TTIC 31020: Introduction to Machine Learning
Autumn 2022

Problem Set #3

Collaborated with Haichuan Wang

1 Kernels

Here we will look at an example of constructing feature spaces for classification problems. We will explore the idea of the "kernel trick" applied to classifiers other than SVM (i.e., classifiers using loss functions other than hinge loss). Specifically, we will introduce the kernel logistic regression (KLR). Recall that in class we have described the general logistic regression model as

$$\hat{p}\left(y=1 \mid \mathbf{x}; \mathbf{w}, w_0\right) = \frac{1}{1+\exp\left(\sum_{j=1}^{d} w_j \phi_j(\mathbf{x})\right)},$$

where $\phi_j(\mathbf{x})$ is the j-th *basis function*, or *feature* - generally, a function mapping $\mathcal{X} \to \mathbb{R}$.

Problem 1 [**15 points**]
Show how by appropriate choice of basis functions $\boldsymbol{\phi}$, given a training set $\mathbf{x}_1, \ldots, \mathbf{x}_N$, one can obtain a logistic regression model whose predictions on a test point $\mathbf{x}_0$ depend on the training data only through the kernel values $K(\mathbf{x}_i, \mathbf{x}_0)$ for $i = 1, \ldots, N$. Then write down the gradient of the loss, with L_2 regularization, on a single example, and show that the training for this model via gradient descent also depends on the training data only through kernel computations.

End of problem 1

$$\boxed{\log \frac{e^{\theta}}{1+e^{\theta}} = \log \frac{1}{e^{-\theta}+1}}$$

• let $\phi_j(x) = \ker(x_j, x)$, $j \in [1, N]$. Let $d=N$.

Then

$$\hat{p}(y=1 \mid x; w, w_0) = \frac{1}{1+\exp\left(\sum_{j=1}^{d} w_j k(x_j, x_0)\right)}$$

Let $\theta = \sum_{j=1}^{d} w_j k(x_j, x_0)$.

• For one sample,

$$L(w) = -\left(y_i \cdot \log \frac{1}{1+e^{\theta}} + (1-y_i) \log \underbrace{\left(1-\frac{1}{1+e^{\theta}}\right)}_{\frac{e^{\theta}}{1+e^{\theta}}}\right) + \lambda \sum_{j=1}^{d} \|w_j\|^2$$

$$= -\left(-y_i \cdot \log(1+e^{\theta}) + \theta - \log(1+e^{\theta}) - y_i\theta + y_i \log(1+e^{\theta})\right) + \lambda \sum_{j=1}^{d} \|w_j\|^2$$

$$= -\left(\theta - y_i\theta - \log(1+e^{\theta})\right) + \lambda \sum_{j=1}^{d} \|w_j\|^2$$

$$= -\left(-y_i\theta + \log e^{\theta} - \log(1+e^{\theta})\right) + \lambda \sum_{j=1}^{d} \|w_j\|^2$$

$$= -\left(y_i\theta + \log\left(\frac{e^{\theta}}{1+e^{\theta}}\right)\right) + \lambda \sum_{j=1}^{d} \|w_j\|^2$$

$$= -\left(y_i\theta - \log(1+e^{-\theta})\right) + \lambda \sum_{j=1}^{d} \|w_j\|^2$$

• Take derivative,

$$\frac{\partial}{\partial w_j} L(w) = -\left(y_i k(x_j, x_0) - \frac{e^{-\theta} \cdot -k(x_j, x_0)}{1+e^{-\theta}}\right) + 2\lambda \|w_j\|$$

$$= -y_i k(x_j, x_0) - \frac{k(x_j, x_0)}{1+e^{\theta}} + 2\lambda \|w_j\|$$

• There fore, the gredient is

$$\nabla L(w) = \begin{pmatrix} \frac{\partial L}{\partial w_1} \\ \vdots \\ \frac{\partial L}{\partial w_d} \end{pmatrix}$$

with the update rule

$$W_{new} = W - \alpha \cdot \nabla L$$

We observe that the proces depends only on traing data through kernel.

$$\log \frac{1}{1+e^{\theta}}$$

2 Support Vector Machines

Now we will consider some details of the dual formulation of SVM, the one in which we optimize the Lagrange multipliers α_i. In class we saw how to derive a constrained quadratic program, which can then be "fed" to an off-the-shelf quadratic program solver. These solvers are usually constructed to handle certain standard formulations of the objective and constraints.

Specifically the canonical form of a quadratic program with linear constraints is, mathematically:

$$\begin{aligned} &\operatorname*{argmin}_{\boldsymbol{\alpha}} \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{H}\boldsymbol{\alpha} + \mathbf{f}^T\boldsymbol{\alpha}, && (1)\\ &\text{such that: } \mathbf{A}\cdot\boldsymbol{\alpha} \le \mathbf{a}, && (2)\\ &\mathbf{B}\cdot\boldsymbol{\alpha} = \mathbf{b}, && (3)\\ & && (4) \end{aligned}$$

The vector $\boldsymbol{\alpha} \in \mathbb{R}^N$, where N is the number of training examples, contains the unknown variables to be solved for. The matrix $\mathbf{H} \in \mathbb{R}^{N\times N}$ and vector $\mathbf{f} \in \mathbb{R}^N$ specify the quadratic objective; the matrix $\mathbf{A} \in \mathbb{R}^{k_{ineq}\times N}$ and vector $\mathbf{a} \in \mathbb{R}^{k_{ineq}}$ specify k_{ineq} inequality constraints. Similarly, $\mathbf{B} \in \mathbb{R}^{k_{eq}\times N}$ and vector $\mathbf{b} \in \mathbb{R}^{k_{eq}}$ specify k_{eq} equality constraints. Note that you can express a variety of inequality constraints by adding rows to $\mathbf{B}$ and elements to $\mathbf{b}$; think how you would do it to express, e.g., a "greater or equal" constraint.

Problem 2 **[15 bonus points]**
Describe in detail how you would compute $\mathbf{H}$, $\mathbf{f}$, $\mathbf{A}$, $\mathbf{a}$, $\mathbf{B}$, and $\mathbf{b}$, to set up the dual optimization problem for the kernel SVM

$$\operatorname*{argmin}_{\mathbf{w}} \left\{\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\max\left[0, 1-y_i\left(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i)-w_0\right)\right]\right\}$$

given a kernel function $K(\cdot,\cdot)$ corresponding to the dot product in $\boldsymbol{\phi}$ space, and N training examples $(\mathbf{x}_i, y_i)$

End of problem 2

Approach1:

- use z_i to represent $\max\{0, 1-y_i(w^T\phi(x_i)-w_0)\}$, with $\begin{cases} z_i \ge 0 \\ z_i \ge 1-y_i(w^T\phi(x_i)-w_0). \end{cases}$

- assign $\begin{cases} \alpha_i \cdot z_i \le 0, \\ \beta_i \cdot z_i \le 1-y_i(w^T\phi(w_i)-w_0) \end{cases}$, where $\alpha_i \le 0$, $\beta_i \le 0$.

- Then, let

$$H = \begin{pmatrix} I \\ 0 \end{pmatrix},\quad \alpha = \begin{pmatrix} w \\ z \end{pmatrix},\quad f = \begin{array}{c} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ C \\ \vdots \\ C \end{pmatrix} \end{array}\begin{array}{l} \}\,d \\ \}\,N \end{array}$$

We get

$$\operatorname*{argmax}_{\alpha_i,\beta_i}\left\{\frac{1}{2}\cdot(w,z)\begin{pmatrix} I \\ 0 \end{pmatrix}\begin{pmatrix} w \\ z \end{pmatrix} + (\underbrace{0,\dots,0}_{d},\underbrace{C,\dots,C}_{N})\cdot\begin{pmatrix} w \\ z \end{pmatrix}\right\}$$

$$=\operatorname*{argmax}_{\alpha_i,\beta_i}\left\{\frac{1}{2}\|w\|^2 + C\cdot\sum_{i=1}^{N} z_i\right\}$$

Approach2:

- From Tutorial 5, the dual is

$$\max_{\alpha_i} \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \phi(x_i)\cdot\phi(x_j)$$

$$\begin{cases} 0 \le \alpha_i \le C \\ \sum_{i=1}^{N}\alpha_i y_i = 0 \end{cases}$$

Which is

$$\min_{\alpha} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \phi(x_i)\cdot\phi(x_j) - \sum_{i=1}^{N}\alpha_i$$

$$\begin{cases} 0 \le \alpha_i \le C \\ \sum_{i=1}^{N}\alpha_i y_i = 0 \end{cases}$$

- Define

- $f = \begin{pmatrix} -1 \\ \vdots \\ -1 \end{pmatrix}\}N$

- $a = \begin{pmatrix} C \\ \vdots \\ C \\ 0 \\ \vdots \\ 0 \end{pmatrix}\begin{array}{l} \}N \\ \}N \end{array}$

- $A = \begin{pmatrix} I_N \\ \hline -I_N \end{pmatrix}\begin{array}{l} \}N\times N \\ \}N\times N \end{array}$

- $b = 0$

- $B = (y_1 \cdots y_N)$

- $H = \overbrace{\begin{pmatrix} \vdots & & \\ \cdots & & \\ & & \end{pmatrix}}^{N}\}N$

$H_{ij} = y_i y_j K(x_i, x_j)$

This will reconstruct our dual function.

Next, we will consider finding the value of b, the threshold (bias) term for a kernel SVM

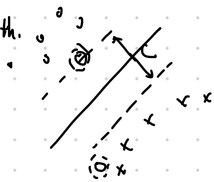
$$h(\mathbf{x}) = \operatorname{sign}\left(\sum_{i:\alpha_i>0} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \right),$$

where α_i are the coefficients for the support vectors in the training data.

Problem 3 [**10 points**]
Suppose you have solved for α_i in the SVM classifier. Explain how exactly you can calculate b.

End of problem 3

• Let C be the margin length.



$$W = \sum_{i=1}^{N} \alpha_i y_i \phi(x_i)$$

As only the points exactly on the boundary matter, we consider the points with $0 < \alpha_i < C$.

• Pick x_j on the boundary,

$$y_i \Big(\sum_{\substack{i \text{ s.t.} \\ 0<\alpha_i<C}} \alpha_i y_i K(x_i, x_j) + b \Big) = 1$$

$$b = \underbrace{\frac{1}{y_i}}_{y_i = \pm 1} - \sum_{\substack{i \text{ s.t.} \\ 0<\alpha_i<C}} \alpha_i y_i k(x_i, x_j)$$

4 Multivariate Gaussians

Since many (probably most) generative models for real-valued data involve multivariate Gaussian distribution, it is worth getting to know them better. In particular, here we will understand why one always sees those elliptical contours drawn when Gaussians are visualized in 2D.

Recall that the probability density function (pdf) of a Gaussian distribution in $\mathbb{R}^d$ is given by

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\mathbf{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right). \quad (5)$$

Problem 6 **[10 points]**
Show that a contour corresponding to a fixed value of pdf is an ellipse in the 2D space, $\mathbf{x} = [x_1, x_2]$

End of problem 6

Advice: You may find it easier to work in the log domain, and to write out explicitly the expression in (5) in terms of x_1 and x_2.

$$\frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\tfrac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) = \text{const}$$

$$(x-\mu)^T\Sigma^{-1}\cdot(x-\mu) = \text{const}$$

• Case $\Sigma = \begin{pmatrix}\sigma_1^2 & 0\\ 0 & \sigma_2^2\end{pmatrix}$, $\Sigma^{-1} = \begin{pmatrix}\frac{1}{\sigma_1^2} & 0\\ 0 & \frac{1}{\sigma_2^2}\end{pmatrix}$

$$(x_1-\mu_1, x_2-\mu_2)\begin{pmatrix}\frac{1}{\sigma_1^2} & 0\\ 0 & \frac{1}{\sigma_2^2}\end{pmatrix}\begin{pmatrix}x_1-\mu_1\\ x_2-\mu_2\end{pmatrix} = C$$

$$\frac{(x_1-\mu_1)^2}{\sigma_1^2} + \frac{(x_2-\mu_2)^2}{\sigma_2^2} = C$$

Divide C on both sides, we get the form of an ellipse.

• Case $\Sigma = \begin{pmatrix}\sigma_1^2 & \sigma_{12}^2\\ \sigma_{12}^2 & \sigma_2^2\end{pmatrix}$,

As covariance matricies are symetric, they are diagonalizable.

Write

$$\Sigma = PDP^{-1}$$

then

$$\Sigma^{-1} = PD^{-1}P^{-1}$$

• $$\underbrace{(x_1-\mu_1, x_2-\mu_2)\begin{pmatrix}a & b\\ c & d\end{pmatrix}\begin{pmatrix}u_1 & 0\\ 0 & u_2\end{pmatrix}}_{①}\underbrace{\begin{pmatrix}a' & b'\\ c' & d'\end{pmatrix}\begin{pmatrix}x_1-\mu_1\\ x_2-\mu_2\end{pmatrix}}_{②}$$

①

$$= \left[(x_1-\mu_1)(a, b) + (x_2-\mu_2)(c, d)\right]\begin{pmatrix}u_1 & 0\\ 0 & u_2\end{pmatrix}$$

$$= \left[(x_1-\mu_1)a\cdot u_1, 0\right] + \left[0, (x_2-\mu_2)du_2\right]$$

$$= \left[(x_1-\mu_1)au_1, (x_2-\mu_2)du_2\right]$$

②

$$= \begin{bmatrix}a'(x_1-\mu_1) + b'(x_2-\mu_2)\\ c'(x_1-\mu_1) + d'(x_2-\mu_2)\end{bmatrix}$$

multiply ① and ②:

We should obtain the form

$$\frac{(x_1-\mu_1)^2}{c_1^2} + \frac{(x_2-\mu_2)^2}{c_2^2} = 1$$

5 Generative models

With help from Haichuan Wang

Here we will consider the Gaussian generative model for $\mathbf{x} \in \mathbb{R}^d$ which assumes equal, *isotropic* covariance matrices for each class:

$$\Sigma_c = \begin{bmatrix} \sigma_1^2 & \cdots & 0 \\ & \ddots & \\ 0 & \cdots & \sigma_d^2 \end{bmatrix}. \tag{6}$$

The classes of course have separate means. That is, the conditional density of the j-th coordinate of $\mathbf{x}$ given class c is a Gaussian $\mathcal{N}(\mu_{c,j}, \sigma_j)$, with these densities independent (given class) for different values of j.

For simplicity, let's consider a two-class problem, with $y \in \{0, 1\}$. As we discussed in class, when training the generative model, we simply fit $p(\mathbf{x} \mid y)$ and $p(y)$ to the training data, and use the resulting discriminant analysis to produce a decision rule which predicts

$$\widehat{y}(\mathbf{x}) = \operatorname*{argmax}_c \{p(\mathbf{x} \mid y = c)\, p(y = c)\}. \tag{7}$$

However, this model does also produce an (implicit) estimate for the posterior $p(y = c \mid \mathbf{x})$.

Problem 7 [15 points]
Show that the posterior $p(y = c \mid \mathbf{x})$ resulting from the generative model above has the same form as the posterior in logistic regression model,

$$p(y = c \mid \mathbf{x}) = \frac{1}{1 + \exp(b + \mathbf{w} \cdot \mathbf{x})}, \tag{8}$$

for appropriate values of $b \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^d$.

End of problem 7

Advice: Start with Bayes rule, and use the simplifying assumptions in (6) to derive the posterior.

- Bayes rule:

$$p(y=c \mid x) = \frac{p(x \mid y=c)\, p(y=c)}{p(x)} = \frac{p(x \mid y=c) \cdot p(y=c)}{- - - - -}$$

- As each class (c and c') has $N(\mu_{c,j}, \sigma_j)$ distribution,

- $$p(x \mid y=c) = \frac{1}{(2\pi)^{\frac{d}{2}} (\sigma_1^2 \cdots \sigma_d^2)^{\frac{1}{2}}} \cdot \exp\left(-\frac{1}{2}(x-\mu_c)^T \begin{pmatrix} \frac{1}{\sigma_1^2} & & \\ & \ddots & \\ & & \frac{1}{\sigma_d^2} \end{pmatrix} (x-\mu_c)\right)$$

$$= \prod_{j=1}^{d} \frac{1}{\sqrt{2\pi}\,\sigma_j} \exp\left(-\frac{1}{2\sigma_j^2}(x_j - \mu_{cj})^2\right)$$

- $$p(x \mid y=c') = \prod_{j=1}^{d} \frac{1}{\sqrt{2\pi}\,\sigma_j^2} \exp\left(-\frac{1}{2\sigma_j^2}(x_j - \mu_{c'j})^2\right)$$

- Now,

$$p(y=c \mid x) = \frac{p(x \mid y=c) \cdot p(y=c)}{p(x \mid y=c) \cdot p(y=c) + p(x \mid y=c')\, p(y=c')}$$

$$= \frac{1}{1 + \dfrac{p(x \mid y=c')\, p(y=c')}{p(x \mid y=c)\, p(y=c)}}$$

$$= \frac{1}{1 + \dfrac{p(y=c')}{p(y=c)} \cdot \exp\left(\sum_{j=1}^{d} \frac{1}{2\sigma_j^2}\left((x_j-\mu_{cj})^2 - (x_j-\mu_{c'j})^2\right)\right)}$$

$$= \frac{1}{1 + \exp\left(\sum_{j=1}^{d} \frac{\mu_{cj}-\mu_{c'j}}{\sigma_j} x_j - \sum_{j=1}^{d} \frac{1}{2\sigma_j^2}(\mu_{c'j}+\mu_{cj})(\mu_{cj}-\mu_{c'j}) + (\log p(y=c') - \log p(y=c))\right)}$$

- To get the form of

$$\frac{1}{1 + \exp(b + w x)},$$

We can let

$$w = \begin{pmatrix} \frac{\mu_{c,1} - \mu_{c',1}}{\sigma_1} \\ \vdots \\ \frac{\mu_{c,d} - \mu_{c',d}}{\sigma_d} \end{pmatrix}$$

$$b = \log p(y=c') - \log p(y=c) - \sum_{j=1}^{d} \frac{1}{2\sigma_j^2}(\mu_{c'j}+\mu_{cj})(\mu_{cj}-\mu_{c'j})$$

Problem 8 [10 points]

The previous problem established that the two models – logistic regression and the linear discriminant analysis based on the isotropic Gaussian model (6) – have the same form of the posterior $p(y \mid \mathbf{x})$. Will the two models produce the same classifier when applied to a given training set? Why or why not?

End of problem 8

- The two models will not produce the same classifier, because in the Gaussian model, we made many additional assumptions e.g. the distribution being $N(\mu_c, \Sigma_c)$ for each class, the coordinates are independent, etc.

- However, for the logistic regression, the assumptions are less. Therefore, the generative model is harder to fit a distribution when there are more data. But logistic regression performs better when the training set is large.