ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

# Современные нейросетевые технологии

## Лекция 3. Обучение линейного классификатора изображений

## Учебные вопросы

ФИНАНСОВЫЙ УНИВЕРСИТЕТ

1) Стохастический градиентный спуск (SGD)
2) Регуляризация параметров W
3) Производные функции нескольких переменных

**Материалы курса:**
**github.com/balezz/modern_dl**
**Срок сдачи А2 – 17.09.2022 г.**

**Источники:**
- dlcourse.ai
- cs231n.stanford.edu
- cs230.stanford.edu

ФИНАНСОВЫЙ УНИВЕРСИТЕТ

# **Softmax Classifier** (Multinomial Logistic Regression)

cat **3.2**

car 5.1

frog -1.7

**scores = unnormalized log probabilities of the classes.**

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$ where $$s = f(x_i; W)$$

Want to maximize the log likelihood, or (for a loss function) to minimize the negative log likelihood of the correct class:

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

---

in summary: $$L_i = -\log(\frac{e^{sy_i}}{\sum_j e^{s_j}})$$

ФИНАНСОВЫЙ УНИВЕРСИТЕТ

# **Softmax Classifier** (Multinomial Logistic Regression)

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

unnormalized probabilities

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

exp → normalize →

unnormalized log probabilities

probabilities

→ L_i = -log(0.13)
= **0.89**

# Поиск оптимальных параметров

ФИНАНСОВЫЙ УНИВЕРСИТЕТ

## Поиск оптимальных параметров

ФИНАНСОВЫЙ УНИВЕРСИТЕТ

**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 dW:**

[-2.5,
?,
?,

(1.25322 - 1.25347)/0.0001
= -2.5

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

?,
?,…]

## Поиск оптимальных параметров

ФИНАНСОВЫЙ УНИВЕРСИТЕТ

**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 dW:**

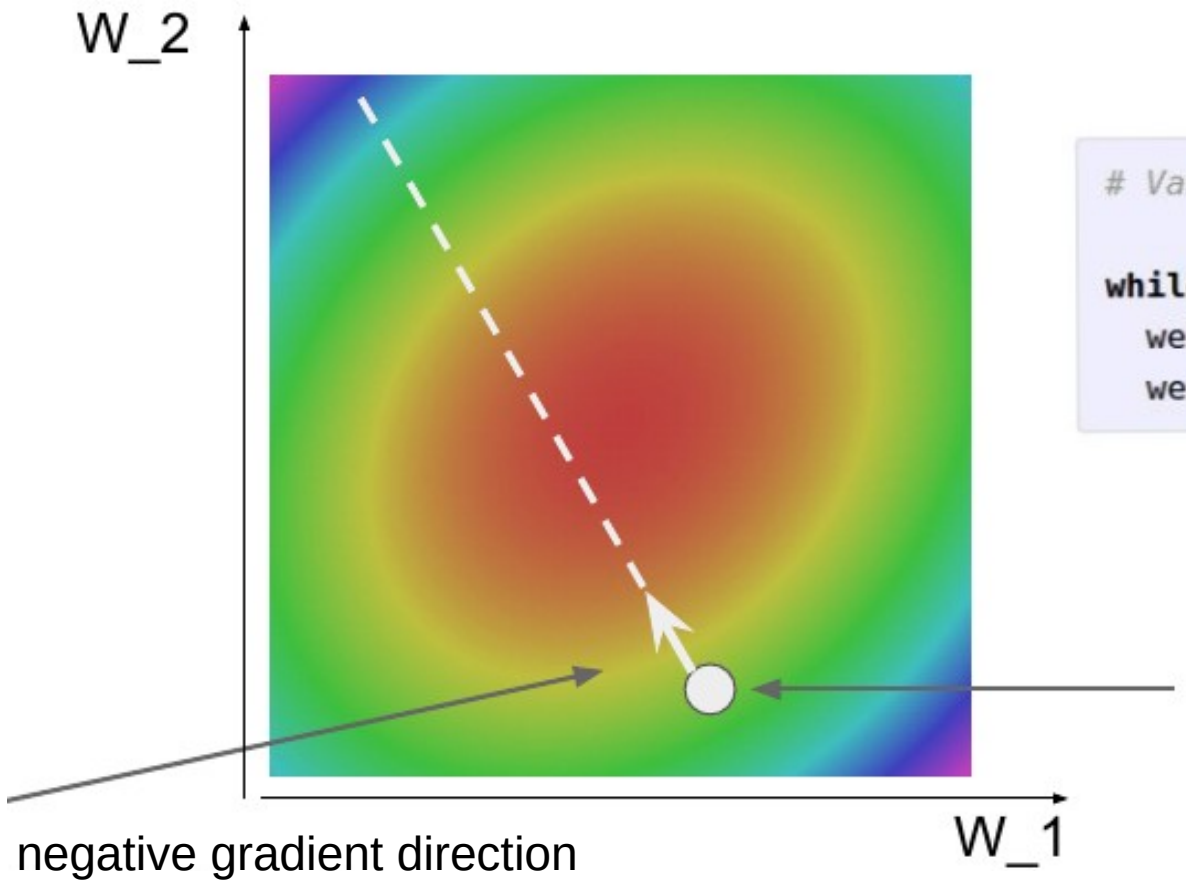
[-2.5,
**0.6**,
?,
?,

(1.25353 - 1.25347)/0.0001
= 0.6

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

?,…]

# Поиск оптимальных параметров




```
# Vanilla Gradient Descent

while True:
  weights_grad = evaluate_gradient(loss_fun, data, weights)
  weights += - step_size * weights_grad # perform parameter update
```

# Поиск оптимальных параметров

ФИНАНСОВЫЙ УНИВЕРСИТЕТ

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

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

Full sum expensive when N is large!

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

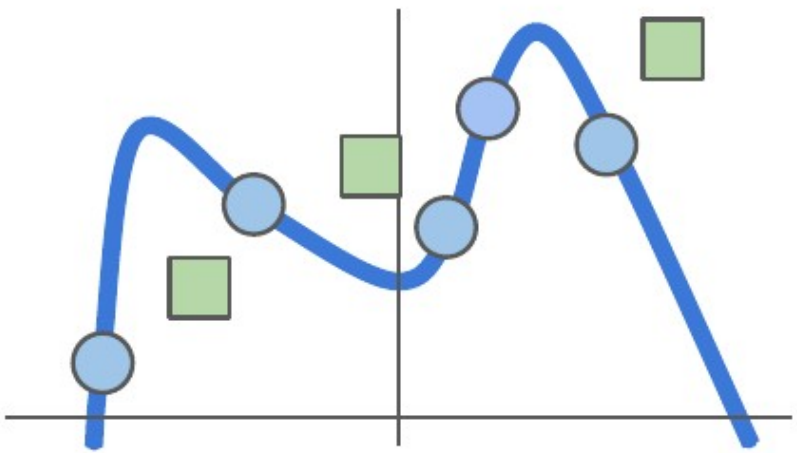
```
# Vanilla Minibatch Gradient Descent

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
```

# Поиск оптимальных параметров

# Регуляризация параметров W

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

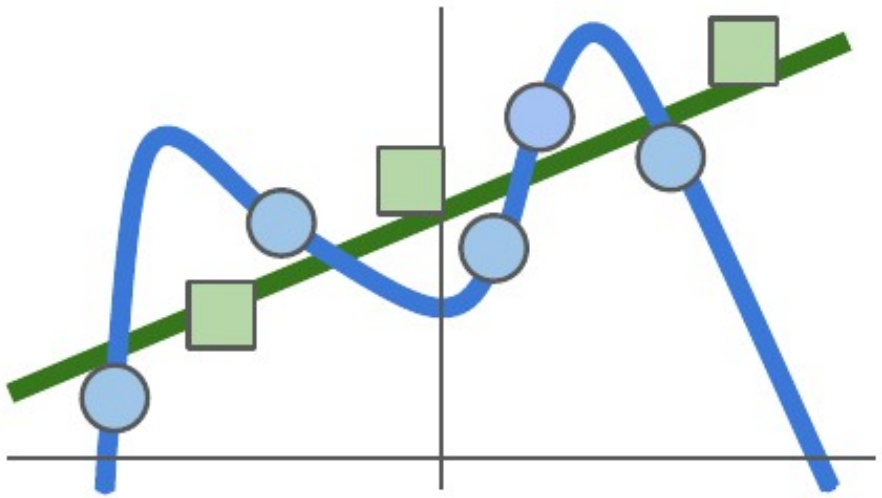
**Data loss**: Model predictions should match training data



# Регуляризация параметров W

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

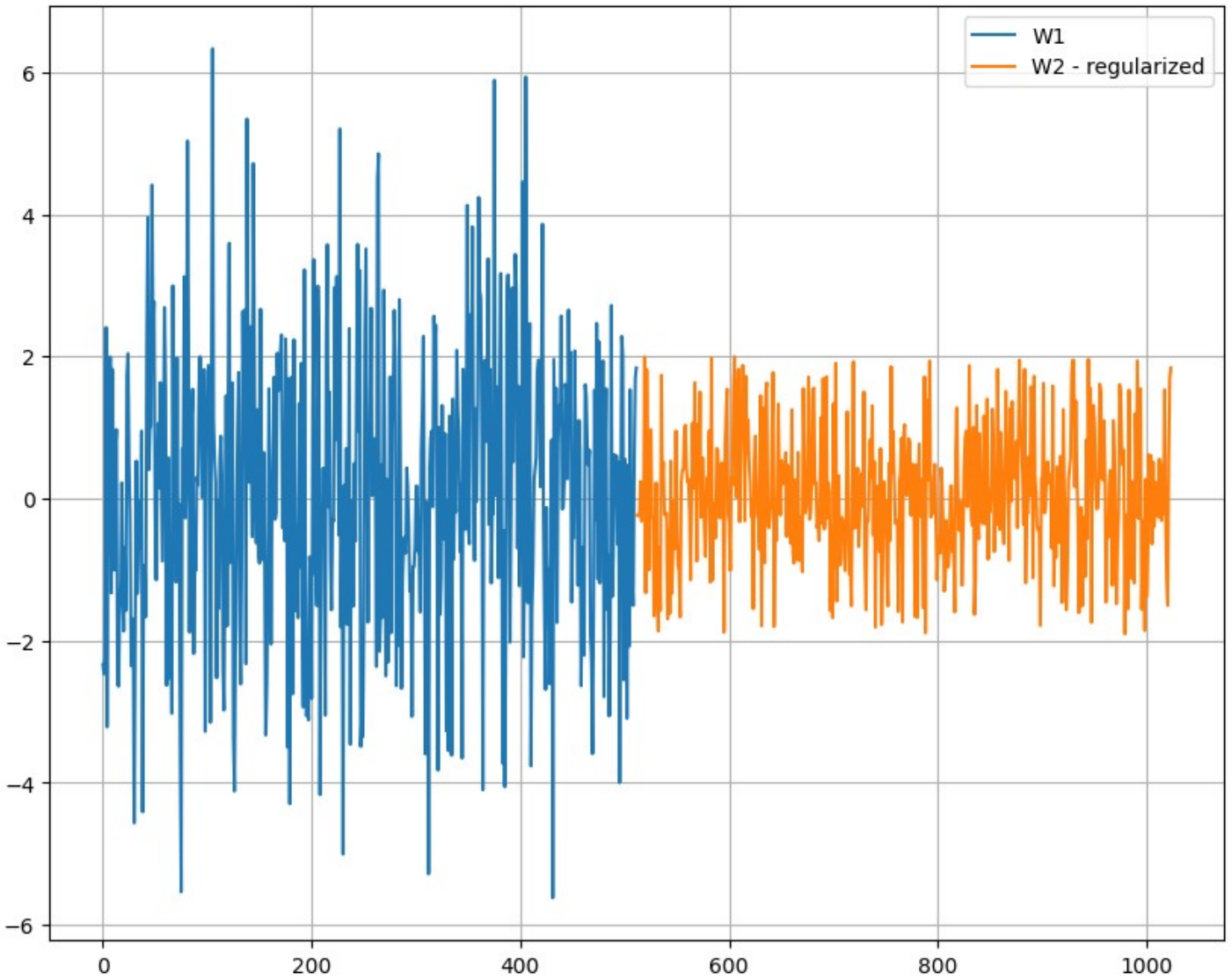
**Data loss**: Model predictions should match training data



**Regularization**: Model should be “simple”, so it works on test data

**Occam’s Razor**:
*“Among competing hypotheses, the simplest is the best”*
William of Ockham, 1285 - 1347

# Регуляризация параметров W




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

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

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

ФИНАНСОВЫЙ УНИВЕРСИТЕТ

# L2 Regularization (Weight Decay)

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

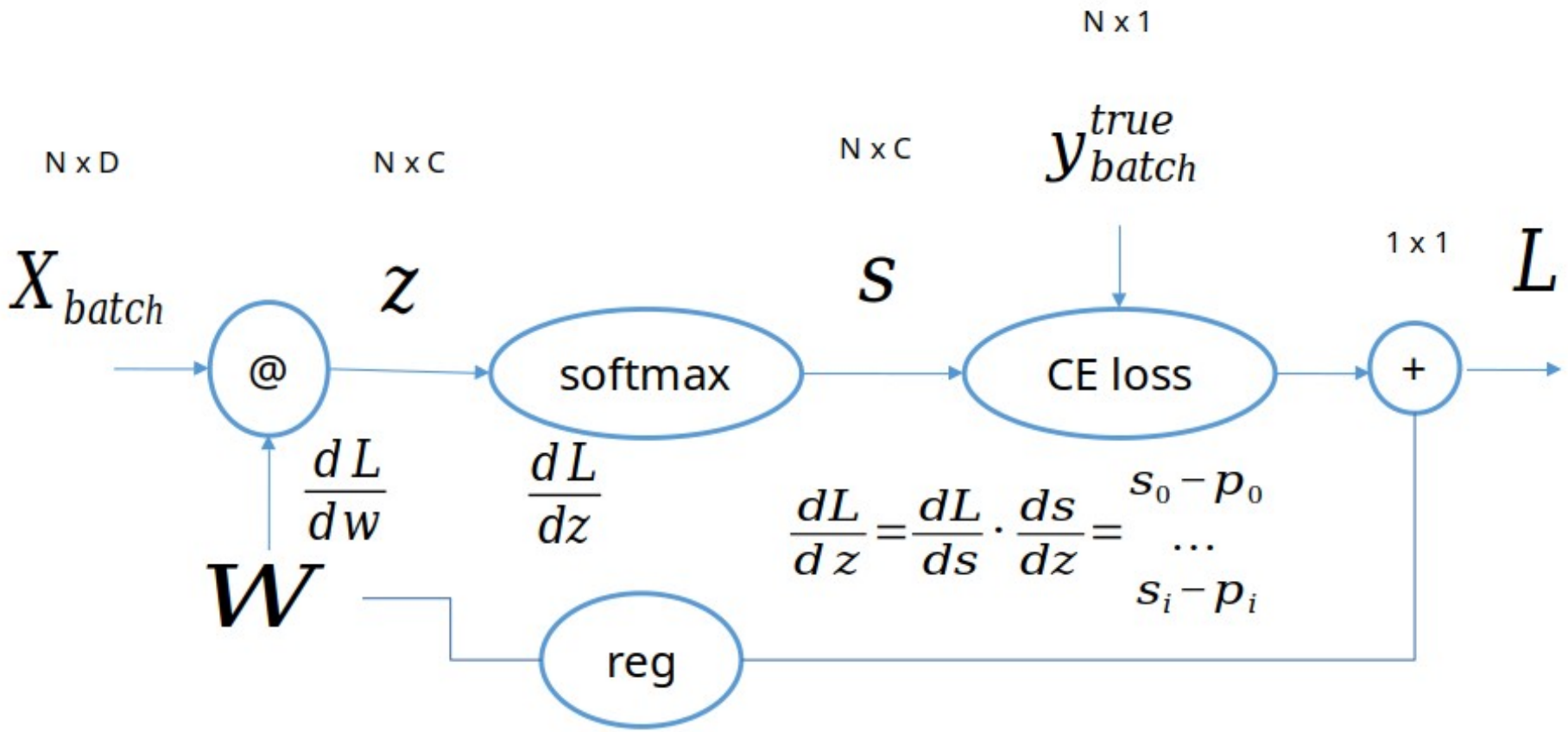
$$R(W) = \sum_k \sum_l W_{k,l}^2$$

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

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

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

# Регуляризация параметров W

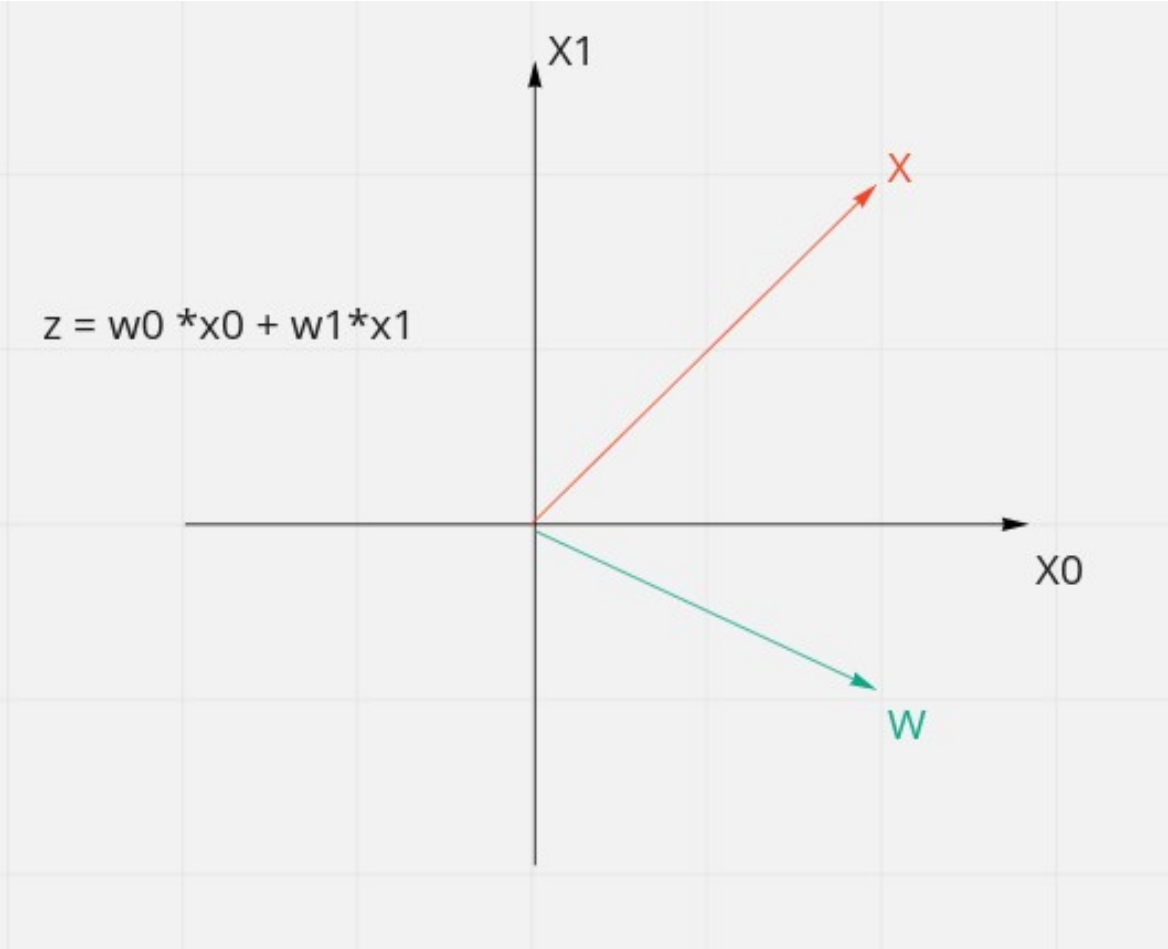



N – number of batch samples
D – dimension of features
C – class

$$s = \frac{e^z}{\sum e^z}$$

$$L = -\frac{1}{N} \sum p \cdot \log(s) + \|W\| \cdot reg$$

# Производная функции нескольких переменных



ФИНАНСОВЫЙ УНИВЕРСИТЕТ

# Gradient descent

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

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

In practice: Derive analytic gradient, check your implementation with numerical gradient

# Производная функции нескольких переменных

ФИНАНСОВЫЙ УНИВЕРСИТЕТ