Machine learning

Lecture 6: Bayesian methods of machine learning.

Aleksei Platonov
DingTalkID: aplatonov

27 April 2020
HDU

- Basics of probability theory
- Probabilistic view to machine learning task
- Non-parametric density estimation
- Maximum likelihood principle

Basic concepts of probability theory

- **Outcomes**. The set of **elementary** events could be produced by an **experiment**.

$$\begin{aligned} \Omega_{dice} &= \{1, 2, \ldots, 6\},\ \Omega_{temp} = [-40; 40], \\ \Omega_{pixel} &= \{(x, y) : x, y \in [-100, 100]\} \end{aligned}$$

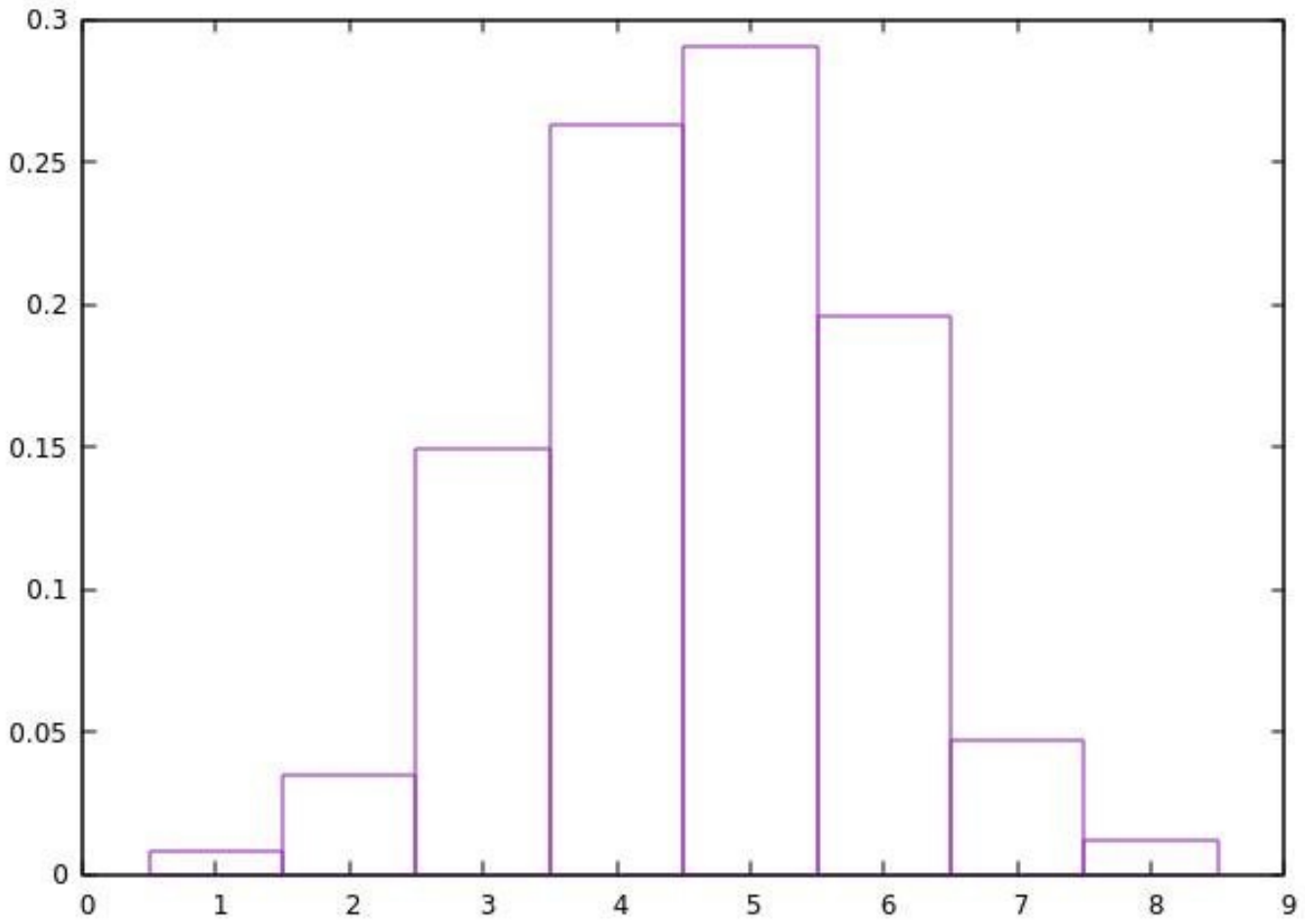
- **Probability distribution function**. The function binds **random variable** value with its **probability**.

$$\forall x \in \Omega : F(x) \in [0, 1], \quad \sum_{x \in \Omega} F(x) = 1$$

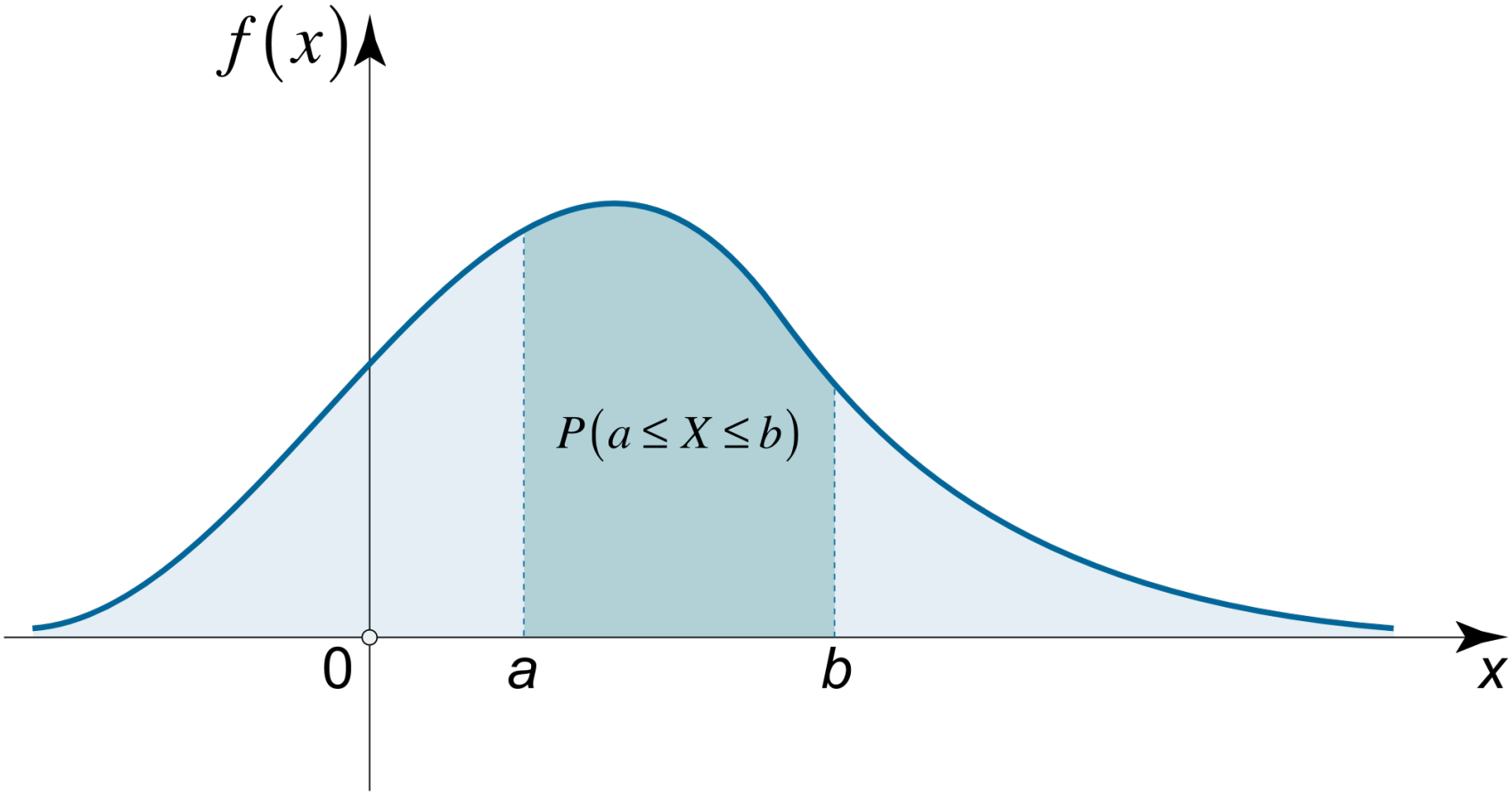
- **Event**. The subset in Ω

$$P(E) = \sum_{x \in E} F(x),\ E \subseteq \Omega$$

Histogram



Probability density function



Conditional probabilities

- If two events or more are independent of each other, then:

$$P(x_1, x_2, \ldots, x_n) = P(x_1)P(x_2)\ldots P(x_n),$$

for example two coin tosses

- But some events can't be independent of each other:

$$P(x_1, x_2) = P(x_2|x_1)P(x_1) \neq P(x_1)P(x_2),$$

where $P(x_2|x_1)$ is a conditional probability, i.e. probability of event x_2 if event x_1 was observed.

Bayesian Theorem

It is interesting that:

$$P(A|B)P(B) = P(A,B) = P(B|A)P(A),$$

so, we can transform this equation into:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}.$$

If B depends on N events, then $P(B) = \sum_{i=1}^{N} P(A_i)P(B|A_i)$ and for Bayesian Theorem of complex event:

$$P(A_j|B) = \frac{P(A_j)P(B|A_j)}{\sum_{i=1}^{N} P(A_i)P(B|A_i)}$$

Bayesian Theorem in reasoning

- $P(B)$, $P(A_j)$ is apriori information about events B and A_j.
- $P(B|A_j)$ is conditional probability of event B if A_j is detected.
- $P(A_j|B)$ is a probability of hypothesis A_j if B was.
- Thus we can estimate probabilities of causes of some event. If we see the event B and we know the probabilities distributions $P(A_j)$ and $P(B|A_j)$ then we can estimate the probabilities of reasons of event B - $P(A_j|B)$.

ML task from probability theory point of view

- The task statement is the same: $X^l = (x_i, y_i)_{i=1}^l$ - the training sample. X - the set of objects represented by vector of features' values, Y - the set of possible answers.
- Probabilistic view of the task: let's suppose that there is probabilities distribution $p(x, y)$ in the space $X \times Y$ of objects and answers pairs.
- If $X^l = (x_i, y_i)$ is independent and identically-distributed sample (i.i.d) we need to find a classifier $a : X \to Y$ that **minimizes the probability of error**.

ML task from probability theory point of view

So, we could machine learning terms into probabilistic representation:

- $P(y)$ - the apriori information about classes distribution.
- $P(x|y)$ - the function of likelihood of class y.
- $P(y|x)$ - the a posteriori probability of class y.
- The principle of maximum a posteriori probability:

$$a(x) = arg \max_{y \in Y} P(y|x) = arg \max_{y \in Y} P(y)P(x|y).$$

Do you remember Bayesian theorem?

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}.$$

Risk assessment using probabilities

- If we use $a : X \to Y$ then we split all set X into non-overlapped regions. Then we can define a subsets corresponding to classes:

$$A_y = \{x \in X | a(x) = y\}, y \in Y$$

- So, the error could be defined as $a(x) = y$, i.e. $x \in A_y$ but $y(x) = y^{'}$ and $y \neq y^{'}$.
- Then the probability of error:

$$P(A_y, y^{'}) = \int\limits_{A_y} p(x, y^{'}) dx$$

Risk assessment using probabilities

- If we add the cost of error $\lambda_{yy'}$ then we can define the classification risk:

$$R(a) = \sum_{y \in Y} \sum_{y' \in Y} \lambda_{yy'} P(A_y, y')$$

- Theorem: if $\lambda_{yy'} = \lambda_{y'y} = \lambda_y$ and $\lambda_{yy} = 0$ then the minimal value of the classification risk has Bayesian classifier using principle of maximum a posteriori probability:

$$a(x) = arg \max_{y \in Y} P(y|x) = arg \max_{y \in Y} \lambda_y P(y) P(x|y).$$

How to build a classifier using probability theory

1. 1st problem: we have $X^l = (x_i, y_i)_{i=1}^l$ and we need to find a **probabilistic model** of task, i.e. we should estimate $P(y)$ and $P(x|y)$.

2. 2nd problem: we have $P(y)$ and $P(x|y)$. How to create a classifier $a : X \to Y$ minimizes the probability of error R(a)? **We've already done it!**

Important notion: we must remember that Bayesian classifier minimizes classification error iff $P(y)$ and $P(x|y)$ is real distributions (but in a real task we usually know just approximations of them).

Statistical approach to find model's parameters

How to estimate $P(y)$ and $P(x|y)$?

- For $P(y)$ the task is very simple. Let's define $X_y = \{x_i \in X | y_i = y\}$ then:

$$P(y) \approx \frac{|X_y|}{l}$$

  It's just a discrete probabilities distribution.

- How to estimate $P(x|y)$? Let's start to answer from the one-dimensional task with one class label:
  Input: $X_y^m = X = \{x_1, x_2, \ldots, x_m\}$ is i.i.d.
  The goal: Define $\overline{P}(x) \to P(x)$ if $m \to \infty$

Naive Bayesian Classifier

Let's start from simple assumption:

- Features in an object description are independent of each other, i.e. feature values don't correlate each other:

$$P(x|y) = P(f_1(x), \ldots, f_n(x)|Y) = \\ = P(f_1(x)|y)P(f_2(x)|y)\ldots P(f_n(x)|y)$$

- We know that $log_{10}(xy) = log_{10}(x) + log_{10}(y)$ and the log is the monotonically increasing function, so:

$$a(x) = arg\max_{y \in Y}(\lambda_y P(y)P(x|y)) = arg\max_{y \in Y}(log_{10}[\lambda_y P(y)P(x|y)])$$

Thus, we have much more simpler task of one-dimensional probability distributions estimations:

$$a(x) = arg\max_{y \in Y}\left(log_{10}\lambda_y P(y) + \sum_{j=1}^{n} log_{10}P(f_j(x)|y)\right)$$

Basic methods

- The parametric approach to probability densities estimation:

$$\overline{P}(x) = \phi(x, \theta)$$

- The mixin of probability densities (the topic of the next lecture):

$$\overline{P}(x) = \sum_{j=1}^{k} w_j \phi(x, \theta_j), k \ll m$$

- The **non-parametric approach** to probability densities estimation:

$$\overline{P}(x) = \sum_{i=1}^{m} \frac{1}{mV(h)} K(\frac{\rho(x, x_i)}{h})$$

(Hm... It seems that we're already seen it before...)

Probability density estimation

So, we need to estimate $p(x)$:

- The discrete case - histogram:

$$p(x) = \frac{1}{m} \sum_{i=1}^{m} [x_i = x],$$

where m - the size objects' set corresponding to one class.

- The continuous case ($X \subseteq \mathbb{R}$):

$$p_h(x) = \frac{1}{2h} \frac{1}{m} \sum_{i=1}^{m} [|x - x_i| < h],$$

but we've seen it somewhere...

Local non-parametric Parzen-Rosenblatt method

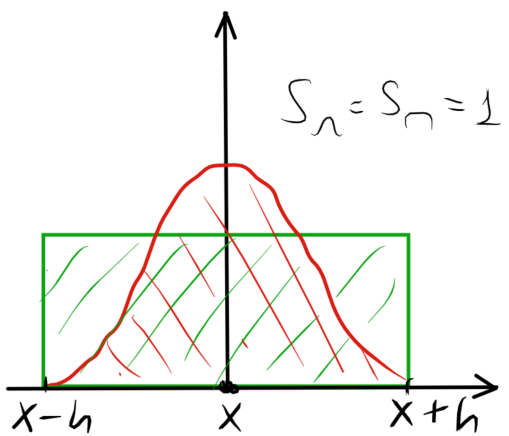
If we rewrite the last expression:

$$p_h(x) = \frac{1}{2h}\frac{1}{m}\sum_{i=1}^{m}\left[\frac{|x - x_i|}{h} < 1\right] = \frac{1}{2h}\frac{1}{m}\sum_{i=1}^{m} K\left(\frac{x - x_i}{h}\right),$$

we'll get the same function as we've seen in k-nearest neighbors method, where $K(r)$ is the kernel.

For a density function $K(r)$ must be:

- non-increasing (on the right side) and positive;
- symmetric function;
- normalized function, i.e. $\int_{-\infty}^{+\infty} K(r)dr = 1.$



Multidimensional generalization

1. If objects are described by numerical vectors with features $f_i : X \to \mathbb{R}, j = 1 \dots n$:

$$p_h(x) = \frac{1}{m} \sum_{i=1}^{m} \prod_{j=1}^{n} \frac{1}{h_j} K\left(\frac{f_j(x) - f_j(x_i)}{h_j}\right).$$

2. If we have a distance function $\rho : X \times X \to \mathbb{R}$ (it is kNN!):

$$p_h(x) = \frac{1}{mV(h)} \sum_{i=1}^{m} K\left(\frac{\rho(x, x_i)}{h}\right),$$

Please note: V(h) is the volume of a sphere with radius equals to h in multidimensional space. It mustn't depend on x_i.

Parzen-window based classifier

We can use this non-parametric approach to make a classifier:

- The density function estimation for class $y \in Y$:

$$p_h(x|y) = \frac{1}{l_y V(h)} \sum_{i:y_i=y} K\left(\frac{\rho(x, x_i)}{h}\right)$$

- Parzen's window-based Bayess classifier:

$$a(x_i, X^l, h) = arg \max_{y \in Y} \lambda_y \frac{P(y)}{l_y} \sum_{i:y_i=y} K\left(\frac{\rho(x, x_i)}{h}\right),$$

note that whether $V(h)$ doesn't depend on y we can remove it from the argmax expression.

Choosing the metric function

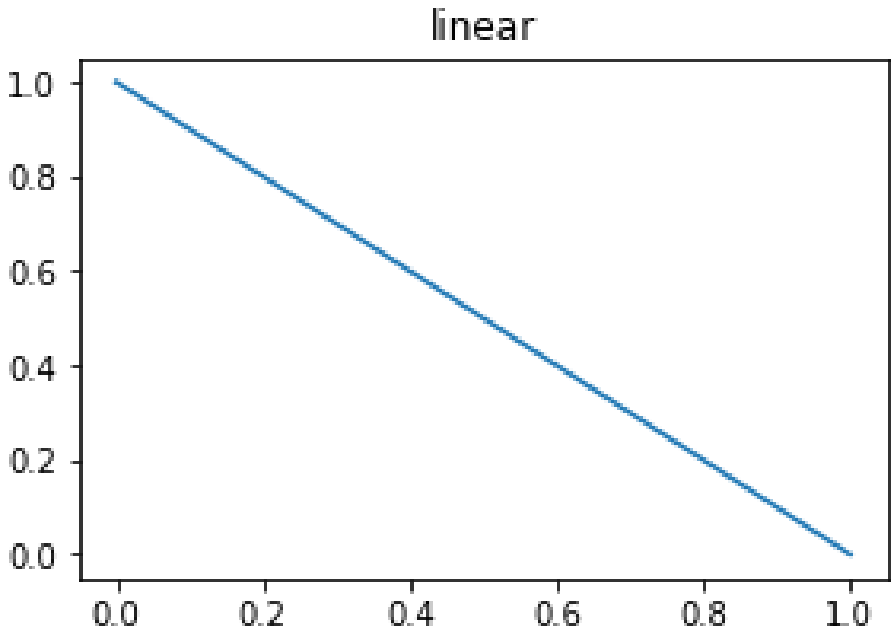
One of the possible variants is weighted Minkowski's distance:

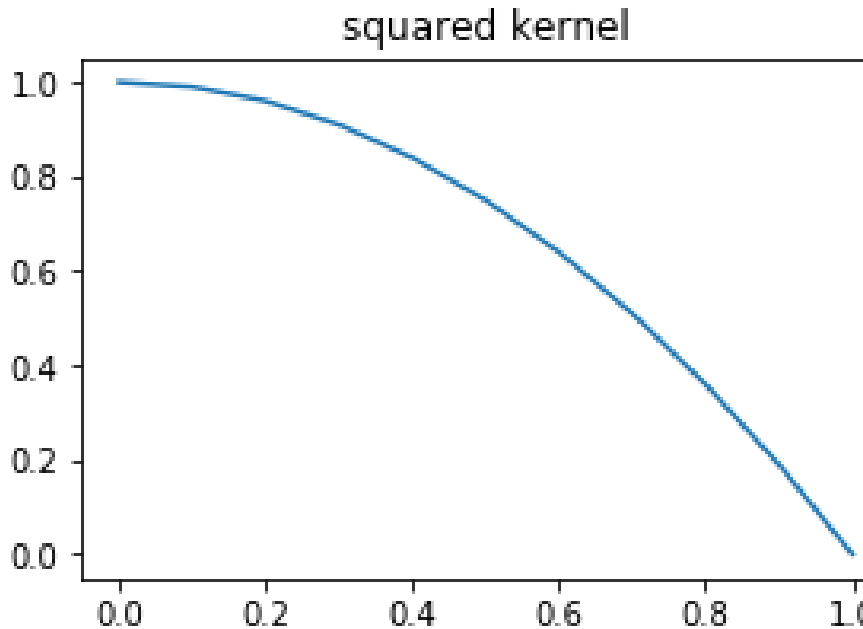
$$\rho(x, x') = \left(\sum_{j=1}^{n} w_j |f_j(x) - f_j(x')|^p \right)^{1/p},$$

where w_j are the weights of features, $p > 0$. In particular, if $w_j = 1$ and $p = 2$ we'll have Euclidean distance.
The vector of weights $[w_j]_{j=1}^n$ can:

- normalizes features' values;
- estimates features' importance;
- selects features (which $w_j = 0$?).

Choosing the kernel





- Linear kernel: $K(\overline{\rho}) = (1 - \overline{\rho}) \cdot [\overline{\rho} \leq 0]$
- Squared kernel: $K(\overline{\rho}) = \frac{3}{4}(1 - \overline{\rho}^2) \cdot [\overline{\rho} \leq 0]$

Parametric probability density models

- The **parametric approach** to probability densities estimation:

$$\overline{P}(x) = \phi(x, \theta)$$

- The **mixin of probability densities** (the topic of the next lecture):

$$\overline{P}(x) = \sum_{j=1}^{k} w_j \phi(x, \theta_j), k \ll m$$

- The non-parametric approach to probability densities estimation:

$$\overline{P}(x) = \sum_{i=1}^{m} \frac{1}{mV(h)} K(\frac{\rho(x, x_i)}{h})$$

Maximum likelihood principle

If we have a probabilistic model with parameter: $\phi(x, \theta)$ we can estimate a likelihood of the training data:

$$L(\theta, X^m) = \phi(x_1, \theta) \cdot \phi(x_2, \theta) \cdot \dots \cdot \phi(x_m, \theta) = \prod_{i=1}^{m} \phi(x_i, \theta)$$

$L(\theta, X^m)$ - the probability of the i.i.d training set observation estimated by model ϕ.

But, if we observe this training set, then the probability of it should be as high as possible:

$$L(\theta, X^m) \to \max_{\theta}$$

- this is maximum likelihood principle.

Log-likelihood and gradient descent

- We know that log-function is a monotonically increasing function, hence:

$$arg\max_{\theta} L(\theta.X^m) = arg\max_{\theta} \sum_{i=1}^{m} ln(\phi(x_i, \theta))$$

- If ϕ is differentiable then maximum will be in this point:

$$\frac{\partial}{\partial\theta} \sum_{i=1}^{m} ln(\phi(x_i, \theta)) = 0$$

- Replacing $L(\theta, X^m) \to max$ to $-L(\theta, X^m) \to min$ we can use already known gradient descent method! (remember Logistic regression)

Multidimensional Gaussian model

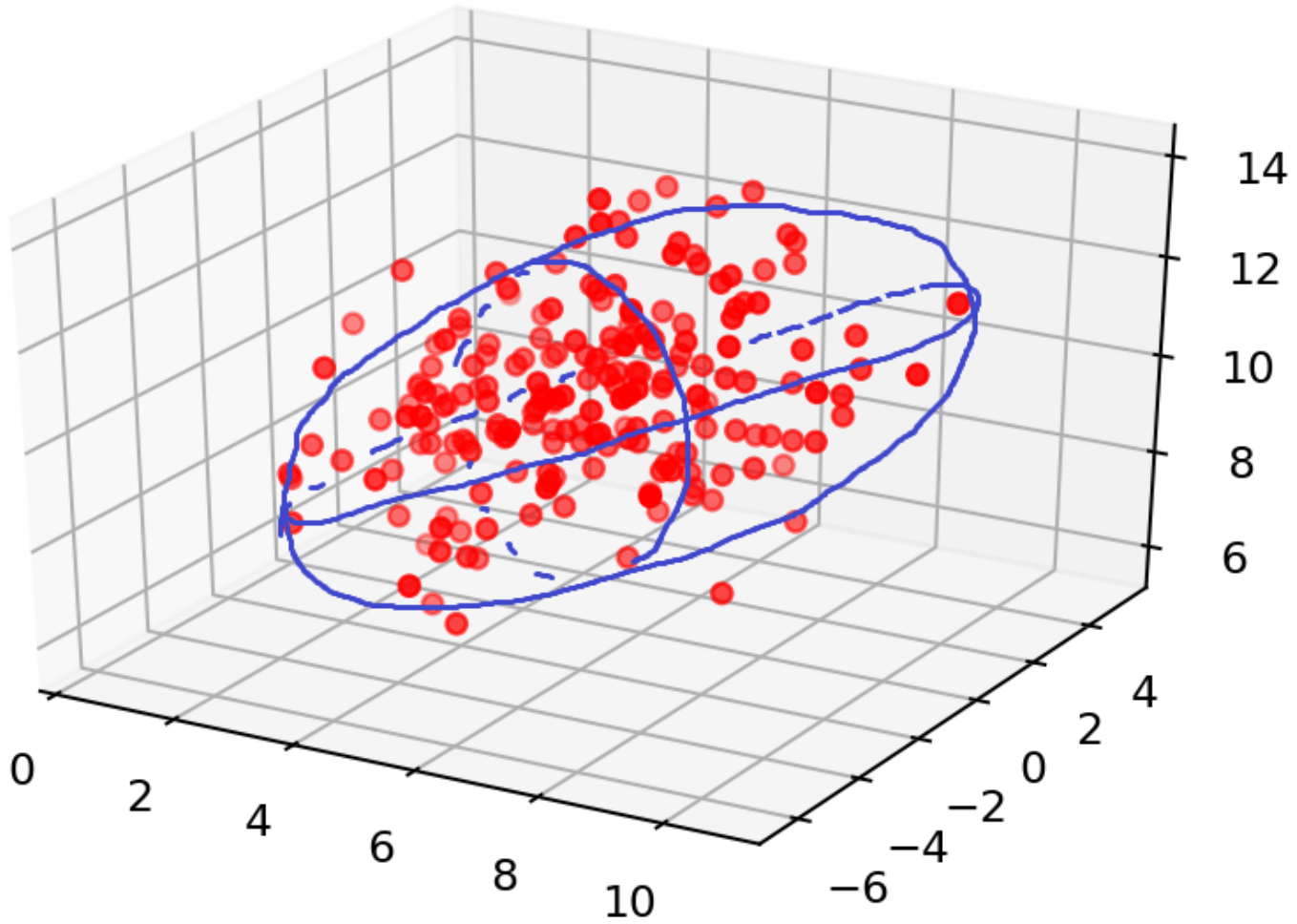
- Suppose that the features description vector consists of only numeric values: $X = \mathbb{R}^n$
- Then we can apply the simplest and well known Gaussian model:

$$P(x|y) = \frac{1}{\sqrt{2\pi^m det \sum_y}} e^{-\frac{1}{2}(x-\mu_y)^T \sum_y^{-1}(x-\mu_y)},$$

where

- $\mu_y = \frac{1}{l_y} \sum_{i:y_i=y} x_i$ - the mean value of object vector x for class y
- $\sum_y = \frac{1}{l_y} \sum_{i:y_i=y} (x_i - \mu_y)(x_i - \mu_y)^T$ - the covariance matrix, the analogue of dispersion.

Geometric representation



Overfitting of Gaussian model

Models based on Gaussian distributions tend to overfitting. Here are the reasons for it:

- if $l_y < n$ then $\sum_y$ is a degenerated matrix (l_y - the number of objects of class y, n - the number of features);
- even though if $l_y > n$, lower values of l_y lead $\sum_y$ to be unstable;
- μ_y and $\sum_y$ are unstable towards outliers in data, i.e.

$$\mu([0,1,2,0,2,1,1]) = 1, \textit{but } \mu([0,1,2,0,2,1,10]) \approxeq 52.286$$

- if classes don't distribute as the normal distribution, then the model will be too overfitted.

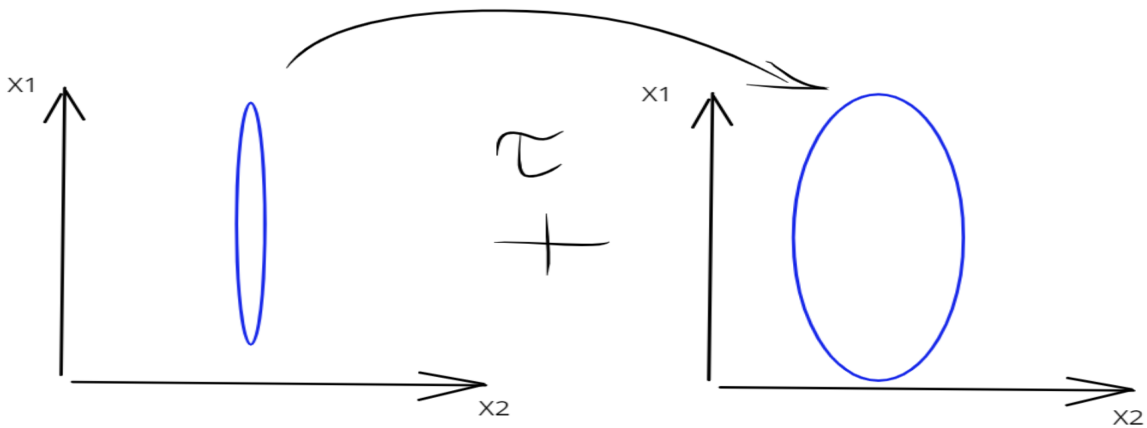
Regularization of covariance matrix

But we can fix some problems with the covariance matrix in case of overfitting.

The idea: transform the covariance matrix $\sum$ saving eigenvectors but changing eigenvalues by constant τ

$$\left(\sum + \tau I_n\right) v = \lambda v \tau v = (\lambda + \tau) v,$$

where v is an eigenvector of $\sum$, λ is an eigenvalue.



Regularization of covariance matrix

The previous idea leads $\sum$ to more diagonal shape, we can diagonalize the matrix explicitly, for all $i, j = 1 \dots n, i \neq j$ (indexes of features):

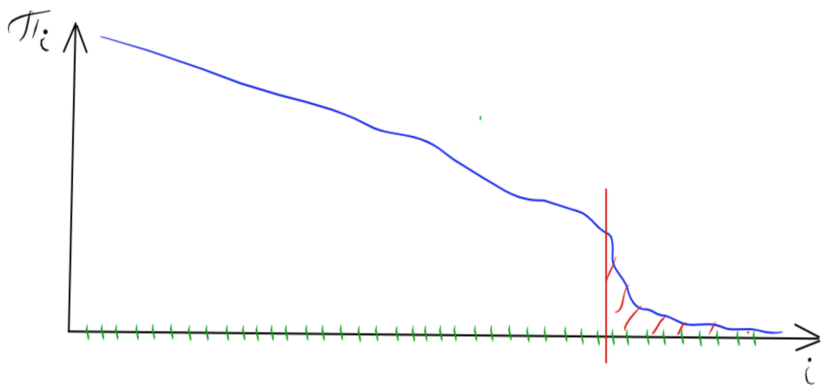
1. compute correlation coefficient $r_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}}$;
2. compute a value of the Student's distribution $T_{ij} = \frac{r_{ij}\sqrt{n-2}}{\sqrt{1-r_{ij}^2}}$;
3. if $|T_{ij}| < t_{1-\alpha/2}$ then set $\sigma_{ij} = 0$.

Note: if we continue this procedure and set all non-diagonal σ_{ij} to 0 it will mean that all features are independent to each other, i.e. we will have naive Bayes classifier.

Outliers problem

Outliers are a huge problem of methods based on Gaussian distribution, but we can fix it censoring them using the same model:

1. Train a model $a(x)$ and compute the likelihood for each element of the training set.
2. Sort all elements by likelihood and remove elements with too low value of the likelihood. Re-train model and go-to 1st step.



Conclusions

- We observed the main principles of the probability theory.
- We studied how to represent machine learning task from the Bayesian point of view.
- We binded the kNN model with non-parametric algorithms of distributions estimation.
- We observed the simplest parametric approach to distributions estimation using multidimensional Gauss distribution.

We studied how to fix problems of overfitting the model based on Gauss distribution but didn't answer the question: what if our data doesn't fit this distribution? This is the topic of the next lecture!