Statistical Machine Learning GR5241
Spring 2021

Homework 3

Due: 03/28/2001 by 11:59pm

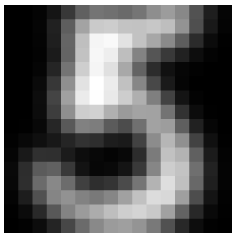
Homework submission: Please submit your homework as a pdf in Canvas.

Problem 1 (SVM and CV [30 points])

In this problem, we will apply a support vector machine to classify hand-written digits. You do not have to implement the SVM algorithm: The R library e1071 provides an implementation, see

`http://cran.r-project.org/web/packages/e1071/index.html`

Download the digit data set from the course website. The zip archive contains two files: Both files are text files. Each file contains a matrix with one data point (= vector of length 256) per row. The 256-vector in each row represents a 16×16 image of a handwritten number. The data contains two classes—the digits 5 and 6—so they can be labeled as -1 and $+1$, respectively. The image on the right shows the first row, re-arranged as a 16×16 matrix and plotted as a gray scale image.



- Randomly select about 20% of the data and set it aside as a test set.
- Train a linear SVM with soft margin. Cross-validate the margin parameter.
- Train an SVM with soft margin and RBF kernel. You will have to cross-validate both the soft-margin parameter and the kernel bandwidth.
- After you have selected parameter values for both algorithms, train each one with the parameter value you have chosen. Then compute the misclassification rate (the proportion of misclassified data points) on the test set.

Homework questions:

1. Plot the cross-validation estimates of the misclassification rate. Please plot the rate as
   (a) a function of the margin parameter in the linear case.
   (b) a function of the margin parameter and the kernel bandwidth in the non-linear case (you are encouraged to use heat map here).
2. Report the test set estimates of the misclassification rates for both cases, with the parameter values you have selected, and compare the two results. Is a linear SVM a good choice for this data, or should we use a non-linear one?

Problem 2 (Kernels [25 points])

Please see the background information provided below. The required tasks for problem 2 are described at the top of page 2.

Recall the definition of a kernel: A function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is called a **kernel** on $\mathbb{R}^d$ if there is *some* function $\phi : \mathbb{R}^d \to \mathcal{F}$ into *some* space $\mathcal{F}$ with scalar product $\langle \cdot, \cdot \rangle_{\mathcal{F}}$ such that

$$k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{F}} \qquad \text{for all } \mathbf{x}, \mathbf{x}' \in \mathbb{R}^d.$$

How do you prove k is a valid kernel? Two recommendations **(RI)** and **(R2)** follow:

RI: The first recommendation is a direct application of Mercer's theorem.

- To prove $k(\mathbf{x}, \mathbf{x}')$ is a valid kernel, it's sufficient to show that $k(\mathbf{x}, \mathbf{x}')$ is positive-definite, i.e., for any function $f : \mathbb{R}^d \to \mathbb{R}$,
$$\int_{\mathbb{R}^d \times \mathbb{R}^d} k_1(\mathbf{x}, \mathbf{x}') f(\mathbf{x}) f(\mathbf{x}') d\mathbf{x} d\mathbf{x}' \geq 0.$$
- Note that a valid kernel $k(\mathbf{x}, \mathbf{x}')$ must always be symmetric. Before showing positive-definiteness, make sure to check that $k(\mathbf{x}, \mathbf{x}')$ symmetric. This step is often glossed over because a theorist would never consider a non-symmetric kernel, and hence prove its validity.

RII: The second recommendation uses the definition of a kernel function directly. Note that it is often convenient to invoke Mercer's theorem.

- To prove $k(\mathbf{x}, \mathbf{x}')$ is a valid kernel, you can **directly identify** the function $\phi : \mathbb{R}^d \to \mathcal{F}$ such that
$$k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{F}} \qquad \text{for all } \mathbf{x}, \mathbf{x}' \in \mathbb{R}^d.$$
For simple examples, the function ϕ is often easy to identify.
- For more complicated cases, it's convenient to invoke Mercer's theorem and exploit the decomposition:
$$k(\mathbf{x}, \mathbf{x}') = \sum_{j=1}^{\infty} \lambda_j \phi_j(\mathbf{x}) \phi_j(\mathbf{x}')$$

Required tasks: (1.i) - (1.v)

Let $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$ and assume that $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$ are valid kernels. Note that you can assume both k_1, k_2 are symmetric and positive-definite. Show the following kernels are also valid:

2.i Prove that the scaler product $k^*(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$ is a valid kernel. This is the most basic case. 2

2.ii For real $a > 0$, show $k^*(\mathbf{x}, \mathbf{x}') = ak_1(\mathbf{x}, \mathbf{x}')$ is a valid kernel. 1

2.iii For any function $g : \mathbb{R}^d \to \mathbb{R}$, show $k^*(\mathbf{x}, \mathbf{x}') = g(\mathbf{x})k_1(\mathbf{x}, \mathbf{x}')g(\mathbf{x}')$ is a valid kernel. 2

2.iv Show $k^*(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}')$ is a valid kernel.

2.v This problem is an application of the radial basis SVM and there is no required proof. Using the dataset "`HW3Problem2.csv`", run the radial basis SVM using a few bandwidths. Plot the resulting decision boundaries with the training data. Use a built-in function from `R` or `Python` to solve tis problem, .e.g., `svm()`.

Problem 3 (Training the Linearly Separable SVM [25 points])

Set-up and background:

The goal of this question is to *manually code* an optimization method for estimating a linearly separable SVM. We consider a simple case with two features x_1, x_2. Denote $\mathbf{v}_H = \begin{pmatrix} w_1 & w_2 \end{pmatrix}^T$ and denote the full parameter vector as $\boldsymbol{\beta} = \begin{pmatrix} c & w_1 & w_2 \end{pmatrix}^T$. The dataset of interest "`svmdata.csv`" consists of $n = 100$ training cases with response labels $y = 1$ or $y = -1$. For a linearly separable SVM, you must solve the optimization problem:

$$\min_{\mathbf{v}_H, c} \{\|\mathbf{v}_H\|\}$$
$$y_i(\langle \mathbf{v}_H, \mathbf{x}_i \rangle + c) \geq 1, \quad \text{for} \quad i = 1, \ldots, 100$$

Note: I swapped "$-c$" with "c" as compared to the expression presented in the class notes. The end result does not matter but it's easier to keep track of the signs when using c.

How to solve this problem:

One method of solving the SVM minimization problem is to define a new objective $Q(\mathbf{v}_H, c)$ as follows:

$$\begin{aligned} Q(\mathbf{v}_H, c) &= \frac{1}{n}\sum_{i=1}^{n} L(y_i, \langle \mathbf{v}_H, \mathbf{x}_i\rangle + c) + \frac{\lambda}{2}\|\mathbf{v}_H\|_2^2 \\ &= \frac{1}{n}\sum_{i=1}^{n} \left(L(y_i, \langle \mathbf{v}_H, \mathbf{x}_i\rangle + c) + \frac{\lambda}{2}\|\mathbf{v}_H\|_2^2 \right) \\ &= \frac{1}{n}\sum_{i=1}^{n} Q_i(\mathbf{v}_H, c), \end{aligned}$$

where $\lambda > 0$ is a tuning parameter and $L(y, f)$ is the **hinge loss** function defined by $L(y, f) = \max\{0, 1 - yf\}$. Also note that $Q_i(\mathbf{v}_H, c)$ is the objective at case i, i.e.,

$$\begin{aligned} L(y_i, \langle \mathbf{v}_H, \mathbf{x}_i\rangle + c) &= L(y_i, \mathbf{v}_H^T\mathbf{x}_i + c) \\ &= \max\{0, 1 - y_i(\mathbf{v}_H^T\mathbf{x}_i + c)\} \\ &= \max\{0, 1 - y_i(w_1 x_{i1} + w_2 x_{i2} + c)\} \\ &= \max\{0, 1 - z_i\}, \end{aligned}$$

where $z_i = y_i(w_1 x_{i1} + w_2 x_{i2} + c)$. The other component can also be simplified:

$$\frac{\lambda}{2}\|\mathbf{v}_H\|_2^2 = \frac{\lambda}{2}(w_1^2 + w_2^2)$$

The goal is to minimize $Q(\mathbf{v}_H, c)$. However, the hinge loss function $g(z) = \max\{0, 1 - z\}$ is not differentiable at the point $z = 1$. Hence the traditional gradient descent algorithm is not well defined in this exercise. Fortunately the function $g(z) = \max\{0, 1 - z\}$ is differentiable for every point except $z = 1$, and we can analytically solve the gradient for the two cases $z < 1$ and $z > 1$.

Required tasks: (3.i) - (3.iv)

3.i Construct a plot over the the range $-1 \le z \le 3$ which shows shows hinge loss $g(z) = \max\{0, 1 - z\}$ as a function of z. This is an easy problem. Students should notice that hinge loss is not differentiable at $z = 1$.

3.ii Consider the objective defined for case i: $Q_i(\mathbf{v}_H, c)$. Derive an expression for $\nabla Q_i(\mathbf{v}_H, c)$ assuming $z_i < 1$. Also derive an expression for $\nabla Q_i(\mathbf{v}_H, c)$ assuming $z_i > 1$.

3.iii Linearity of gradients allows us to compute the full gradient:

$$\nabla Q(\mathbf{v}_H, c) = \frac{1}{n}\sum_{i=1}^{n} \nabla Q_i(\mathbf{v}_H, c).$$

Task: Using the training data "`svmdata.csv`", estimate the SVM model by running the gradient descent algorithm:

$$\boldsymbol{\beta}_{(t+1)} := \boldsymbol{\beta}_{(t)} - \eta_t \frac{1}{n}\sum_{i=1}^{n} \nabla Q_i(\boldsymbol{\beta}_{(t)})$$

Here we denote the full parameter vector as $\boldsymbol{\beta} = \begin{pmatrix} c & w_1 & w_2 \end{pmatrix}^T$. The key difference between the above algorithm and regular gradient descent is that you must check whether $z_i < 1$ or $z_i > 1$ for each case $i = 1, 2, \ldots, 100$. I recommend choosing

$$\eta_t = \frac{1}{t\lambda} \quad \text{and} \quad \lambda \approx .25.$$

You are welcome to try different values of λ and compare your estimated model to the `R` function `svm()`.

3.iv Construct a plot of x_1 versus x_2 with your estimated linear decision boundary from part 2.iii. Also display your estimated coefficients

$$\hat{\boldsymbol{\beta}} = \begin{pmatrix} \hat{c} & \hat{w}_1 & \hat{w}_2 \end{pmatrix}^T$$

A few notes follow:

(a) This particular problem is known as **The Pegasos Algorithm**. The development presented in HW3 is not formal but the key ideas still hold true.

(b) You must use `R`, `Python` or another programming language to manually solve problem 3.iii. Don't just run the `svm()` function and call it a day!

(c) I encourage students to check your final answer with the `svm()` function from `R` or similar packages.

(d) You must choose a threshold $\epsilon > 0$ to break your algorithm or just run it for several iterations.

Expect one more SVM problem by the end of the week.