

# Introduction to Deep Learning Theory

Empirical Risk and its Approximation. Loss Function. (Stochastic) Gradient Descent.  
MLE and MAP. KL-divergence and Cross Entropy.

Aleksandr Petiushko

MIPT  
RAIRI

February 9, 2023



AP

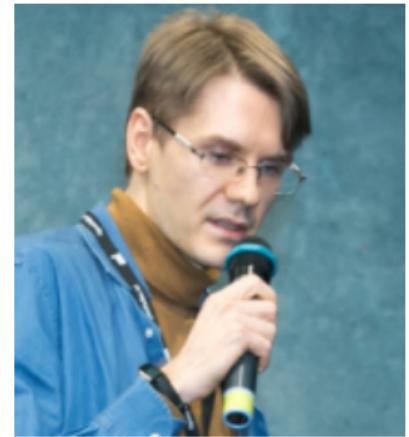
# Content

- ➊ Intro
- ➋ Empirical Risk and its Minimization
- ➌ Separating hyperplane
- ➍ GD and SGD
- ➎ Regularization and MAP
- ➏ Logistic Regression and MLE
- ➐ Cross-Entropy and Kullbak-Leibler

AP

## About lecturer<sup>1</sup>

- Aleksandr Petiushko, PhD in theoretical CS (2016)
- Lecturer in Lomonosov MSU / MIPT for Machine Learning, Computer Vision, Deep Learning Theory, Python for an ML Researcher since 2019
- Former Huawei Chief Scientist (Scientific Expert), AIRI Director of Key Research Programs (Leading Scientific Researcher)
- Currently at Nuro, leading the Autonomy Interaction Research



<sup>1</sup>Homepage: <https://petiushko.info/>

## About lectures

- Will be done: orally in Russian, materials in English
- Supposed to be held weekly on Thursdays, 18:00
- More about implied logic inside, than cool stuff outside
- The lecturer can make mistakes (maybe even severe sometimes): feel free to correct!
- Our main telegram channel: [https://t.me/+tq\\_ebegjIJ4xMDFi](https://t.me/+tq_ebegjIJ4xMDFi)
- Pdfs (and probably videos) will be shared
- Final reporting form: exam
- Possible: own project, or even publication

nanos gigantum humeris insidentes<sup>2</sup>

---

<sup>2</sup>Bernard of Chartres, 12th century: “We are dwarfs, standing on the shoulders of giants”

# Instance-based learning

учить

определять  
y<sup>\*</sup> - бп

использовать бп - бп

- $X$  – set of objects descriptions,  $Y$  – set of objects labels
- Unknown target dependency: mapping  $y^* : X \rightarrow Y$
- Finite training set:  $X^m = \{(x_1, y_1), \dots, (x_m, y_m)\}$ , so as  $y_i = y^*(x_i)$
- **Task of inductive (or instance-based) learning:** construct the algorithm  $a : X \rightarrow Y$ , to approximate the target dependency  $y^*$  not only on training set  $X^m$ , but also on the whole set  $X$
- **Empirical Risk** – average error of  $a$  on  $X^m$
- **Empirical Risk Minimization (ERM)** – the common approach to solve the broad range of tasks of inductive learning (e.g., classification / regression tasks)

Качество предсказаний оценивается

# Empirical risk: definitions

## Loss function $L(y, y')$

Characteristics of difference between the prediction  $y = a(x)$  and the *ground truth* label  $y' = y^*(x)$  for object  $x \in X$

## Set of algorithms $A = \{a : X \rightarrow Y\}$

We will conduct the search of mapping approximating the unknown target dependency inside this set

## Empirical Risk (ER)

Performance metric reflecting the average error made by an algorithm  $a$  upon the set  $X^m$ :

$$R(a, X^m) = \frac{1}{m} \sum_{i=1}^m L(a(x_i), y^*(x_i))$$

# Empirical Risk Minimization

## Empirical Risk Minimization (ERM)

Given a set of algorithms  $A$  need to find the algorithm minimizing the empirical risk:

$$a = \arg \min_{a \in A} R(a, X^m)$$

### ERM pros

Universal and constructive approach allowing to reduce the learning task to the task of numerical optimization

### ERM cons

Overfitting on the training set  $X^m$ . Happens almost always when using ERM, because the performance criteria is the error **on the very same set** (solution: to measure the performance it makes sense to change the set)

# Loss functions examples

cp-1 Krasnaya  
zagora on tverdym

## Classification task

- Classification error:  $L(y, y') = [y \neq y'] = 1 - \delta_{y'}(y)$
- The function is discontinuous  $\Rightarrow$  ERM is a task of combinatorial optimization  $\Rightarrow$  in many practical applications can be reduced to the search of maximal consistent subsystem of inequality system (number of inequalities is equal to the number of training examples  $m$ )  $\Rightarrow$  NP-hard



## Regression task

- Squared error:  $L(y, y') = (y - y')^2$

- Loss function for classification task  $L(y, y') = [y \neq y']$  is not differentiable (and hard to minimize)
- To utilize well-known differentiable methods two new definitions are introduced — *separating surface* and *ER Approximation*
- Consider the task of binary classification:  $X \rightarrow Y, Y = \{+1, -1\}$  using training set  $X^m = (x_i, y_i)_{i=1}^m$
- Algorithm's search will be done in terms of  $a(x, w) = \text{sign } g(x, w)$ , where  $g(x, w)$  — discriminant function,  $w$  — vector of parameters (to learn)

  $g(x, w) = 0$  - **separating surface** (boundary between classes); !  
classification error then is  $a(x_i, w) \neq y_i \Leftrightarrow y_i g(x_i, w) < 0$ .

## Separating surface: variability

Горбатая поверхность  
и гамма

- The simplest variant of a separating surface is the (straight) line (or hyperplane in general case)
- Separating surface can be **non-linear**
- There can be **multiple** separating surfaces

Не всегда существует  
одинаковый результат



Appox. function. maxa

- We can redefine the classification error using the separating surface definition:  
 $a(x_i, w) \neq y_i \Leftrightarrow y_i g(x_i, w) < 0$
- But it is needed to introduce the approximation  $\tilde{R}$  of ER  $R$  with the following properties:
  - $\tilde{R}$  — differentiable
  - $\tilde{R}$  — upper bound for ER  $R$  (so as the minimization of  $\tilde{R}$  implies the minimization of  $R$ )
- ERM:  $R(a, X^m) = \frac{1}{m} \sum_{i=1}^m [y_i g(x_i, w) < 0] \leq \tilde{R}(a, X^m) = \frac{1}{m} \sum_{i=1}^m L(y_i g(x_i, w))$ , where the new loss function  $L(y_i g(x_i, w))$  is non-increasing and non-negative approximation of function  $[y_i g(x_i, w) < 0]$ , i.e.:  $L(y_i g(x_i, w)) \geq [a(x_i, w) \neq y_i]$

**Excercise.** Why we need the properties of non-increase and non-negativity of  $L$ ?

**Note.** In what follows, we'll assume the manipulations directly with ER approximation  $\tilde{R}$ , therefore the sign  $\sim$  will be omitted.

$$R_{\text{emp}}(h) = \frac{1}{n} \sum_{i=1}^n L(h(x_i), y_i)$$

Spezif.  
 fkt.  
 ap-wei haseyo AP

Probabilistic view on minimization of ER approximation

Что такое правонагогоре? Типы. ≠ Вероятность

Consider the principle of Maximum Likelihood Estimation (MLE).

- Parametric probability density model:  $p(x, y|w) = \prod_{i=1}^m p(x_i, y_i|w)$
- Maximization of the log-likelihood:

Несколько.

Собственное  
деление  
на оси X  
одна точка

$$LL(w, X^m) = \ln \prod_{i=1}^m p(x_i, y_i|w) = \sum_{i=1}^m \ln p(x_i, y_i|w)$$

$\rightarrow \max_w$  Требует  
оценки

- Minimization of ER approximation (ERA):

Несколько  $w$

Собственное  
деление  
на оси Y

$$R(w, X^m) = \frac{1}{m} \sum_{i=1}^m L(y_{ig}(x_i, w)) \rightarrow \min_w$$

Правило –  
один из  
принципов  
на оси X

одна точка

одна верн. в то  
же время,

требует  
оценки

значима  
при всех методах

верн. в то же  
время

**Conclusion.** These two principles are equivalent under assumption

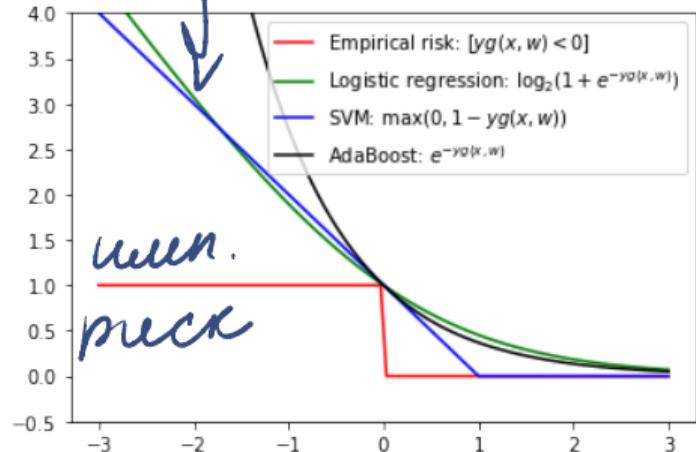
$L(y_{ig}(x_i, w)) = -\ln p(x_i, y_i|w)$  (coefficient  $\frac{1}{m}$  doesn't affect the optimization procedure).

# About Approximation

- Consider approximation of error for training example:  $L(y_i g(x_i, w)) \geq [y_i g(x_i, w) < 0]$
- Hereafter we'll consider mostly continuously differentiable functions  $L(y_i g(x_i, w))$
- Some approximations can improve the **generalization ability** of classifier
- Continuous approximations allow the application of known **numerical optimization methods** for tuning the weights  $w$  (e.g., gradient methods / methods of convex optimization)

maxue  
y momus  
gradient Tb

Approximation examples of function  $[yg(x, w) < 0]$ :



# Classical Gradient Descent → *наиболее простой* - *одно*

Task: to minimize the ERA (algorithm search space is based on  $w$ ):

$$R(w) = \frac{1}{m} \sum_{i=1}^m L(y_i g(x_i, w)) = \frac{1}{m} \sum_{i=1}^m L_i(w) \rightarrow \min_w$$

*близко к нулю*  
*близко к нулю*  
*близко к нулю*  
↑

Numerical optimization by gradient descent (GD) method

- $w^{(0)} :=$  some initialization
- $w^{(t+1)} := w^{(t)} - \eta \cdot \nabla R(w^{(t)})$  – algorithm iteration
- $\eta$  – gradient step

*learning rate*

*некоторый град*  
*алгоритм*  
*η и.т. большую*  
*снижение*  
*нестабильность*

Problem: complex calculations in case of large number of examples inside training set.

## Stochastic Gradient Descent

SGD

Стохастический градиентный спуск  
использует один шаг

### Stochastic Gradient Descent (SGD) algorithm

- Weights initialization  $w$
- Initialization of ERA:  $R := \frac{1}{m} \sum_{i=1}^m L_i(w)$

design choice

### Iterations

- Object selection  $x_i \in X^m$  (e.g., by random choice)
- Chosen object's error calculation:  $\varepsilon_i = L_i(w)$
- Gradient descent step:  $w := w - \eta \cdot \nabla L_i(w)$
- Update of (smoothed) ERA:  $R := (1 - \lambda)R + \lambda \varepsilon_i$

**Note:** smoothing factor  $\lambda \in [0, 1]$  (can start e.g. with 0.1).

неприменим  
в задачах с градиентом

# SGD variability

## Initialization

- $w_j = 0 \quad \forall j = 1, \dots, n$  (where  $n$  is the weight number)
- $w_j = \text{rand}(-\frac{1}{2n}, \frac{1}{2n})$
- Pre-train on different training set



## Object's selection order $x_i \rightarrow$ как выбрать?

- Random shuffle: just randomly take the objects of different classes
- Take more frequently the objects with bigger error (small value of  $y_{ig}(x_i, w)$ )
- Take more frequently the objects with bigger uncertainty (small value of  $|y_{ig}(x_i, w)|$ )

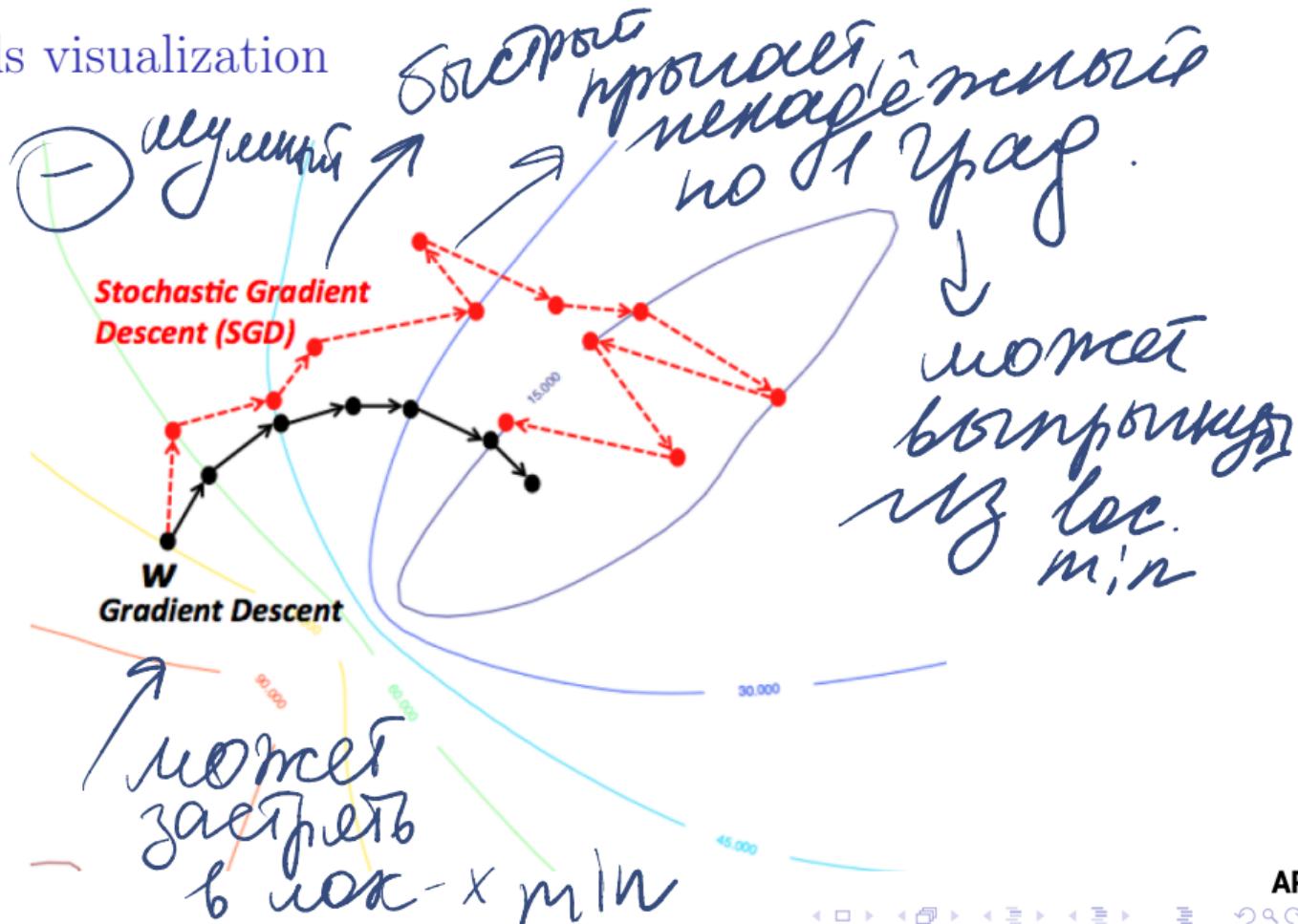
ну  
ну

## Stopping criteria

Когда заканчивать непрерывно?

- Exhausted the upper limit on step (iteration) number
- Values of ERA / weights stuck on some plateau (no any significant change)

## Gradient methods visualization



## Batched SGD

Собираются

некоторые - сел

Mini-batch SGD

→ непримечательно

**Idea:** to use more robust estimation of the gradient taking into account not single but multiple examples at every step (sort of trade-off between GD and SGD)

### Iterations

- Object subset selection of cardinality  $1 < k < m$ :  $J = \{i_1, \dots, i_k\}$
- Error calculation for those objects:  $L_{i_1}(w^{(t)}), \dots, L_{i_k}(w^{(t)})$
- Gradient descent step:  $w^{(t+1)} := w^{(t)} - \eta \cdot \frac{1}{k} \sum_{j=1}^k \nabla_w L_{i_j}(w^{(t)})$

## SGD gradient step selection: some theoretical notes

Скорость только для борьбы  
и уменьшения ошибки.

кто бы  
был  
ошибки

- Convergence is guaranteed<sup>3</sup> only for convex functions of ERA and bounded gradient under

$$\eta_t \rightarrow 0, \sum_{t=0}^{\infty} \eta_t = \infty, \sum_{t=0}^{\infty} \eta_t^2 < \infty$$

- E.g.  $\eta_t = \frac{1}{t}$  ~~номер шага алгоритма~~
- Convergence speed is also  $O(\frac{1}{t})$

no learning rate

огранич

Погодка в зоне

исследование

- Steepest gradient descent method

$$R(w - \eta \nabla R(w)) \rightarrow \min_{\eta}$$
 allows to find the optimal  $\eta^*$

установлено

<sup>3</sup>Robbins, H. and Monro, S. (1951). "A stochastic approximation method"

# SGD gradient step selection

## Means of SGD gradient step control

- Decrease (e.g., divide by 2...10) every  $N$  iterations;
- Decrease (e.g., divide by 2...10) every  $N$  iterations, when the value of ERA stopped to significantly change for the last  $K$  steps; *если не меняется*  
*значение*
- Usage of “warm-up”<sup>4</sup> strategy;
- Usage of cosine<sup>5</sup> / linear<sup>6</sup> law (instead of discrete divisions) for gradient step update;
- Usage of cyclicality<sup>7</sup>.

**Note.** Because of local minimums, sometimes it makes sense to make larger steps to jump out of them.

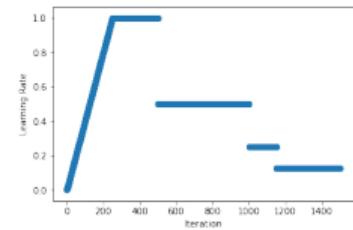
<sup>4</sup>Goyal, Priya, et al. “Accurate, large minibatch sgd: Training imagenet in 1 hour.” 2017

<sup>5</sup>Loshchilov, Ilya, and Frank Hutter. “Sgdr: Stochastic gradient descent with warm restarts.” 2016

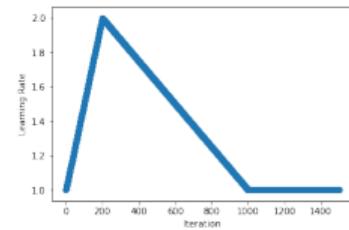
<sup>6</sup>Howard, Jeremy, and Sebastian Ruder. “Universal language model fine-tuning for text classification.”

# SGD gradient step selection illustration

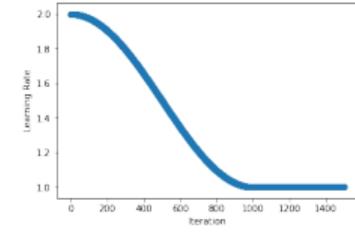
Warm-up



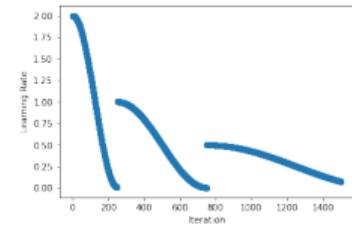
Triangular



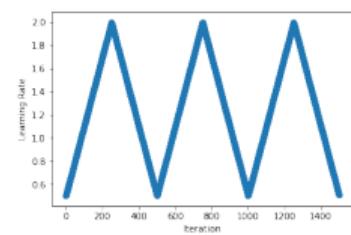
Cosine



Cosine + division



Cycle



## About overfitting ↴ неправильные

### Overfitting causes

- Small training set / large number of weights; *очень мало примеров*
- Large number of object's features; *Большое кол-во блогов*
- Non-informative (noisy / dependent) object's features. *Большое кол-во*

### Overfitting observation

- Sudden norm change of  $w$  (attention to the specific object's features); *норма*
- Significant difference between train and test set errors. *нар 1 приставка*

*Горячий трек* *Большой тест* *один вопрос*

### Overfitting reduction

- Weights norm decrease (**regularization**);
- Cross-validation procedure,
- Earlier training stop.

*Позитивное отрицание*

# Принцип максимальной постериворной вероятности

Consider the principle of **Maximum A Posteriori Probability** (MAP).

Given:

- Parametric probability density model  $p(x, y|w)$
- Prior probability density of model weights  $p(w)$   
E.g., parametric family of priors  $p(w; h)$ , where  $h$  – unknown and fixed value (hyperparameter).

Then:

- Posterior probability by Bayes' rule:  $p(w|X^m) = \frac{p(X^m|w)p(w;h)}{p(X^m)} \propto p(X^m|w)p(w;h)$
- Maximizing of the log-posterior:

$$-L(w, X^m) = \ln p(w|X^m) = \sum_{i=1}^m \ln p(x_i, y_i|w) + \ln p(w; h) \rightarrow \max_{w,h}$$

М

# A probabilistic view on regularization

Бережіть ненадані даний

$$-L(w, X^m) = \sum_{i=1}^m \ln p(x_i, y_i | w) + \ln p(w; h) \rightarrow \max_{w, h}$$

So if the probabilistic family of model weights is the normal one, i.e.

$$p(w; h) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\|w-\mu\|^2}{2\sigma^2}}$$
 with fixed  $\mu = 0$  и  $\sigma$ , then

$$\sum_{i=1}^m \ln p(x_i, y_i | w) + \ln p(w) \rightarrow \max_w \Leftrightarrow - \sum_{i=1}^m \ln p(x_i, y_i | w) + \frac{\tau}{2} \|w\|^2 \rightarrow \min_w$$

L<sub>2</sub> Penalt

**Exercise.** Find the explicit dependency  $\tau(\sigma)$ .

## $L_2$ -regularization and SGD

Consider the squared penalty on weights norm inside ERA optimization:

$$R_\tau(w, X^m) = R(w, X^m) + \frac{\tau}{2} \|w\|^2 \rightarrow \min_w$$

*y Pack e  
yretou  
anocrep. bsp.*

Then ERA gradient:  $\nabla R_\tau(w, X^m) = \nabla R(w, X^m) + \tau w,$

And gradient step:  $w^{(t+1)} = (1 - \tau\eta)w^{(t)} - \eta \nabla R(w^{(t)}, X^m).$

### Selection of regularization factor $\tau$

- Large value of  $\tau$  – bigger penalty on overfitting (but the convergence is slower!);
- By means of cross-validation procedure.

**Note.** This is also called “Weight Decay” (WD) (because weights are decreased every step linearly), and the WD coefficient is usually explicitly set up (and equals to  $\tau\eta$  in notations above).

**Exercise.** How  $L_1$ -regularization can be understood from the probabilistic point of view?

AP

# Logistic regression

## Sigmoid

**Sigmoid** (or logistic function) is the function  $\sigma : R \rightarrow [0, 1]$  so as  $\sigma(z) = \frac{1}{1+e^{-z}}$ .

## Logistic regression

*Klassifikator*

Classification model for two classes ( $Y = -1, +1$ ) where probability of positive<sup>8</sup> class is the sigmoid of linear function of input:  $p(y = +1|x) = \sigma(g(x, w))$ , where  $g(x, w) = \langle w, x \rangle$ .

- Logistic regression decision rule:  $P(y = +1|x) \geq 1/2 \Rightarrow y = +1$
- It implies linear classification rule:  $\sigma(g(x, w)) \geq 1/2 \Leftrightarrow \langle w, x \rangle \geq 0$

<sup>8</sup>Exercise. Prove: for logistic regression the probability of *any* class  $y$  is  $\sigma(g(x, w)y)$ .

## Logistic loss function

Plug into the logistic regression  $p(x, y) = p(y|x) \cdot p(x) = \sigma(g(x, w)y) \cdot const(w)$  into log-likelihood maximization  $LL(w, X^m) = \log \prod_{i=1}^m p(x_i, y_i) \rightarrow \max_w$ :

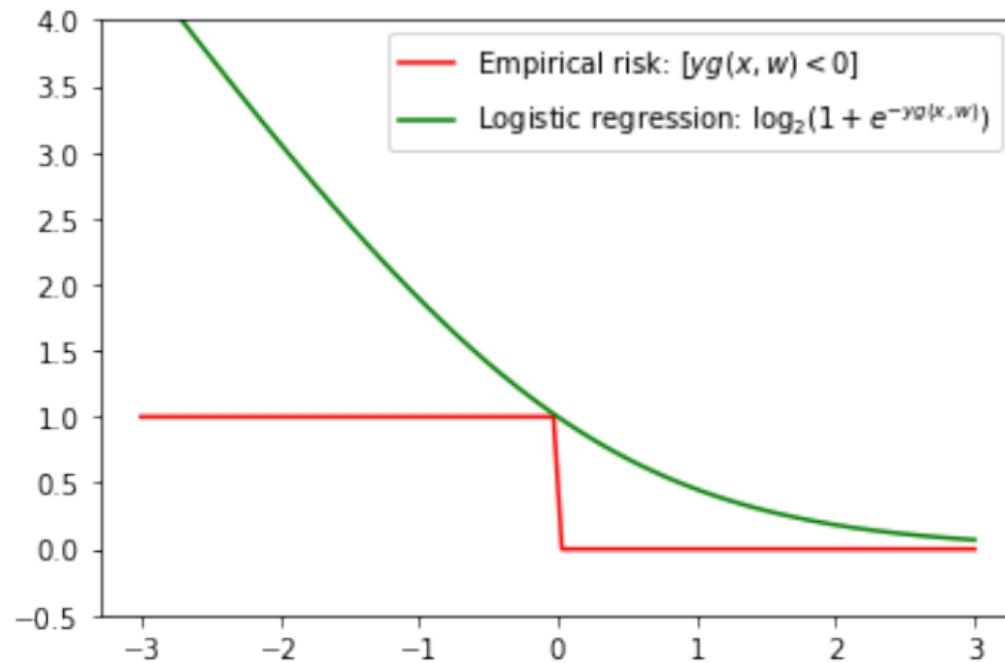
- $LL(w, X^m) = \sum_{i=1}^m \log \sigma(g(x_i, w)y_i) + const(w) \rightarrow \max_w$

So the maximization of  $LL$  is equivalent to minimization of ERA  $R$ :

$$R(w, X^m) = \sum_{i=1}^m \log(1 + \exp(-g(x_i, w)y_i)) \rightarrow \min_w$$

*paralelnye yazyki  
nBB - D.*

## Logistic loss function vs ER



The correct ER approximation.

## Binary cross-entropy and logistic loss

### Binary cross-entropy

Let us assume  $Y = \{0, 1\}$ ,  $p_1 = p(y = 1|x) = \sigma(g(x, w))$  и  $p_0 = p(y = 0|x) = 1 - p_1$ . Then logistic loss (see above slides) can be written<sup>9</sup> in the form of cross-entropy:

$$R(w, X^m) = - \sum_i (y_i \log p(y_i|x_i) + (1 - y_i) \log(1 - p(y_i|x_i)))$$

**Note.** A single-layered neural net with sigmoid activation and a binary cross-entropy loss functions is a logistic regression.

**Note.** But what is the usual (non-binary) **cross-entropy**?

<sup>9</sup>Exercise. Prove it.

## Multi-class sigmoid

In order to consider the multi-class cross-entropy, let's define the multi-class sigmoid.  
Consider multi-class case  $|Y| > 2$ . Then *↪ I forgot prob. in*

- linear classifier<sup>10</sup>  $a(x, w) = \arg \max_{c \in Y} g(x, w^c)$ , where  $g(x, w) = \langle w, x \rangle$
- class probability  $c$  conditioned on  $x$  for this classifier is defined by a so-called **SoftMax** function:

$$\text{SoftMax}(g(x, w^c)) = P(y = c|x, w) = \frac{\exp(g(x, w^c))}{\sum_{z \in Y} \exp(g(x, w^z))}$$
  $\sum=1$

- Classification rule  $\arg \max_{c \in Y} \text{SoftMax}(g(x, w^c)) \Leftrightarrow \arg \max_{c \in Y} \langle w^c, x \rangle$

It means that function  $\text{SoftMax} : \mathbb{R}^{|Y|} \rightarrow \mathbb{R}^{|Y|}$  maps any real-valued vector into a vector of some discrete probability distribution.

Now we are ready for the multi-class logistic loss and cross-entropy.

---

<sup>10</sup>Exercise: prove it!

# One-hot encoding and its entropy

- Define  $p = (p_1, \dots, p_n)$ ,  $\sum_{i=1}^n p_i = 1, 0 \leq p_i \leq 1$ .
- **Entropy**  $H(p) = -\sum_{i=1}^n p_i \log p_i$
- One-hot encoding for the class  $y_c$ :  
 $y_{one-hot} = (0, \dots, 1, \dots, 0), \quad y_i = 0, i \neq y_c, \quad y_i = 1, i = y_c$
- In the case  $i = y_c$ :  $p_i \log p_i = 1 \cdot \log 1 = 1 \cdot 0 = 0$
- And for  $i \neq y_c$  let's use the L'Hopital's rule:  $p_i \log p_i = \lim_{x \rightarrow 0} x \log x = \lim_{x \rightarrow 0} \frac{\log x}{1/x} = \lim_{x \rightarrow 0} \frac{(\log x)'}{(1/x)'} = \lim_{x \rightarrow 0} \frac{1/x}{-1/x^2} = -\lim_{x \rightarrow 0} x = 0$
- All cases are resolved:  $H(p) = -\sum_{i=1}^n p_i \log p_i = 0$ .

Итак, получим нулевую  
функцию

## Multi-class logistic loss

- Assume that the correct label in a one-hot encoding for an object  $x_i$  is  $y_i = (0, \dots, 1, \dots, 0)$ ,  $y_i^j = 0, j \neq c_i$ ,  $y_i^j = 1, j = c_i$  (i.e.  $c_i$  is the correct class)
- The same situation as for binary logistic loss: we are maximizing the log-likelihood  $LL(w, X^m)$ , or equivalently, minimizing the negative log-likelihood  
$$R(w, X^m) = -\sum_{i=1}^m \log p(y_i|x_i) = -\sum_{i=1}^m \log SoftMax(g(x_i, w^{c_i})) =$$
$$-\sum_{i=1}^m (g(x_i, w^{c_i}) - \log \sum_{c \in Y} p(g(x_i, w^c)))$$
- Note, that it can be re-written as a  $R(w, X^m) =$   
$$-\sum_{i=1}^m \sum_{j=1}^n y_i^j \log SoftMax(g(x_i, w^j)) = -\sum_{i=1}^m \sum_{j=1}^n y_i^j \log p(y = j|x_i)$$
- It is the cross-entropy between one-hot encoding of the correct labels and the model output  $p(y|x)$
- But what is interesting about cross-entropy itself? Let's move on

# Gibbs Inequality (1)

- Let's  $p = (p_1, \dots, p_n)$ ,  $\sum_{i=1}^n p_i = 1$ ,  $0 \leq p_i \leq 1$ ,
- $q = (q_1, \dots, q_n)$ ,  $\sum_{i=1}^n q_i = 1$ ,  $0 \leq q_i \leq 1$ .

## Gibbs Inequality

$\sum_{i=1}^n p_i \log p_i \geq \sum_{i=1}^n p_i \log q_i$  for any distributions  $p, q$ , and the equality sign iff  $p = q$ .

## Proof.

- If  $x > 0$  then  $\log x \leq x - 1$ , and the equality sign iff  $x = 1$ .
- Define  $I$  as the set of such indices so as  $p_i > 0$ .
- Then  $\sum_{i \in I} p_i \log q_i - \sum_{i \in I} p_i \log p_i = \sum_{i \in I} p_i \log \frac{q_i}{p_i} \leq \sum_{i \in I} p_i \left( \frac{q_i}{p_i} - 1 \right) = \sum_{i \in I} q_i - \sum_{i \in I} p_i \leq 1 - \sum_{i \in I} p_i = 1 - 1 = 0$

## Gibbs Inequality (2)

### Proof.

- In case of  $p_i = 0$ :
  - ▶  $p_i \log p_i = 0$  (see above),
  - ▶ Then  $\sum_{i:p_i=0} p_i \log q_i - \sum_{i:p_i=0} p_i \log p_i = \sum_{i:p_i=0} p_i \log q_i \leq \sum_{i:p_i=0} p_i \log 1 = 0$ .
- Thereby,  $\sum_{i=1}^n p_i \log p_i \geq \sum_{i=1}^n p_i \log q_i$ , and the equality sign only if  $\frac{q_i}{p_i} = 1$  in case of  $i \in I$ ,
- But then  $p_i = q_i = 0, i \notin I$ . ■

**Corollary 1.** Maximal entropy for  $p = (p_1, \dots, p_n)$ ,  $\sum_{i=1}^n p_i = 1, 0 \leq p_i \leq 1$  is equal to:

$$H(p) = -\sum_{i=1}^n p_i \log p_i \leq -\sum_{i=1}^n p_i \log \frac{1}{n} = \log n \sum_{i=1}^n p_i = \log n.$$

**Corollary 2.** Entropy bounds for  $p = (p_1, \dots, p_n)$ ,  $\sum_{i=1}^n p_i = 1, 0 \leq p_i \leq 1$ :

$$0 \leq H(p) \leq \log n.$$

# Kullback-Leibler divergence

от-дел пагоды

## Definition

Divergence between two distributions, defined on the same space of distribution functions  $S$  – is a function  $D(\cdot||\cdot) : S \times S \rightarrow \mathbb{R}$  with the following properties:

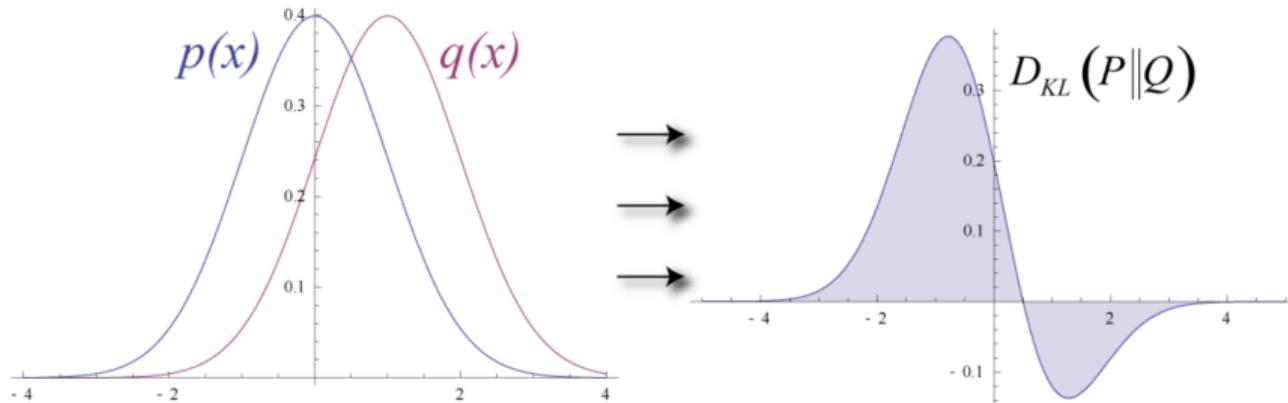
- ①  $D(P||Q) \geq 0$  for all  $P, Q \in S$
- ②  $D(P||Q) = 0 \Leftrightarrow P = Q$

## Kullback-Leibler divergence

$$D_{KL}(P||Q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

**Proof.**  $D_{KL}(P||Q) = \sum_x p(x) \log \frac{p(x)}{q(x)} = \sum_x p(x) \log p(x) - \sum_x p(x) \log q(x) \geq 0$ , and the equality sign iff  $P = Q$  (Gibbs Inequality consequence)  $\Rightarrow D_{KL}$  is a divergence. ■

# Kullback-Leibler divergence

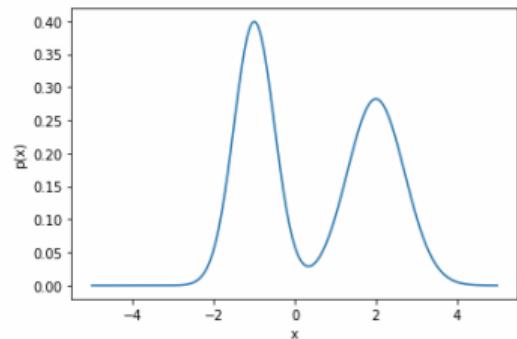


# Kullback-Leibler: forward vs reverse divergence

*discrepancy*

- Suppose we have a distribution  $p$ , and the distribution we'd like to optimize in relation to it is  $q$  *referred to*
- **Forward KL** divergence target: minimize  $D_{KL}(p||q)$   $P > 0$
- **Reverse KL** divergence target: minimize  $D_{KL}(q||p)$
- Minimizing Forward  $D_{KL}$  means minimize the discrepancy in the regions where  $p > 0$   
⇒ it leads to  $q$  covering the support of  $p$   $P > 0$
- Minimizing Reverse  $D_{KL}$  means minimize the discrepancy in the regions where  $q > 0$   
⇒ it leads to  $q$  covering the major mode of  $p$   $q > 0$

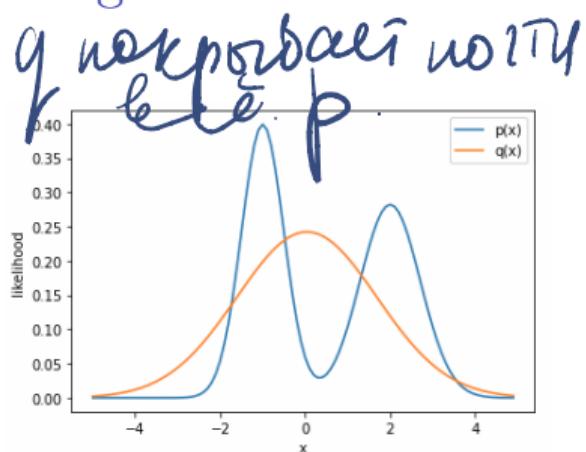
# Forward vs reverse KL divergence illustration<sup>11</sup>



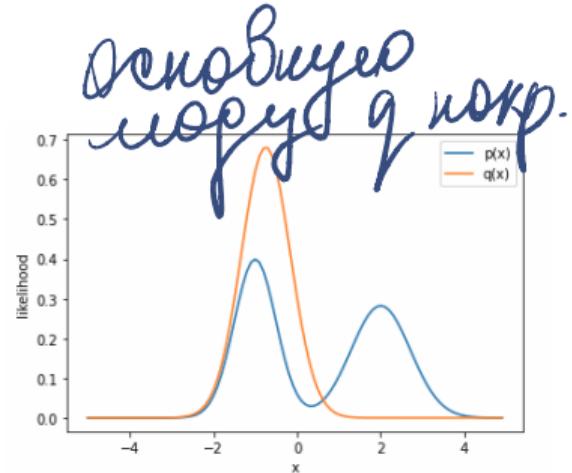
Initial distribution  $p$

Analogies:

- ➊ Forward KL: maximize recall
- ➋ Reverse KL: maximize precision



Optimizing in relation to  
forward KL



Optimizing in relation to  
reverse KL

<sup>11</sup>[towardsdatascience.com](https://towardsdatascience.com)

# Why we use cross-entropy for classification

- Let's  $p$  — one-hot encoding of the true label,  $q$  — output of SoftMax (e.g., the last NN layer)
- Cross-entropy (CE)

$$H(p, q) = - \sum_x p(x) \log q(x) = H(p) + D_{KL}(p||q)$$

- As  $H(p) \geq 0$  and  $D_{KL}(p||q) \geq 0$ , then CE is always non-negative:  $H(p, q) \geq 0$
- For any one-hot encoding entropy is 0 (cf. previous slides)
- In our case  $H(p, q) = D_{KL}(p||q) \geq 0$ , and  $H(p, q) = 0 \Leftrightarrow p = q$
- And this is another universal explanation why to use the CE

**Question.** What is the maximum value of CE? Or KL-divergence?

# Metric based on Jensen–Shannon divergence (1)

## Jensen–Shannon divergence

$$D_{JS}(P||Q) = \frac{1}{2}D_{KL}(P||M) + \frac{1}{2}D_{KL}(Q||M), \text{ where } M = \frac{1}{2}(P + Q).$$

## Metric definition

Function  $d : X \times X \rightarrow \mathbb{R}$  is a metric on the space  $X$ , if:

- ①  $d(x, y) = 0 \Leftrightarrow x = y,$
- ②  $d(x, y) = d(y, x),$
- ③  $d(x, z) \leq d(x, y) + d(y, z).$

## Theorem

$\sqrt{D_{JS}(P||Q)}$  – metric.

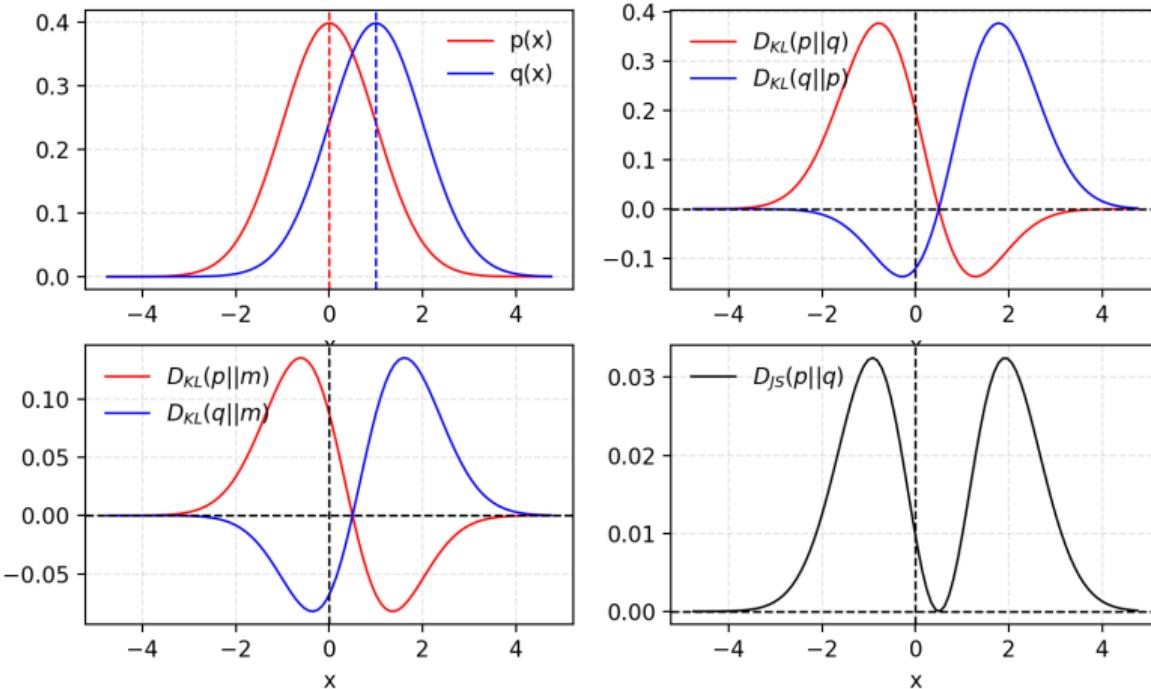
## Metric based on Jensen–Shannon divergence (2)

### Proof.

- ①  $\sqrt{D_{JS}(P||Q)} = 0 \Leftrightarrow D_{JS}(P||Q) = 0 \Leftrightarrow D_{KL}(P||M) = D_{KL}(Q||M) = 0, P = Q = \frac{1}{2}(P + Q)$  (cf. the properties of divergence  $D_{KL}$ );
- ②  $\sqrt{D_{JS}(P||Q)} = \sqrt{\frac{1}{2}D_{KL}(P||M) + \frac{1}{2}D_{KL}(Q||M)} = \sqrt{\frac{1}{2}D_{KL}(Q||M) + \frac{1}{2}D_{KL}(P||M)} = \sqrt{D_{JS}(Q||P)}$ ;
- ③ A little bit more harder to prove the triangle inequality. Please refer to the work<sup>12</sup> about the correct proof. ■

<sup>12</sup>Endres D. M., Schindelin J. E. “A new metric for probability distributions” 2003.

# JS vs different types of KL divergence<sup>13</sup>



<sup>13</sup>[lilianweng.github.io](https://lilianweng.github.io)

# Takeaway notes

→ Kullback-Leibler divergence

- ① Classifier performance is measured by empirical risk
- ② In practice empirical risk approximation is used
- ③ Mini-batch SGD is the practical variant of GD
- ④  $L_2$ -regularization is based on MAP principle
- ⑤ Sigmoid and SoftMax, logistic and cross-entropy loss, and MLE are all connected
- ⑥ Kullback-Leibler divergence is not a metric;  $\text{sqrt}(\text{Jensen-Shannon divergence})$  is
- ⑦ Minimization of CE  $\Leftrightarrow$  KL  $\Leftrightarrow$  making two distributions (ground truth and predicted) closer

# Thank you!