Machine learning

Lecture 7: Bayesian methods of machine learning (part 2).

Aleksei Platonov
DingTalkID: aplatonov

6 May 2020
HDU

- We'll recall the last material.
- We'll review the parameterized approach from linear models point of view.
- We'll investigate EM-algorithm to estimate parameters of probabilistic models mixture.

The problem statement

- **The machine learning task statement**: we have $X^l = (x_i, y_i)_{i=1}^l$ - the training sample. X - the set of objects represented by vector of features' values, Y - the set of possible answers. The goal is to find the decision function $a : X \to Y$.
- **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**.

The optimal Bayesian classifier

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

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

where $PA_y, y^{'}$ - the probability of classification error, when $a(x) = y$ but $y(x) = y^{'}$

- 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:

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

Estimation of a probability density

We have $X^l = (x_i, y_i)_{i=1}^l$ and we need to find a **probabilistic model** of task, we need to estimate $P(y)$ and $P(x|y)$ - a priori information about classes distribution and likelihood functions of objects.

- For $P(y)$ it is a simple task:

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

  where $X_y = \{x_i \in X | y_i = y\}$

- For likelihood function $P(x|y)$ we observed several approaches.

Methods of probability densities estimation

- The parametric approach to probability densities estimation:

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

- The mixin of probability densities:

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

- The non-parametric approach to probability densities estimation:

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

Multidimensional Gaussian model

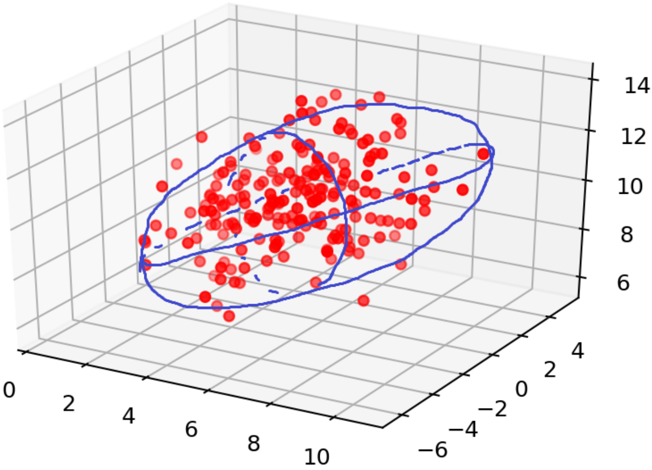
- Suppose that the features description vector consists of only numeric values: $X = \mathbb{R}^n$
- Then we can apply the simplest and well known Gaussian model:

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

where

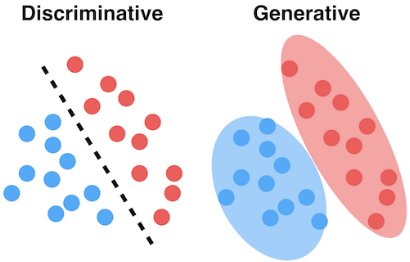
- $\mu_y = \frac{1}{l_y} \sum_{i:y_i=y} x_i$ - the mean value of object vector x for class y
- $\sum_y = \frac{1}{l_y} \sum_{i:y_i=y} (x_i - \mu_y)(x_i - \mu_y)^T$ - the covariance matrix, the analogue of dispersion.

Geometric representation



Discriminative models

- Generative models try to model a class. They defines a shape of class, density of it, etc. (**Gaussian model, kNN**)
- Discriminative models try to define hyperplanes or non-linear shapes in features' vector space to separate classes (**Linear models, decision trees**).

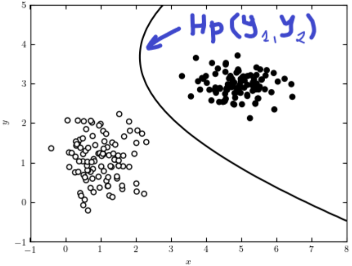


Gaussian model as a discriminative model

But if we know shapes of classes then we can find separating surfaces:

$$Hp(y_1, y_2) = \{x \in X : \lambda_{y_1} P(y_1) P(x|y_1) = \lambda_{y_2} P(y_2) P(x|y_2)\},$$

where λ_i - the cost of error.

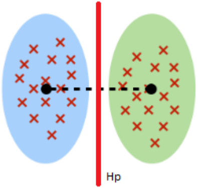


For Gaussian model the separating surface is a quadratic function. But there is interesting observation...

Fisher's linear discriminant

- Let's suppose that all covariance matrices are the same: $\sum_{y_1} = \sum_{y_2} = \dots = \sum$. Then the orientations of all classes' clouds will be the same.
- Thus: the separating surface could be hyperplane (i.e. linear function). This is the linear classifier! Indeed:

$$
\begin{aligned}
a(x) &= arg \max_{y \in Y} \lambda_y P(y) P(x|y) \\
&= arg \max_{y \in Y} ln\left(\lambda_y P(y) P(x|y)\right) = |P(x|y) \leftarrow gaussian| \\
&= arg \max_{y \in Y} \left(ln(\lambda_y P(y)) - \frac{1}{2} \mu_y^T \Sigma^{-1} \mu_y + x^T \Sigma^{-1} \mu_y \right) \\
&= |\alpha_y \leftarrow \Sigma^{-1} \mu_y, \beta_y \leftarrow ln(\lambda_y P(y)) - \frac{1}{2} \mu_y^T \Sigma^{-1} \mu_y| \\
&= |\alpha_y, \beta_y - const_1(x)| = arg \max_{y \in Y} (x^T \alpha_y + \beta_y)
\end{aligned}
$$



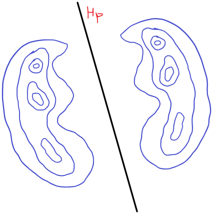
Advantages and disadvantages

+ Fisher's linear discriminant is a quite simple linear model. We need just calculate Σ and μ_y for each y;
+ It's much more stable than Gaussian model with quadratic discriminative function;
- if $l_y < n$ then Σ 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 Σ to be unstable;

Exponential family of probability distributions

If we can consider Fisher's model as a linear model based on Gaussian distribution, could we use any other exponential distribution for the linear model?

$$P(x|y) = e^{c_y(\delta)<\theta_y,x>+b_y(\delta,\theta_y)+d(x,\delta)},$$



where

- $\theta_y \in \mathbb{R}^n$ - the shift parameter (analogue of μ);
- δ - the variance parameter (analogue of Σ);
- b_y, c_y, d - arbitrary numerical functions.

Bayes classifier as linear model

An optimal Bayesian classifier for two classes $Y = \{-1, +1\}$:

$$
\begin{aligned}
a(x) &= arg \max_{y \in Y} \lambda_y P(y) P(x|y) \\
&\equiv sign(\lambda_{+1} P(+1) P(x|+1) - \lambda_{-1} P(-1) P(x|-1)) \\
&= sign\left(\frac{P(+1)P(x|+1)}{P(-1)P(x|-1)} - \frac{\lambda_{-1}}{\lambda_{+1}}\right)
\end{aligned}
$$

So, if we proof that this classifier could be represented as a linear classifier then all exponential probabilities models could be linear discriminative models.

Bayes classifier as linear model [proof step 1]

Let's substitute exponential model:

$$P(x|y) = e^{c_{\pm1}(\delta)<\theta_{\pm1},x>+b_{\pm1}(\delta,\theta_{\pm1})+d(x,\delta)}$$

into Bayesian binary classifier:

$$a(x) = sign\left(\frac{P(+1)P(x|+1)}{P(-1)P(x|-1)} - \frac{\lambda_{-1}}{\lambda_{+1}}\right) = sign\left(ln\frac{P(+1)P(x|+1)}{P(-1)P(x|-1)} - ln\frac{\lambda_{-1}}{\lambda_{+1}}\right),$$

then

$$\begin{aligned} ln\frac{P(+1)P(x|+1)}{P(-1)P(x|-1)} &= ln(P(x|+1)) - ln(P(x|-1)) + ln\frac{P(+1)}{P(-1)} = |ln(e^x) = x| = \\ &= c_{+1}(\delta)\left\langle\theta_{+1}, x\right\rangle + b_{+1}(\delta,\theta_{+1}) + d(x,\delta) + ln(P(+1)) - \\ &- c_{-1}(\delta)\left\langle\theta_{-1}, x\right\rangle - b_{-1}(\delta,\theta_{-1}) - d(x,\delta) - ln(P(-1)) = \\ &= |c_{-1}(\delta) = c_{+1}(\delta)| = c(\delta)\left\langle\theta_{+1} - \theta_{-1}, x\right\rangle + \\ &+ b_{+1}(\delta,\theta_{+1}) - b_{-1}(\delta,\theta_{-1}) + ln\frac{P(+1)}{P(-1)} \rightarrow \left\langle w, x\right\rangle + \beta \end{aligned}$$

Bayes classifier as linear model [proof step 2]

1. if $ln\frac{P(+1)P(x|+1)}{P(-1)P(x|-1)} = \langle w, x\rangle + const$ then $\frac{P(+1|x)}{P(-1|x)} = e^{\langle w,x\rangle}$
2. Using formula of total probability $P(+1|x) + P(-1|x) = 1$ we have:

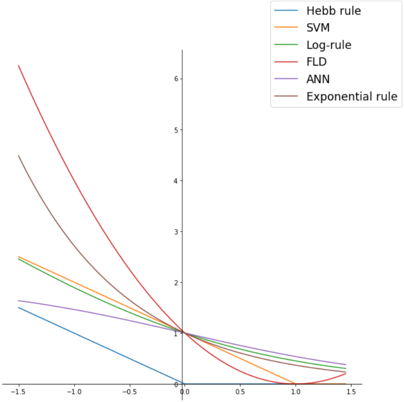
$$P(+1|x) = \frac{1}{1 + e^{-\langle w,x\rangle}}, P(-1|x) = \frac{1}{1 + e^{\langle w,x\rangle}}$$

3. Thus, $P(y|x) = \sigma(\langle w, x\rangle\, y)$, where $y \in -1, +1$ and $\sigma(z) = \frac{1}{1+e^{-z}}$

We've just inferred the logistic regression model! So, the logistic regression model is a Bayesian classifier with a linear separating hyperplane.

Loss functions for linear classifiers

- Hebb rule: $H(M) =$ if $M < 0$ then $-M$ else 0
- SVN rule: $SVM(M) =$ if $M < 1$ then $1 - M$ else 0
- Log-rule: $L(M) = ln(1 + e^{-M})$
- FLD: $FLD(M) = (1 - M)^2$
- ANN: $ANN(M) = \frac{2}{1+e^M}$
- Exponential rule: $E(M) = e^{-M}$



The logarithmic loss function of logistic regression

We could substitute sigmoid into maximum of the likelihood principle:

$$L(w) = ln \prod_{i=1}^{l} P(x_i, y_i) \to \max_w \Rightarrow$$

$$|P(x_i, y_i) = P(y_i)P(x_i|y_i) \leftarrow \sigma(\langle w, x_i \rangle \, y_i) \cdot const(w)| \Rightarrow$$

$$L(w) = \sum_{i=1}^{l} ln\left(\sigma(\langle w, x_i \rangle \, y_i)\right) + const(w) \to \max_w$$

It's an equivalent of minimization of $Q(w)$:

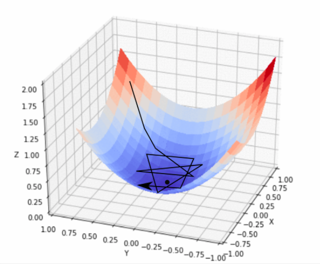
$$Q(w) = \sum_{i=1}^{l} ln\left(1 + e^{-\langle w, x_i \rangle y_i}\right) = \sum_{i=1}^{l} {\color{red} ln\left(1 + e^{-M_i(w)}\right)} \to \min_w$$

Training of logistic regression

The first-order method. the gradient descent algorithm:

$$w^{t+1} = w^t + \eta_t y_i x_i (1 - \sigma(\langle w, x_i \rangle)),$$

where η_t - the gradient descent step, $\sigma(\langle w, x_i \rangle) = P(y_i | x_i)$



The task of probabilistic models mixture training

Let's suppose that the target probabilistic distribution looks like this:

$$P(x) = \sum_{j=1}^{k} w_j P_j(x, \theta_j) \text{ and } \sum_{j=1}^{k} w_i = 1, w_j \geq 0,$$

where $P_j(x, \theta_j)$ - the j-th component of mixture (likelihood of x), w_j - the probability of j-th component, k - the number of components in a mixture.

So, we have two tasks:

- Taking $X^l = (x_i, y_i)_{i=1}^{l}$ and k how to find the parameters vector $\{w_1, \ldots, w_k, \theta_1, \ldots, \theta_k\}$?
- How to define k?

The maximum likelihood principle and EM-philosophy

Recall the main principle of probabilistic models:

$$L(\Theta) = ln \prod_{i=1}^{m} p(x_i) = \sum_{i=1}^{m} ln \left(\sum_{j=1}^{k} w_j p_j(x, \theta_j) \right) \to \max_{\Theta},$$

where $\sum_{j=1}^{k} w_i = 1, w_j \geq 0$

The problem: there is no analytical solving of this task.
Let's solve it using approximating iterative algorithm:

Algorithm 1 EM-algorithm

procedure $\mathrm{EM}(X^l, \Theta_0)$
  $\Theta_1 \leftarrow \Theta_0,\ t \leftarrow 1$
  while $|\Theta_t - \Theta_{t-1}| > \delta$ **do**
    $G \leftarrow$ E-step$(\Theta_t,\ X^l)$ ▷ Estimate hidden variables
    $\Theta_{t+1} \leftarrow$ M-step(Θ_t, G) ▷ Infer new Θ from G
    $t \leftarrow t + 1$
  return Θ

EM-algorithm for probabilistic distribution mixture

If the probabilistic model is a mixture of probability densities and $G = (g_{ij})$ is the hidden variables, then:

- E-step:
$$g_{ij} = \frac{w_j p_j(x_i, \theta_j)}{\sum_{s=1}^{k} w_s p_s(x_i, \theta_s)},$$
where $i = 1 \dots m$, $j = 1 \dots k$
- M-step:
$$\theta_j = arg\max_{\theta} \sum_{m}^{i=1} g_{ij} ln\left(p_j(x_i, \theta)\right), w_j = \frac{1}{m} \sum_{i=1}^{m} g_{ij},$$
where $j = 1 \dots k$

Probabilistic interpretation of hidden variables

What are hidden variables g_{ij}? It turned out that they have an interpretation:

$$g_{ij} = P(j|x_i) = \frac{P(j)P(x_i|j)}{P(x_i)} = \frac{w_j P_j(x_i, \theta_j)}{P(x_i)} = \frac{w_j P_j(x_i, \theta_j)}{\sum_{s=1}^{k} w_s P_s(x_i, \theta_s)},$$

obviously that $\sum_{j=1}^{k} g_{ij} = 1$

So, g_{ij} is a posterior information about j-th component for object x_i, the probability of j-th component for x_i

Inference of EM-algorithm. Karush–Kuhn–Tucker conditions.

There is a theorem that tells us if we have an optimization problem with restrictions

$$\begin{cases} f(x_1, \ldots, x_n) \to \min \\ g_i(x_1, \ldots, x_n) \geq 0, i = 1 \ldots k_1 \end{cases},$$

then this system could be solved using another system:

$$\begin{cases} \mathbb{L}(X) = f(X) + \sum_{i=1}^{k_1} \lambda_i g_i(X) \\ \nabla \mathbb{L}(X) = 0 \\ \lambda_i \geq 0 \\ \lambda_i g_i(X) = 0 \end{cases},$$

Inference of EM-algorithm. Main equations

So, we can rewrite the target of the maximum likelihood principle for probabilistic densities mixture using the Kuhn-Tukker theorem:

$$\mathbb{L}(\Theta) = \sum_{i=1}^{m} ln\left(\sum_{j=1}^{k} w_j p_j(x_i, \theta_j)\right) - \lambda\left(\sum_{j=1}^{k} w_j - 1\right)$$

and solve it:

$$\frac{\partial \mathbb{L}}{\partial w_j} = \sum_i \frac{p_j(x_i, \theta_j)}{p(x_i)} - \lambda = 0 \Rightarrow$$

$$\sum_i w_j \frac{p_j(x_i, \theta_j)}{p(x_i)} - \lambda w_j = \sum_j g_{ij} - \lambda w_j = 0 \Rightarrow$$

$$\sum_i \sum_j g_{ij} = \lambda \sum_j w_j \Rightarrow \left| \sum_j g_{ij} = 1, \sum_j w_j = 1 \right| \Rightarrow \lambda = m$$

Inference of EM-algorithm. Main equations

Using the equation $\lambda = m$ in the main system, from $\frac{\partial \mathbb{L}}{\partial w_j} = 0$ we can infer:

$$w_j = \frac{1}{m} \sum_{i=1}^{m} \frac{w_j p_j(x_i, \theta_j)}{p(x_i)} = \frac{1}{m} \sum_{i=1}^{m} g_{ij}$$

Also, we can infer θ_j:

$$\frac{\partial \mathbb{L}}{\partial \theta_j} = \sum_{i=1}^{m} \frac{w_j p_j(x_i, \theta_j)}{p(x_i)} \frac{\partial}{\partial \theta_j} ln\left(p_j(x_i, \theta_j)\right) = \frac{\partial}{\partial \theta_j} \sum_{i=1}^{m} g_{ij} ln\left(p_j(x_i, \theta_j)\right) = 0$$

It the equivalent of *argmax*:

$$\frac{\partial}{\partial \theta_j} \sum_{i=1}^{m} g_{ij} ln\left(p_j(x_i, \theta_j)\right) = 0 \equiv arg \max_{\theta} \sum_{i=1}^{m} g_{ij} ln\left(p_j(x_i, \theta_j)\right)$$

EM-algorithm

Algorithm 2 EM-algorithm

procedure $\mathrm{EM}(X^l, k, \delta, \Theta = (w_j, \theta_j)_{j=1}^k)$
    Init g_{ij} randomly
    do
        for $i = 1 \dots m, j = 1 \dots k$ **do** ▷ E-Step
            $g_{ij}^0 \leftarrow g_{ij}$
            $g_{ij} \leftarrow \frac{w_j p_j(x_i,\theta_j)}{\sum_{s=1}^k w_s p_s(x_i,\theta_s)}$
        for $j = 1 \dots k$ **do** ▷ M-Step
            $w_j \leftarrow \frac{1}{m} \sum_{i=1}^m g_{ij}$
            $\theta_j \leftarrow arg \max\limits_{\theta} \sum_{i=1}^m g_{ij} ln\left(p_j(x_i, \theta_j)\right)$
    while $\max\limits_{i,j} |g_{ij} - g_{ij}^0| > \delta$
    return $(w_j, \theta_j)_{j=1}^k$

EM-algorithm for gaussian distributions mixture (GMM)

Let's remember that finding the *arg* $\max_{\theta} p_j(x_i, \theta_j)$ for **Gaussian** model corresponds to finding mean and variance values. Then we can find an optimal solution for mixture of Gaussian distributions:

$$\begin{cases} \mu_{yjd} = \frac{1}{l_y w_{yj}} \sum\limits_{i:y_i=y} g_{yij} f_d(x_i) \\ \sigma^2_{yjd} = \frac{1}{l_y w_{yj}} \sum\limits_{i:y_i=y} g_{yij} (f_d(x_i) - \mu_{yjd})^2 \end{cases},$$

where l_y - the number of objects of class y, d - the index of d-th feature value, μ_{yjd} - the mean value of j-th component for d-th feature for a class with label y, $sigma^2_{yjd}$ - variance.

These μ_{yjd} and σ^2_{yjd} correspond to θ-parameters of EM-algorithm.

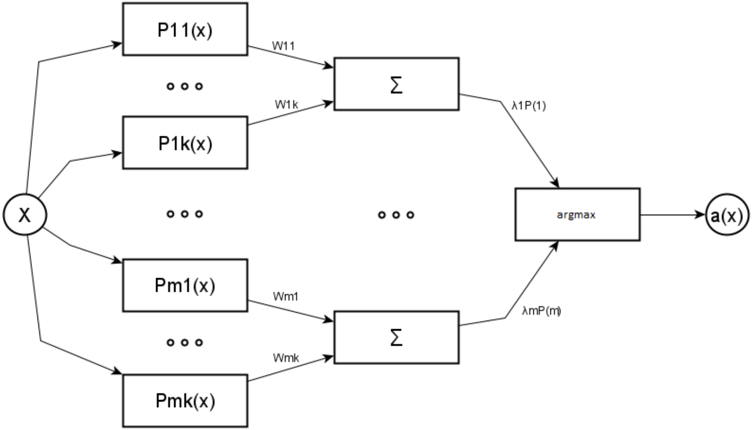
GMM: the classification algorithm

After that we can create a Gaussian distributions mixture model of classification:

$$a(x) = arg \max_{y \in Y} \lambda_y P(y) \sum_{i=1}^{k} w_{yj} N_{yj} e^{-\frac{1}{2}\rho_{yj}^2(x, \mu_{yj})},$$

where $N_{yj} = (2\pi)^{-n/2}(\sigma_{yj1} \cdot \dots \cdot \sigma_{yjn})^{-1}$ - normalizing term, $\rho_{yj}^2(x, \mu_{yj}) = \sum_{d=1}^{n} \frac{1}{\sigma_{yjd}^2}(f_d(x) - \mu_{yjd})^2$ - the Gaussian probability distribution of the feature with index d.

Radial basis functions



m - the number of classes, $P(i)$ - a priori probability of class i, k - the number of components

Advantages and disadvantages

+ EM-algorithm automatically clusterizes classes.
+ EM-algorithm usually requires fewer iterations than other iterative algorithms.
+ This algorithm allows for a risk assessment.
- EM-algorithm is very sensitive to the initial Θ.
- There is no common method of defining the parameter k - the number of components.

Conclusions

- We investigated the main ideas of probabilistic models to solve classification problems.
- We learned the difference between generative and discriminative models, and we saw how to interpret Bayesian models as discriminative models.
- We prove that logistic regression is a linear model.
- We investigated how to train probabilities densities mixture using EM-algorithm and proved it's work.