Introduction to Machine and Deep Learning Theory

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

Aleksandr Petiushko

Lomonosov MSU
Faculty of Mechanics and Mathematics

February 8, 2023

AP

Content

AP

Course Scientific Advisor

About advisor[1]

- Ivan Mazurenko, PhD in theoretical CS (2001)
- Senior Researcher in Laboratory for Problems of Theoretical Cybernetics, Lomonosov MSU
- Former Software Engineer Principal, Advanced Software Development Team lead at LSI
- Currently Director at Huawei MRC Intelligent Systems and Data Science Technology Center

[1] Homepage: `http://intsys.msu.ru/staff/mazurenko/`

AP

Course Lecturer

About lecturer[2]

- 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

[2]Homepage: `https://petiushko.info/`

Brief info

About lectures:

- Will be done: orally in Russian, materials in English
- Supposed to be held weekly on Wednesdays, 16:45
- 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/+OX_Ie45QTghjZmJi`
- Pdfs (and probably videos) will be shared: `https://github.com/papermsucode/intromldlt2023spring`
- Final reporting form: exam (for mech and math faculty)
- Possible: own project, or even publication

AP

Motto

nanos gigantum humeris insidentes[3]

[3] Bernard of Chartres, 12th century: "We are dwarfs, standing on the shoulders of giants"

Instance-based learning

- X – set of objects descriptions, Y – set of objects labels
- Unknown target dependency: mapping $y^* : X \to Y$
- Finite training set: $X^m = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, so as $y_i = y^*(x_i)$
- **Task of inductive (or instance-based) learning**: construct the algorithm $a : X \to Y$, 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)

AP

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 $A = \{a : X \to Y\}$

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

AP

Empirical Risk Minimmization

Empirical Risk Minimmization (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

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

AP

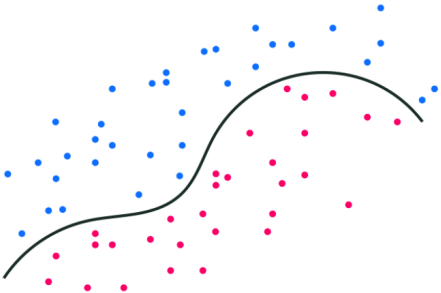
Separating surface

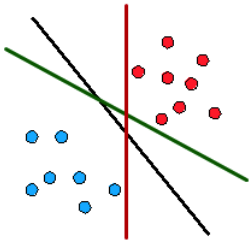
- Loss function for classification task $L(y, y') = [y \neq y']$ is not differentiable (and hard to minimize)
- To utilize well-known differentiable methods two new definitions are introduced — *separating surface* and *ER Approximation*
- Consider the task of binary classification: $X \to Y$, $Y = \{+1, -1\}$ using training set $X^m = (x_i, y_i)_{i=1}^m$
- Algorithm's search will be done in terms of $a(x, w) = \operatorname{sign} g(x, w)$, 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 $a(x_i, w) \neq y_i \Leftrightarrow y_i g(x_i, w) < 0$.

AP

Separating surface: variability

- 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





AP

Empirical Risk Approximation

- 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:
  1. $\tilde{R}$ — differentiable
  2. $\tilde{R}$ — 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 ˜ will be omitted.

AP

Probabilistic view on minimization of ER approximation

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:

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

- Minimization of ER approximation (ERA):

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

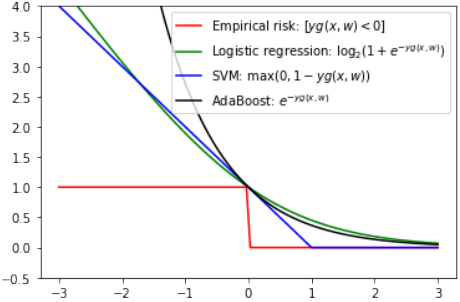
Conclusion. These two principles are equivalent under assumption $L(y_i g(x_i, w)) = -\ln p(x_i, y_i|w)$ (coefficient $\frac{1}{m}$ doesn't affect the optimization procedure).

AP

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)

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



Classical Gradient Descent

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

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

Numerical optimization by gradient descent (GD) method

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

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

AP

Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) algorithm

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

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).

AP

SGD variability

Initialization

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

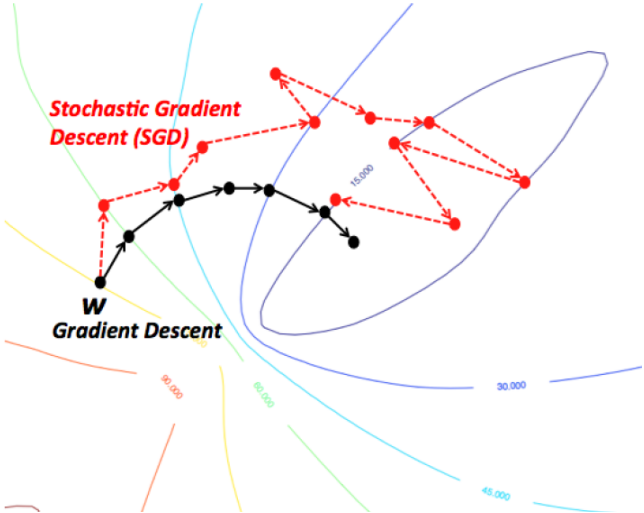
Object's selection order x_i

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

Stopping criteria

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

Gradient methods visualization



AP

Batched SGD

Mini-batch SGD

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

Iterations

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

AP

SGD gradient step selection: some theoretical notes

- Convergence is guaranteed[4] only for convex functions of ERA and bounded gradient under
  $\eta_t \to 0, \sum_{t=0}^{\infty} \eta_t = \infty, \sum_{t=0}^{\infty} \eta_t^2 < \infty$
  - E.g. $\eta_t = \frac{1}{t}$
  - Convergence speed is also $O(\frac{1}{t})$
- Steepest gradient descent method
  $R(w - \eta \nabla R(w)) \to min_\eta$ allows to find the optimal η^*

[4] Robbins, H. and Monro, S. (1951). "A stochastic approximation method"

AP

SGD gradient step selection

Means of SGD gradient step control

- Decrease (e.g., divide by $2 \dots 10$) every N iterations;
- Decrease (e.g., divide by $2 \dots 10$) every N iterations, when the value of ERA stopped to significantly change for the last K steps;
- Usage of "warm-up"[5] strategy;
- Usage of cosine[6] / linear[7] law (instead of discrete divisions) for gradient step update;
- Usage of cyclicality[8].

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

[5] Goyal, Priya, et al. "Accurate, large minibatch sgd: Training imagenet in 1 hour." 2017

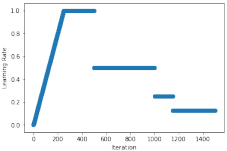
[6] Loshchilov, Ilya, and Frank Hutter. "Sgdr: Stochastic gradient descent with warm restarts." 2016

[7] Howard, Jeremy, and Sebastian Ruder. "Universal language model fine-tuning for text classification." 2018

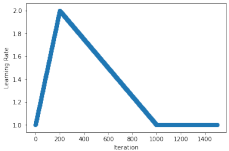
[8] Smith, Leslie N. "Cyclical learning rates for training neural networks." 2017

SGD gradient step selection illustration

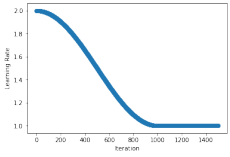
Warm-up



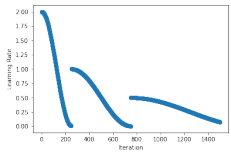
Triangular



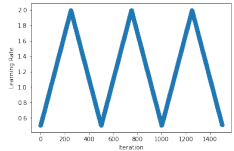
Cosine



Cosine + division



Cycle



About overfitting

Overfitting causes

- Small training set / large number of weights;
- Large number of object's features;
- Non-informative (noisy / dependent) object's features.

Overfitting observation

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

Overfitting reduction

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

MAP

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:

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

AP

A probabilistic view on regularization

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

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 σ, then

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

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

AP

L_2-regularization and SGD

Consider the squared penalty on weights norm inside ERA optimization:

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

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 τ

- Large value of τ – bigger penalty on overfitting (but the convergence if 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 $\sigma : R \to [0, 1]$ so as $\sigma(z) = \frac{1}{1+e^{-z}}$.

Logistic regression

Classification model for two classes $(Y = -1, +1)$ where probability of positive[9] 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: $\sigma(g(x, w)) \geq 1/2 \Leftrightarrow \langle w, x \rangle \geq 0$

[9] **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 $LL(w, X^m) = \log \prod_{i=1}^{m} p(x_i, y_i) \to \max_w$:

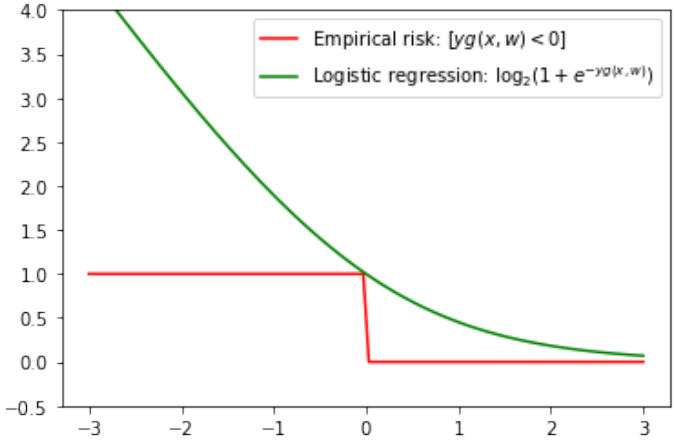
- $LL(w, X^m) = \sum_{i=1}^{m} \log \sigma(g(x_i, w)y_i) + const(w) \to \max_w$

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

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

AP

Logistic loss function vs ER



The correct ER approximation.

AP

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[10] 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**?

[10] **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

- linear classifier[11] $a(x, w) = \arg\max_{c \in Y} g(x, w^c)$, 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:

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

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

It means that function $SoftMax : \mathbb{R}^{|Y|} \to \mathbb{R}^{|Y|}$ 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.

[11] **Exercise**: prove it!

One-hot encoding and its entropy

- Define $p = (p_1, \dots, p_n), \quad \sum_{i=1}^n p_i = 1, 0 \leq p_i \leq 1.$
- **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: $p_i \log p_i = \lim_{x \to 0} x \log x = \lim_{x \to 0} \frac{\log x}{1/x} = \lim_{x \to 0} \frac{(\log x)'}{(1/x)'} = \lim_{x \to 0} \frac{1/x}{-1/x^2} = -\lim_{x \to 0} x = 0$
- All cases are resolved: $H(p) = -\sum_{i=1}^n p_i \log p_i = 0.$

AP

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

AP

Gibbs Inequality (1)

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

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 $\sum_{i \in I} p_i \log q_i - \sum_{i \in I} p_i \log p_i = \sum_{i \in I} p_i \log \frac{q_i}{p_i} \leq \sum_{i \in I} p_i (\frac{q_i}{p_i} - 1) = \sum_{i \in I} q_i - \sum_{i \in I} p_i \leq 1 - \sum_{i \in I} p_i = 1 - 1 = 0$

AP

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 $p_i = q_i = 0, \quad i \notin I$. ■

Corollary 1. Maximal entropy for $p = (p_1, \ldots, p_n), \quad \sum_{i=1}^n p_i = 1, 0 \leq p_i \leq 1$ 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 $p = (p_1, \ldots, p_n), \quad \sum_{i=1}^n p_i = 1, 0 \leq p_i \leq 1$: $0 \leq H(p) \leq \log n$.

AP

Kullback-Leibler divergence

Definition

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

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

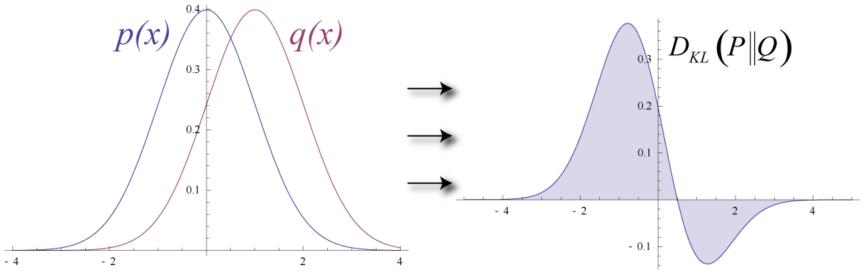
Kullback-Leibler divergence

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

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. ■

AP

Kullback-Leibler divergence

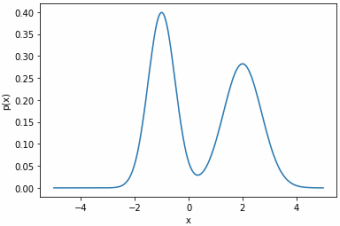


Kullback-Leibler: forward vs reverse divergence

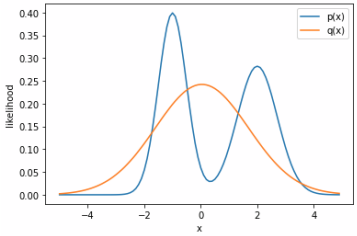
- Suppose we have a distribution p, and the distribution we'd like to optimize in relation to it is q
- **Forward KL** divergence target: minimize $D_{KL}(p||q)$
- **Reverse KL** divergence target: minimize $D_{KL}(q||p)$
- Minimizing Forward D_{KL} means minimize the discrepancy in the regions where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

AP

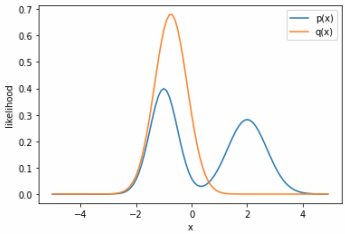
Forward vs reverse KL divergence illustration[12]



Initial distribution p



Optimizing in relation to forward KL



Optimizing in relation to reverse KL

Analogies:

1. Forward KL: maximize recall
2. Reverse KL: maximize precision

[12] 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?

AP

Metric based on Jensen–Shannon divergence (1)

Jensen–Shannon divergence

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

Metric definition

Function $d : X \times X \to \mathbb{R}$ 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.

AP

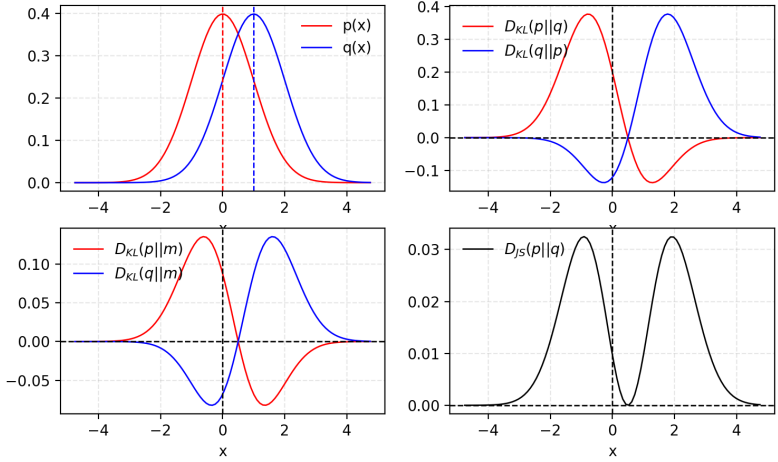
Metric based on Jensen–Shannon divergence (2)

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[13] about the correct proof. ■

[13] Endres D. M., Schindelin J. E. "A new metric for probability distributions". 2003.

JS vs different types of KL divergence[14]



[14] lilianweng.github.io

Takeaway notes

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; sqrt(Jensen-Shannon divergence) is
7. Minimization of CE $\Leftrightarrow$ KL $\Leftrightarrow$ making two distributions (ground truth and predicted) closer

AP

Thank you!