1.0**
car	5.1		164.0		0.87	0.0
frog	-1.7		0.18		0.00	0.0

Cross Entropy

$$H(\boldsymbol{P}, \boldsymbol{Q}) = H(P) + D_{KL}(P||Q) = -\sum_y \boldsymbol{P}(y) \log \boldsymbol{Q}(y)$$

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

Softmax Classifier (Multinomial Logistic Regression)

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

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

Maximize probability of correct class

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

Putting it all together

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

cat	**3.2**
car	5.1
frog	-1.7

Softmax Classifier (Multinomial Logistic Regression)

cat	**3.2**
car	5.1
frog	-1.7

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

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

Maximize probability of correct class

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

Putting it all together

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

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

Softmax Classifier (Multinomial Logistic Regression)

cat	**3.2**
car	5.1
frog	-1.7

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

$$s = f(x_i, W) \qquad P(Y = y_i | X = x_i) = \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$$

probability of x_i image has y_i label

Maximize probability of correct class

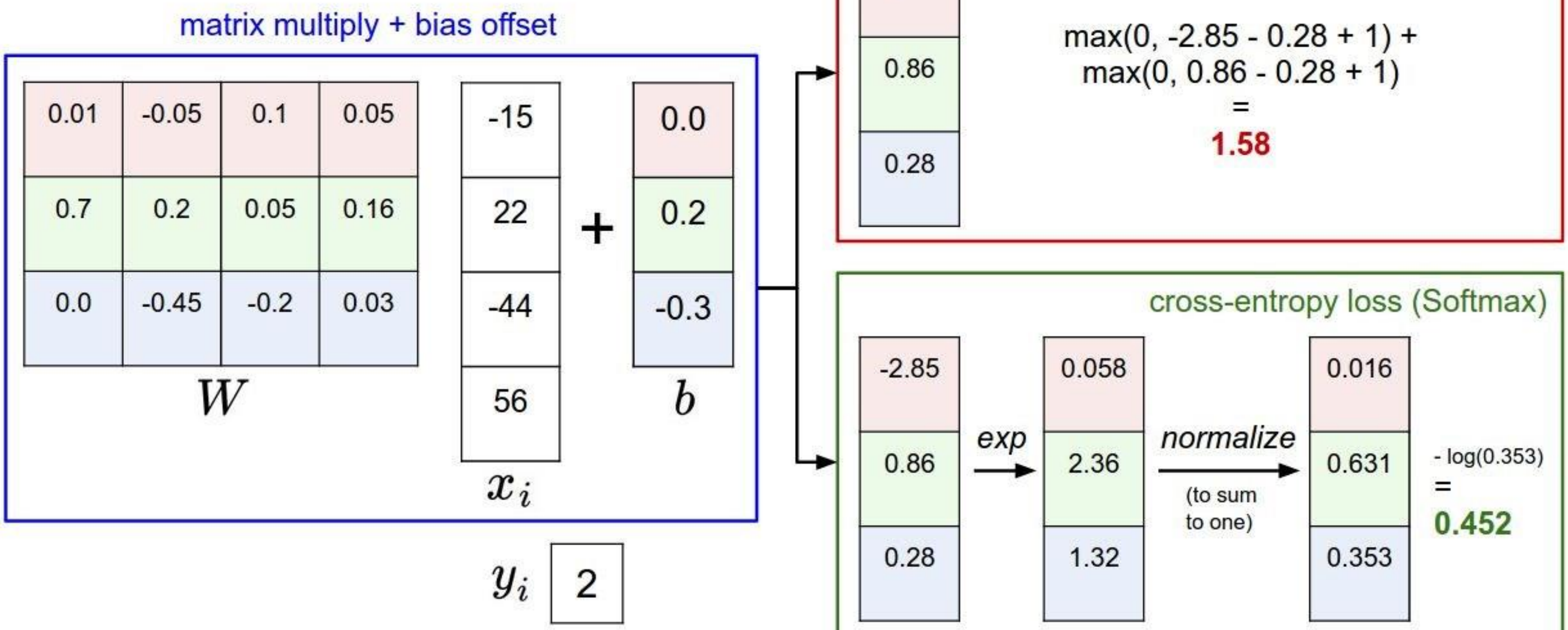
$$L_i = -\log P(Y = y_i | X = x_i)$$

Putting it all together

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

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

Softmax vs SVM



Softmax vs SVM

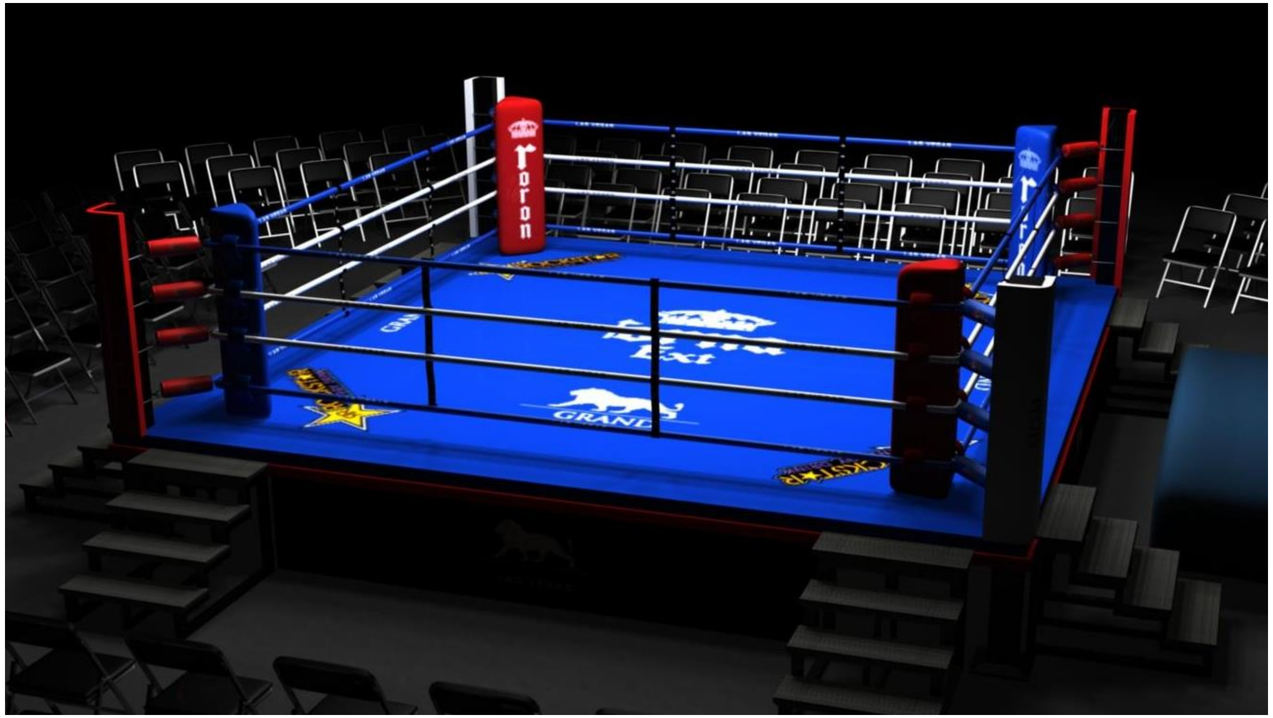
Softmax

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

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?

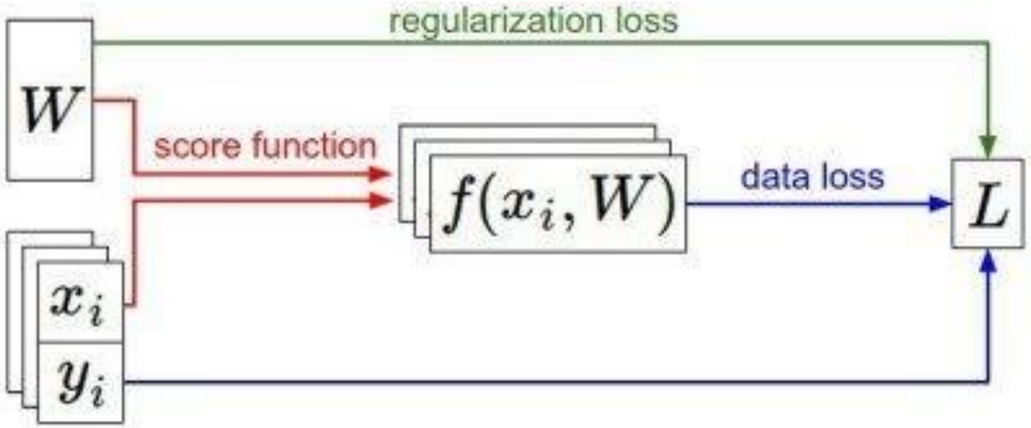
Recap

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

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

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

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

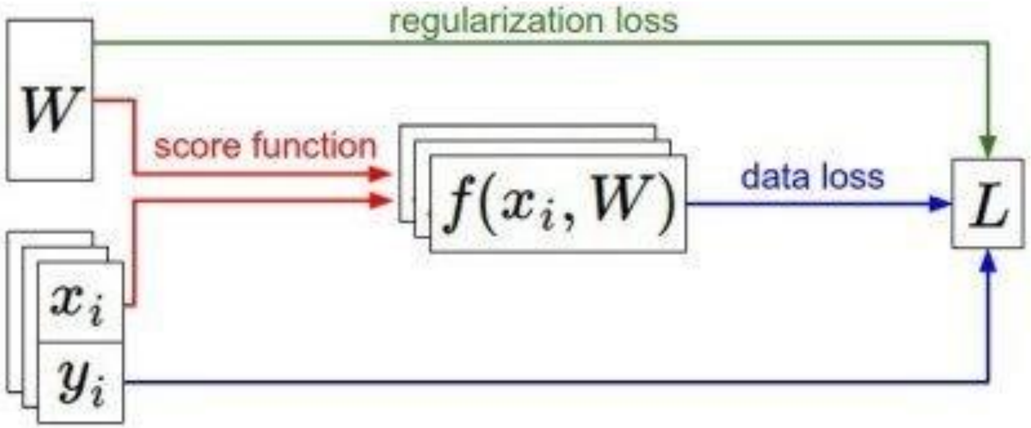
$$L_i = -\log \frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$$ Softmax

SVM

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

Full loss

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



Оптимизация

Оптимизация

Оптимизация: **случайный поиск**

Оптимизация: **случайный поиск**

```python
# 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
# ... (trunctated: 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 ≈96%)

Оптимизация: **следуем по градиенту**

Оптимизация: **следуем по градиенту**

В одномерном случае производная скаляр

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

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:	**W + h** (first dim)**:**	**gradient dL/dW:**
[0.34,	[0.34 + **0.0001**,	[?,
-1.11,	-1.11,	?,
0.78,	0.78,	?,
0.12,	0.12,	?,
0.55,	0.55,	?,
2.81,	2.81,	?,
-3.1,	-3.1,	?,
-1.5,	-1.5,	?,
0.33,…]	0.33,…]	?,…]
loss 1.25347	**loss 1.25322**	

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

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

current W:	**W + h** (second dim)**:**	**gradient dL/dW:**
[0.34,	[0.34,	[-2.5,
-1.11,	-1.11 + **0.0001**,	?,
0.78,	0.78,	?,
0.12,	0.12,	?,
0.55,	0.55,	?,
2.81,	2.81,	?,
-3.1,	-3.1,	?,
-1.5,	-1.5,	?,
0.33,…]	0.33,…]	?,…]
loss 1.25347	**loss 1.25322**	

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

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

current W:	**W + h** (third dim):	**gradient dL/dW:**
[0.34,	[0.34,	[-2.5,
-1.11,	-1.11,	0.6,
0.78,	0.78 + **0.0001**,	?,
0.12,	0.12,	?,
0.55,	0.55,	?,
2.81,	2.81,	?,
-3.1,	-3.1,	?,
-1.5,	-1.5,	?,
0.33,…]	0.33,…]	?,…]
loss 1.25347	**loss 1.25347**	

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

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

Can we do better?

The loss is just a function of W:

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

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

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 = ...
(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

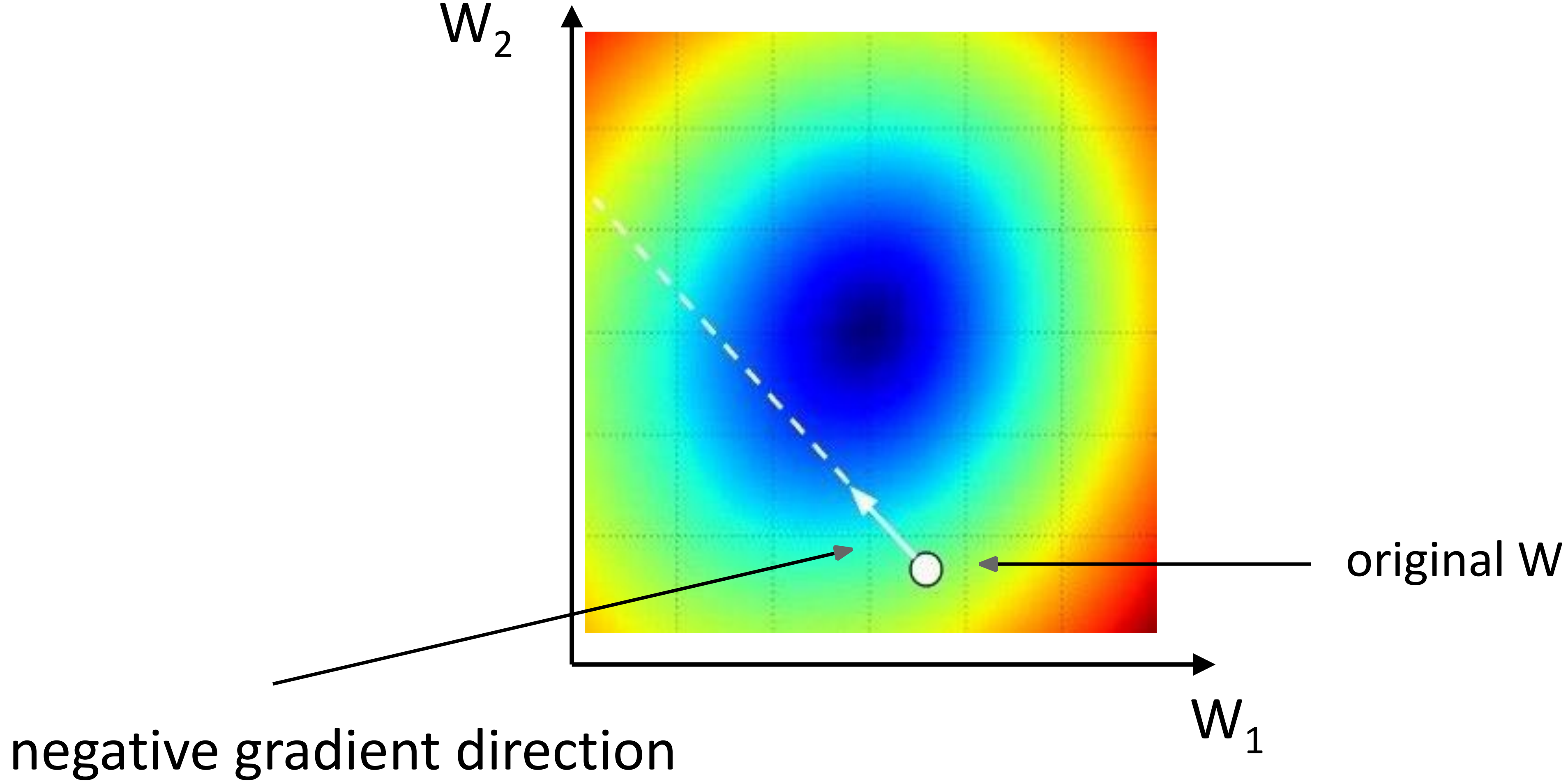
In practice: Always use analytic gradient, but check implementation with numerical gradient. This is called a **gradient check.**

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)

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

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

Full sum expensive when N is large!

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)

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

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

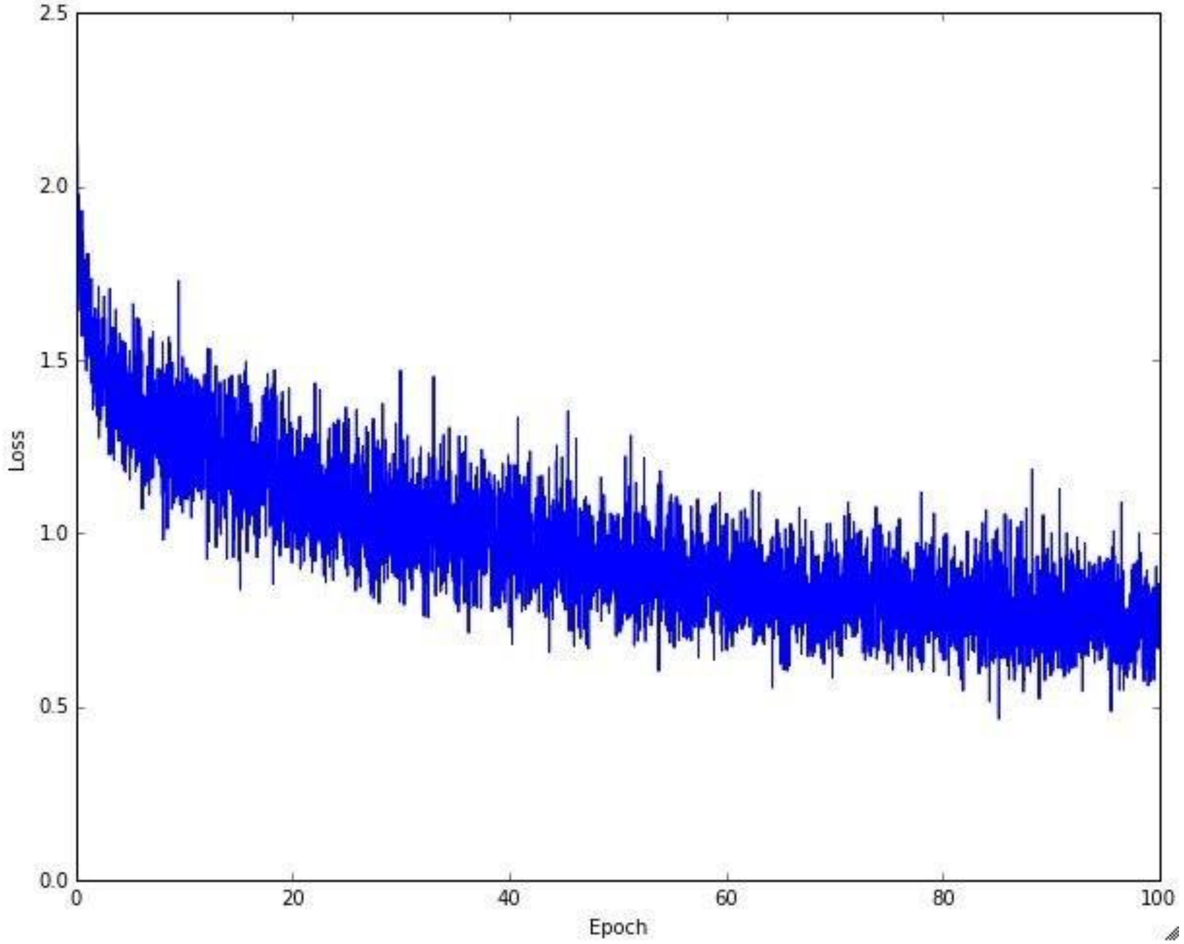
Full sum expensive when N is large!

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
```

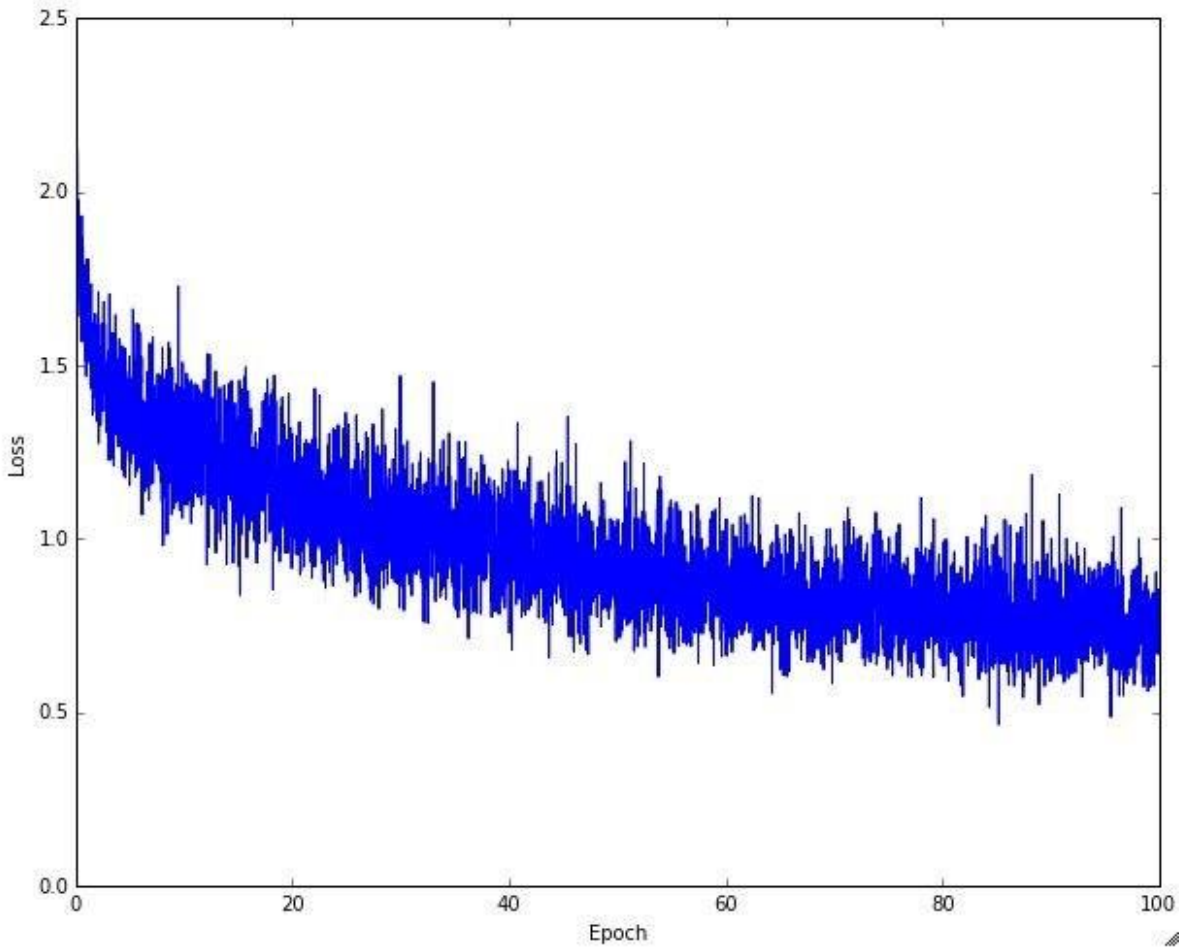
we will look at more fancy update formulas (momentum, Adagrad, RMSProp, Adam, …)

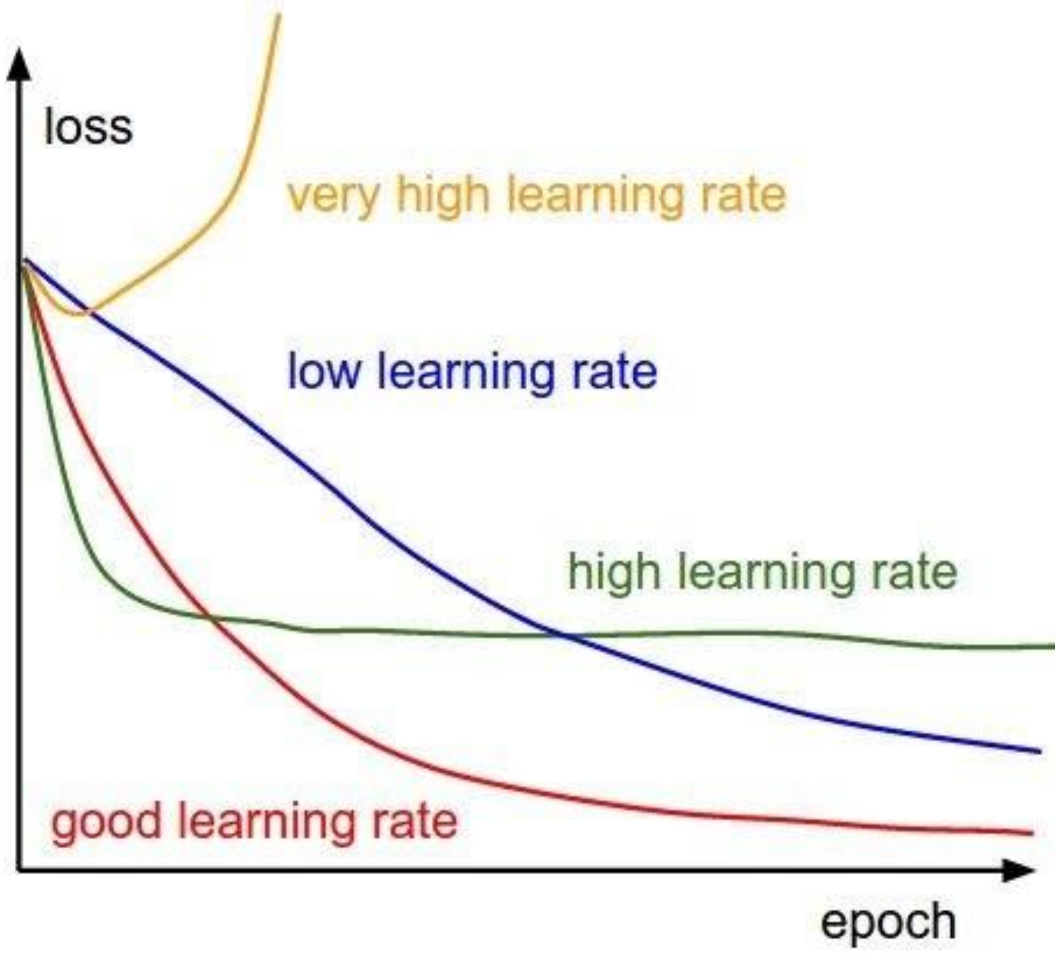


Example of optimization progress while training a neural network.

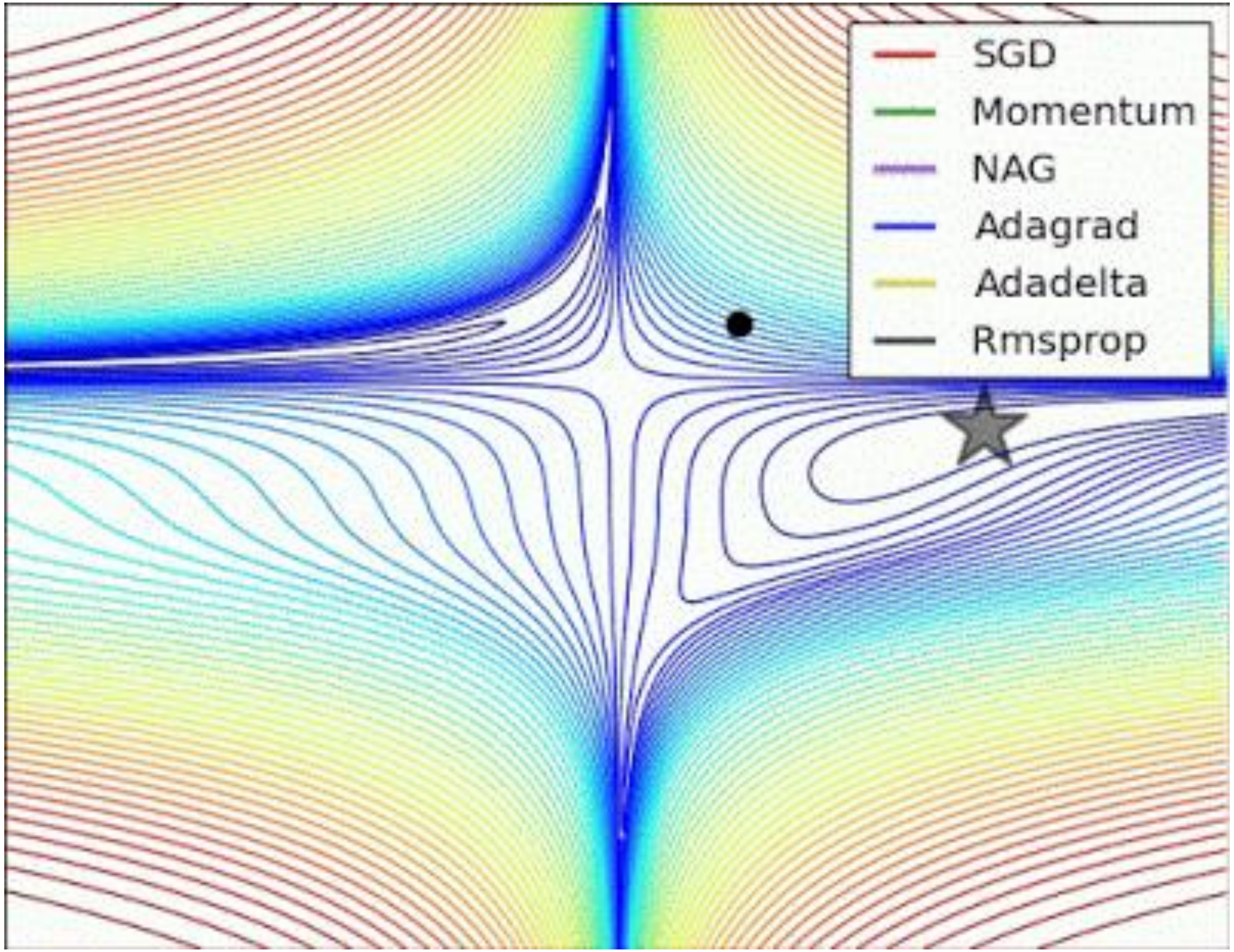
(Loss over mini-batches goes down over time.)

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

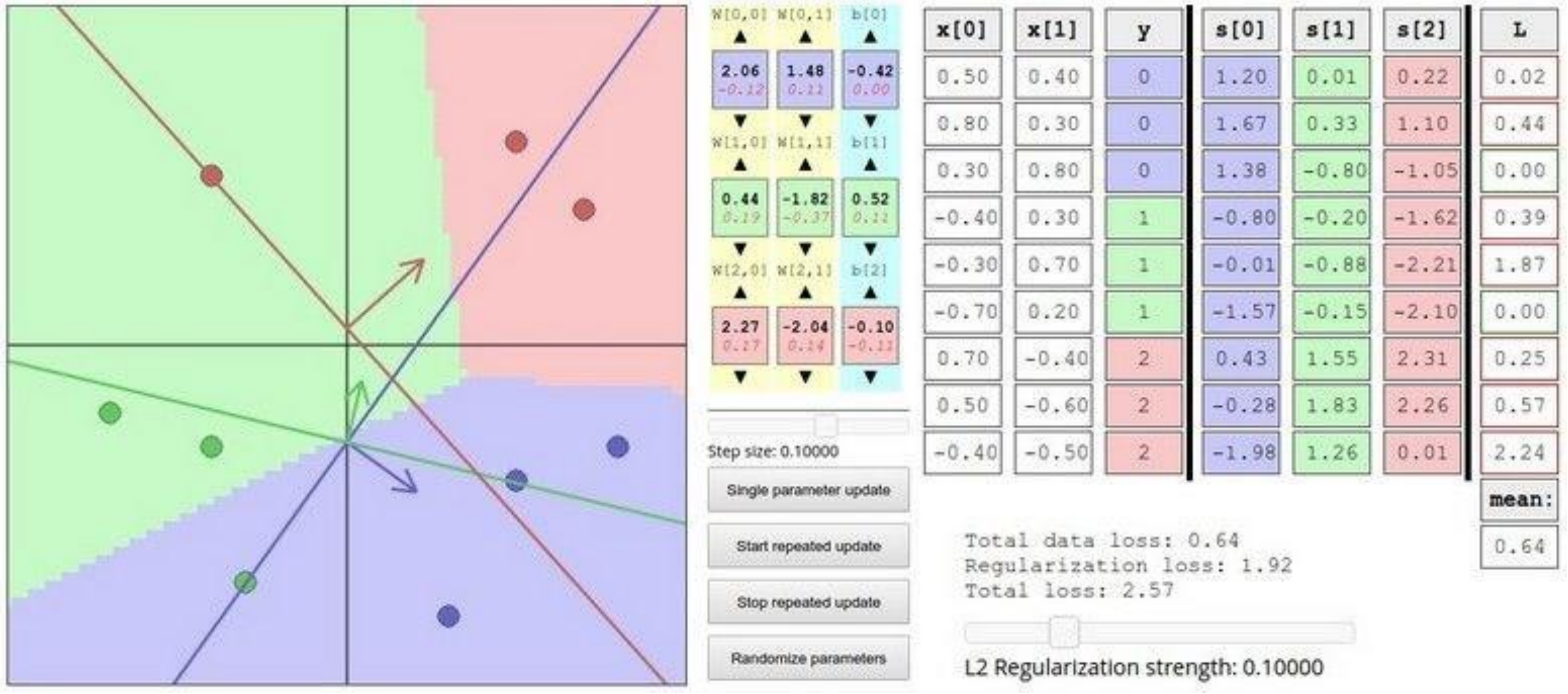




The effects of different update form formulas



Interactive Web Demo time....



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

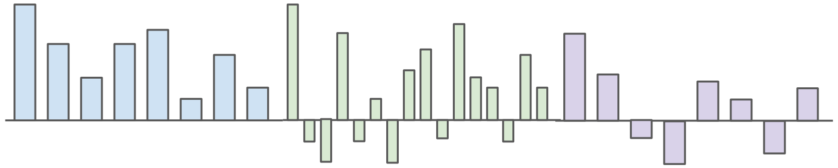
Image Features

Class scores

$$f(x) = Wx$$

Image Features

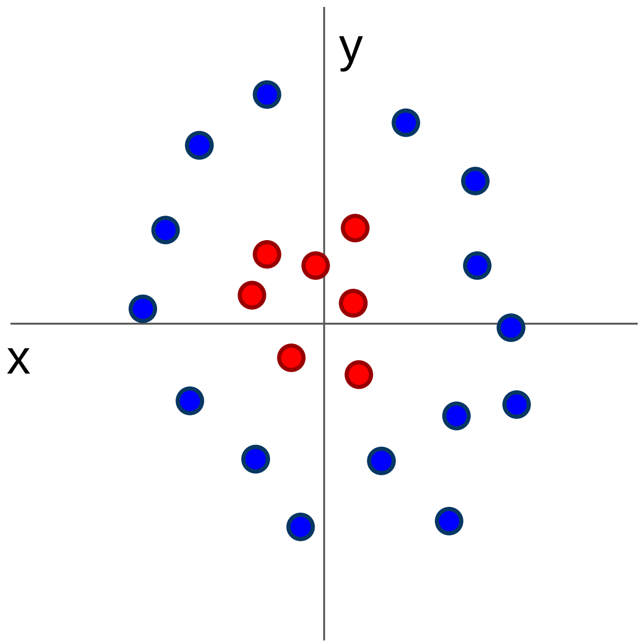
$f(x) = Wx$



Feature Representation

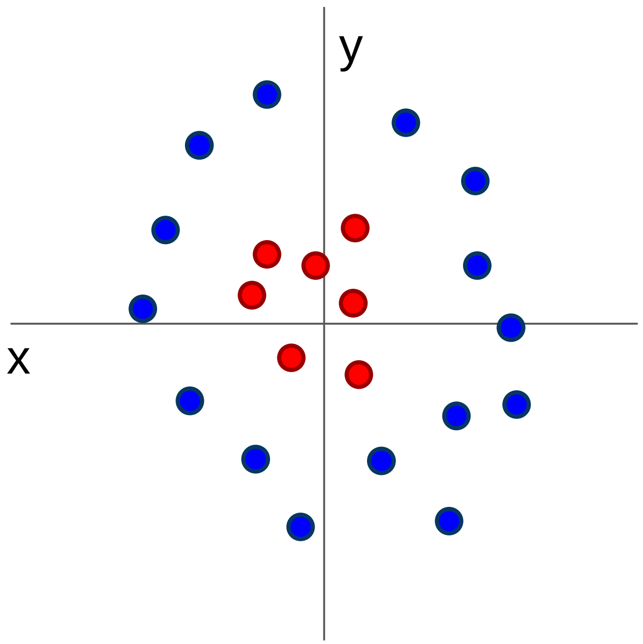
Class scores

Image Features: Motivation

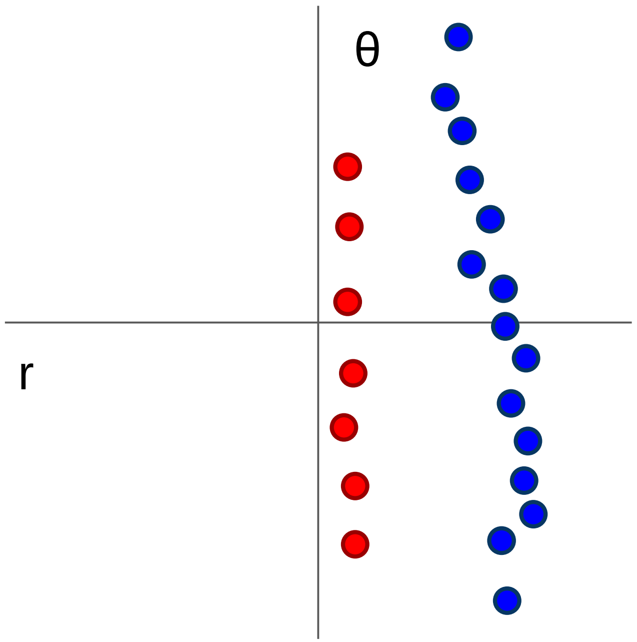


Cannot separate red and blue points with linear classifier

Image Features: Motivation



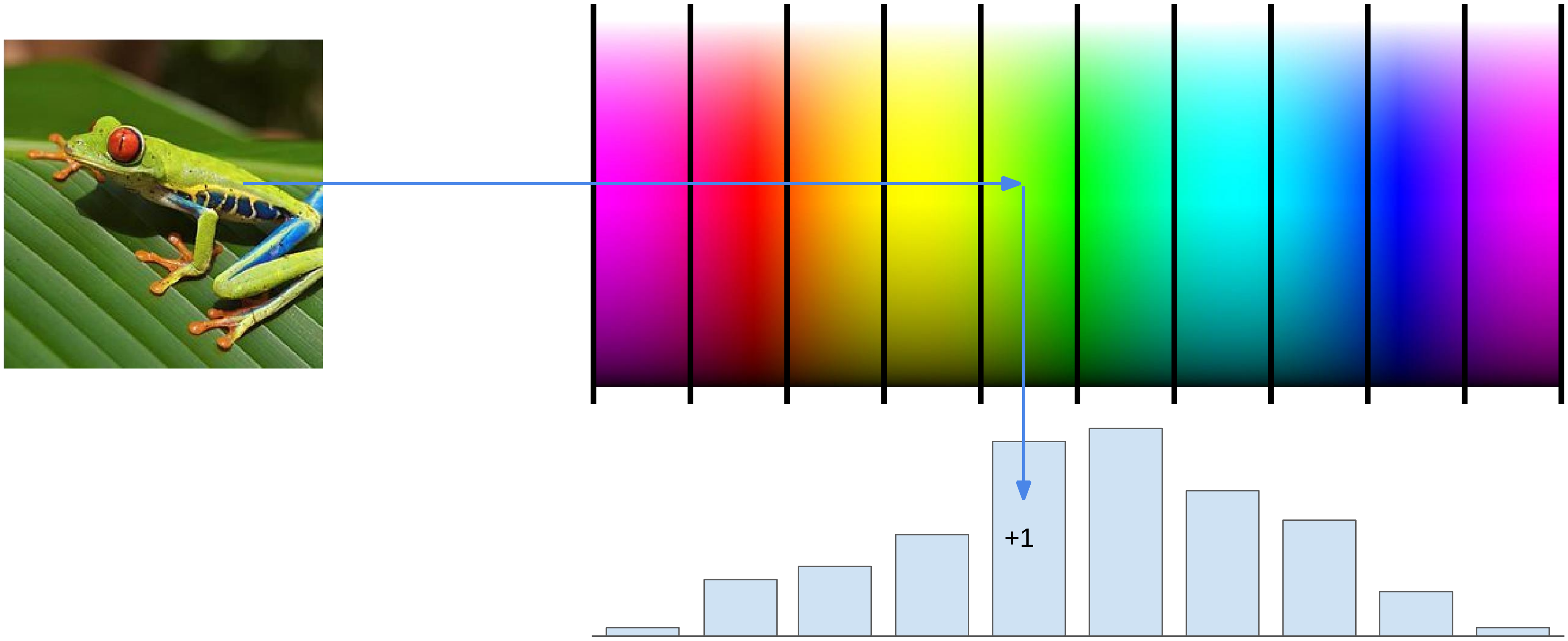
$$f(x, y) = (r(x, y), \theta(x, y))$$



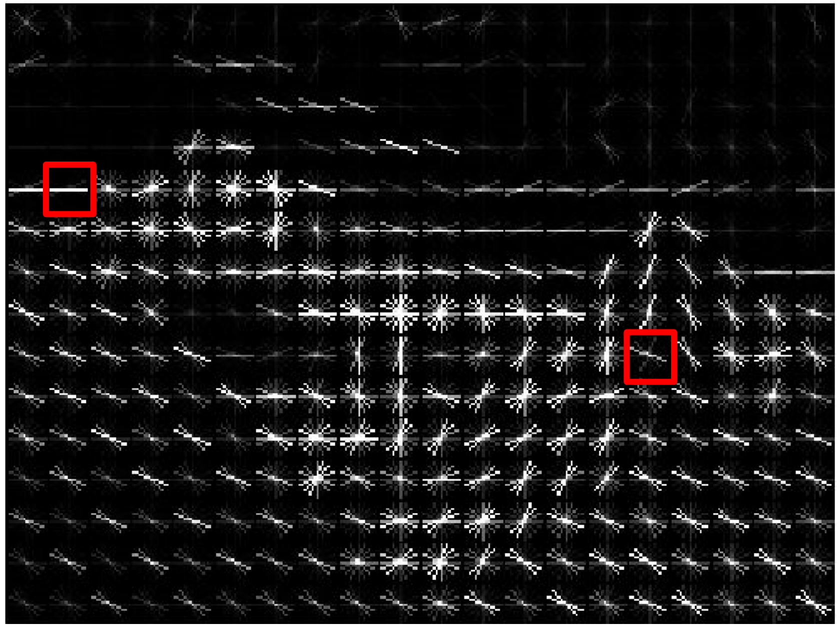
Cannot separate red and blue points with linear classifier

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

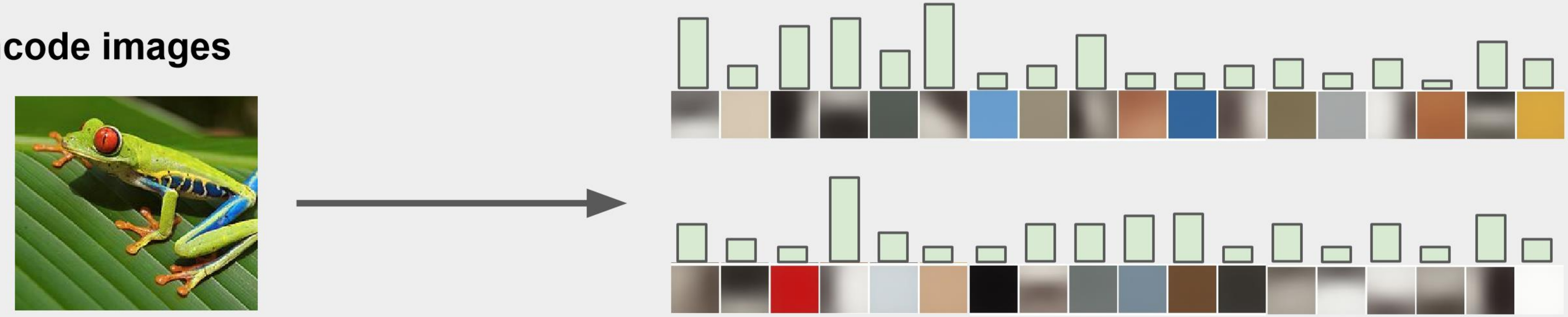
Example: 320x240 image gets divided into 40x30 cells; in each cell there are 9 numbers so feature vector has 30*40*9 = 10,800 numbers

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

Step 2: Encode images



Fei-Fei and Perona, “A bayesian hierarchical model for learning natural scene categories”, CVPR 2005

Image Features

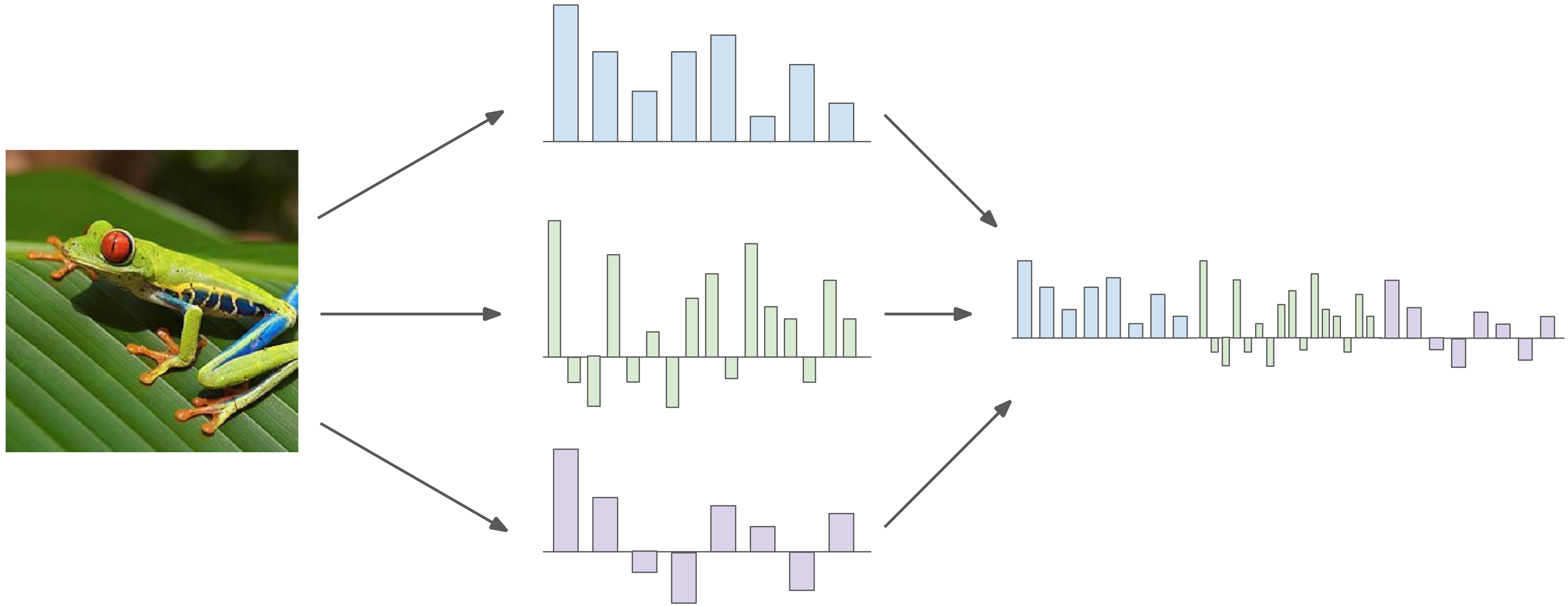


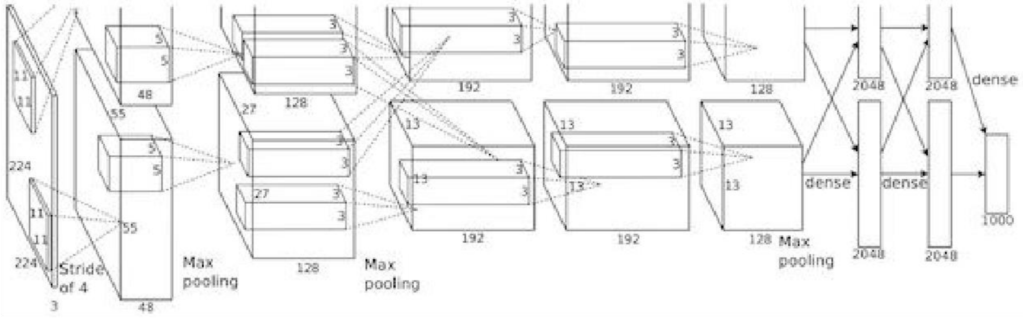
Image features vs ConvNets

Feature Extraction

f

10 numbers giving scores for classes

training



Krizhevsky, Sutskever, and Hinton, "Imagenet classification with deep convolutional neural networks", NIPS 2012.

10 numbers giving scores for classes

training

В следующий раз

- Введение в нейронные сети
- Backpropagation