Exercises
Machine Learning
HS 2016

Series 4, Nov 21th, 2015 (Non-linear SVMs & Kernels)

Machine Learning Laboratory
Dept. of Computer Science, ETH Zürich
Prof. Joachim M. Buhmann
Web https://ml2.inf.ethz.ch/courses/ml/

Email questions 1,2,3,4 to: **Jiadong Guo** jguo@student.ethz.ch
Email questions 5,6 to: **Junyao Zhao** zhaoju@student.ethz.ch

Please turn in solutions until Friday, Nov 28th.
("*"-exercies are a little bit more difficult, but still useful)

Problem 1 (Overfitting with non-linear SVMs):

We have seen that the formulation of the primal, soft-margin, SVM is:

$$\begin{aligned} \underset{\mathbf{w},\xi}{\text{minimize}} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & z_i\mathbf{w}^T\mathbf{y}_i \geq 1-\xi_i, \quad \forall i, \\ & \xi_i \geq 0 \quad \forall i. \end{aligned}$$

.

and, that the formulation of the dual, soft-margin, SVM is:

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j}^{n}\alpha_i\alpha_j z_i z_j \mathbf{y}_i^T\mathbf{y}_j \\ \text{subject to} \quad & \sum_{i}^{n} z_i\alpha_i = 0 \\ & 0 \leq \alpha_i \leq C \; \forall i. \end{aligned}$$

If we replace the inner product $\mathbf{y}_i^T\mathbf{y}_j$ with the kernel $k(\mathbf{y}_i, \mathbf{y}_j)$ we obtain the soft, non-linear SVM formulation. The optimization variable in this formualation is α while the tuning parameters are C and $k(x, y)$. In this exercise we explore the role that these tuning parameters have in the bias-variance trade off, as it applies to SVMs.

Note: You will be asked to simulate data and fit different SVM models to it. You may do it in any programming language, however MATLAB is recommended as we provide a code skeleton and a graphing function.

1. **Generate data**. First we generate some data with a fairly complex structure, to mimic real data we might encounter.
   (a) Generate 10 means $m_{1k}, k \in \{1, ..., 10\}$ from a bivariate Gaussian distribution $m_{1k} \sim N((1, 0)^T, I)$ and label this class 1.
   (b) Generate 10 means $m_{2k}, \in \{1, ..., 10\}$ from a bivariate Gaussian distribution $m_{2k} \sim N((0, 1)^T, I)$ and label this class 2.
   (c) Generate 100 class 1 training points: For each observation, pick an m_{1k} at random with probability 1/10, and then generate $Y_{1i} \sim N(m_{1k(i)}, I/5)$.
   (d) Generate 100 class 2 training points: For each observation, pick an m_{2k} at random with probability 1/10, and then generate $Y_{2i} \sim N(m_{2k(i)}, I/5)$.

(e) Generate test data: Generate 10,000 class 1 and 10,000 class 2 test points, as with training data. **Note**: Since we are able to simulate as much data as we want, we will be able to estimate the generalization error of fitted models without cross-validation.

2. **Linear kernel** Fit a linear SVM to simulated data.

   (a) Use MATLAB's fitcsvm to to fit linear SVM to the training data, for a few different penalty parameters $C = 0.02, 1, 1000$.

   (b) Graph the fitted model using the function graphSVM provided. What qualitative changes can be observed in terms of the margin, support vectors and classification regions.

   (c) Estimate training and generalization error. For 50 different values of $C \in \{C_1, ..., C_{50}\}$ where $C_1 = 0.01$ and $C_{50} = 1000$.

      i. Fit a linear SVM with penalty parameter C_i to training data.

      ii. Apply fitted model to training and test data and calculate error.

   (d) Graph the training and generalization error versus the penalty parameter. Can you observe any over-fitting going on for different values of C?

3. **Non-linear kernels** Fit non-linear SVM to simulated data.

   (a) Repeat steps in part 2, this time fitting an SVM with a 4th degree polynomial kernel.

   (b) Repeat steps in part 2, this time fitting an SVM with a RBF kernel.

   (c) What is the relationship between the kernel $k(\mathbf{x}, \mathbf{y})$ used, the penalty parameter C and the bias-variance trade-off? **Hint1**: Recall the size of the underlying feature space for the differnt kernels involved. **Hint2**: By looking at the primal formulation of the SVM model, determine the role that the parameter C plays in determing which term, $\frac{1}{2}\mathbf{w}^T\mathbf{w}$ or $\sum_{i=1}^{n} \xi_i$ is more important. How do different values of the optimization variable ξ influence the smoothness of the decision boundary in the original feature space?

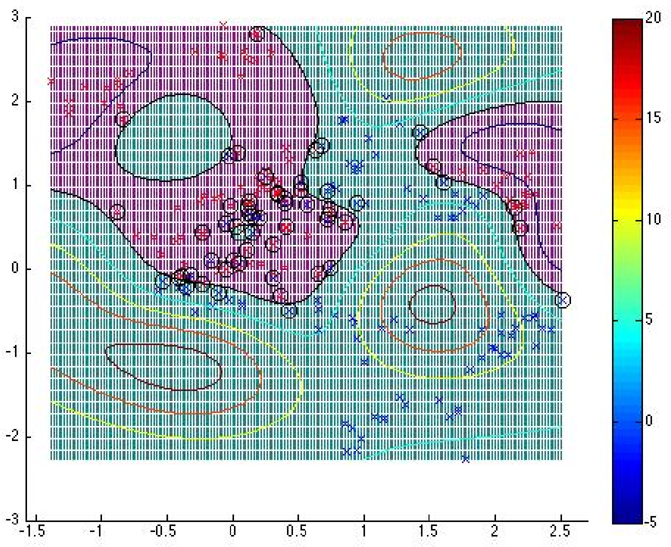

Figure 1: Visualization of a RBF SVM

Figure 1 above shows the output of function graphSVM for an SVM (RBF kernel) fitted to bivariate data generated as described in part 1. Training data is plotted with crosses, blue for one class, red for the other. Circled crosses indicate that a data point is a support vector. The green and purple background colours indicate the class 1 and class 2 prediction regions corresponding to the fitted model. Contour lines are included which indicate the distance to the separating hyperplane in the enlarged feature space.

Problem 2 (Identifying Kernel Functions):

For each of the following functions, determine which are kernel functions and which are not. Note that in order to show that a function is not a kernel, it suffices to show an example using either two or more points for which a property of a kernel function is violated. Unless mentioned otherwise, assume throughout that $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ with $\mathbf{x} = (x_1, \ldots, x_d)$ and $\|\cdot\|$ to be the Euclidean norm.

1. $k(\mathbf{x}, \mathbf{y}) = \prod_{i=1}^{d} h(\frac{x_i - c}{a}) h(\frac{y_i - c}{a})$ where $h(x) = \cos(1.75x)\exp(-x^2/2)$.
2. $k(\mathbf{x}, \mathbf{y}) = -\log_2(\|\mathbf{x} - \mathbf{y}\| + 1)$.
3. $k(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{d}(x_i - y_i)$.

4.* $k(X, Y) = 2^{|X \cap Y|}$ for $X, Y \subseteq \Omega$ where Ω is a finite set and $|\cdot|$ denotes cardinality. (**Hint:** *You might want to represent each $X \subseteq \Omega$ as a binary string $(x_1, x_2, \ldots, x_{|\Omega|})$ where $x_i = 1$ if $i \in X$ and 0 otherwise. Furthermore, you might also want to make use of some closure properties of kernels discussed in class*)

Problem 3 (Normalized and Gaussian kernels):

Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a kernel function and $f : \mathcal{X} \to \mathbb{R}$ be an arbitrary function. Then prove the following.

1. **(Normalized kernel)** $k'(\mathbf{x}, \mathbf{y}) = \frac{k(\mathbf{x}, \mathbf{y})}{\sqrt{k(\mathbf{x}, \mathbf{x})k(\mathbf{y}, \mathbf{y})}}$ is a kernel function taking values in $[-1, 1]$.
2. $\tilde{k}(\mathbf{x}, \mathbf{y}) = f(\mathbf{x})k(\mathbf{x}, \mathbf{y})f(\mathbf{y})$ is a kernel function.
3. **(RBF kernel)** With the help of (2) and kernel closure properties discussed in class, show that $\tilde{k}(\mathbf{x}, \mathbf{y}) = \exp(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{h^2})$ is a kernel function.

Problem 4 (Support Vector Regression):

[1] In class, we have encountered the support vector classifier and derived its primal and dual formulation. In this exercise, the support vector regressor is introduced and the dual formulation will be derived. The Primal-Dual formulation exist for all constraint optimization problem, no matter for solving a classifier or a regressor.

The following training data is given, $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots (\mathbf{x}_n, y_n)$, where $\mathbf{x}_i \in R^m, y_i \in R$.

A different loss function, The epsilon sensitive loss, is used here as the hinge loss:

$$L_\epsilon(\mathbf{x}, y, f) = |y - f(\mathbf{x})|_\epsilon = max(0, |y - f(\mathbf{x})| - \epsilon)$$

Here the $\mathbf{x}$ is the input vector, y is the output, and f is the function used for predicting the output of new data points.

With this type of loss function, we can now mimic the boundaries in the SVM classifier, as shown in figure 2.The doted lines are the boundaries inside which the loss function would take value zero, the variable ξ indicates the size of the violation if the training point goes beyond boundary.

[1]This problem is derived from Exercise in Machine Learning class 10-701/15-781 from Carnegie Mellon University, 2014.

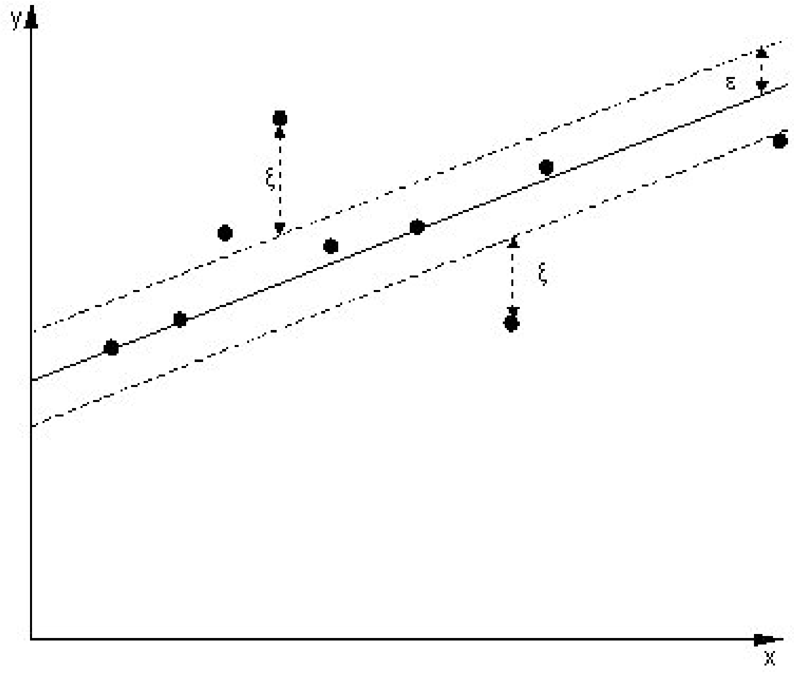

Figure 2: sketch of regressor using epsilon sensitive loss function

Recall the soft margin SVM derived in class, one can now define the SVR cost function in a similar manner:

$$COST = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} L_\epsilon(\mathbf{x}, y, f)$$

where $f(x) = \mathbf{w}^T\mathbf{x}$ (Homogeneous coordinates), and $C, \epsilon > 0$ are given parameters.

1. Introduce proper slack variables and rewrite the problems as a quadratic problem with linear inequality constraints, similar to the Primal form of the Soft margin SVM. This is called the Primal form of the support vector regression. Introduce multiple slack variables if necessary.
2. Construct the Lagrangian function as in tutorial class for the derived primal form. Keep the variable $\mathbf{w}$ in the formulation.
3. Now derive the dual. Start with setting the partial derivatives w.r.t. primal variables of the Lagrangian function to 0.
4. When is a point selected as a support vector, given the dual formulation and its variables?
5. Write down the function you would use for predicting the label of and unseen sample X, both in primal and dual.
6. Kernelize the **dual** now both for the training and prediction. Write down the new optimization function without its constraints for the training phase and the new prediction function.
7. Discuss how ϵ and C can change the regression result regarding the Bias-Variance trade-off. Answer the question like this: When ϵ/C grows, the Bias will become higher/lower and the variance will decrease/increase.

Problem 5 (Boosting):

Suppose you obtain a data set $(\mathbf{x}_1, y_1) \dots (\mathbf{x}_n, y_n)$ sampled i.i.d. from some distribution $P(\mathbf{X}, Y)$. You wish to learn a mapping $f : \mathcal{X} \to \mathcal{Y}$.

Similar to the AdaBoost algorithm, gradient boosting is a methodology to greedily approximate f using gradient descent based on the additive form

$$f_M(x) = \sum_{i=1}^{M} \beta_i h_i(\mathbf{x})$$

where h_i are weak learners.

The goal is to find an approximated function $\hat{f}$ of the function f^* that minimizes the expected value of a differentiable loss function $L(y, f)$ over the joint distribution of all $(\mathbf{x}, y)$ values.

$$f^*(\mathbf{x}) = \arg\min_{f(\mathbf{x})} \mathbb{E}_{y|\mathbf{x}}[L(y, f(\mathbf{x}))]$$

At an iteration m, the greedy approach attempts to find the update $\beta_m h_m(\mathbf{x})$ that minimizes the expected loss.

$$\beta_m, h_m = \arg\min_{\beta, h} \sum_{i=1}^{n} L(y_i, f_{m-1}(\mathbf{x}_i) + \beta h(\mathbf{x}_i))$$

and then,

$$f_m(\mathbf{x}) = f_{m-1} + \beta_m h_m(\mathbf{x}).$$

The hypothesis h_m is selected to be the most correlated to the negative gradient of the loss function.

The algorithm works as follows:

1. Initialize $\hat{f}_0(\mathbf{x}) = \arg\min_h \sum_{i=1}^{n} L(y_i, h(\mathbf{x}_i))$
2. For m = 1 to M:
   (a) Compute the negative gradient
   $$-g_m(\mathbf{x}_i) = -\left[\frac{\partial L(y_i, f(\mathbf{x}_i))}{\partial f(\mathbf{x}_i)}\right]_{f=\hat{f}_{m-1}(\mathbf{x}_i)}, i = 1 \dots n$$
   (b) Fit a function h_m to the negative gradient by least squares
   $$h_m = \arg\min_h \sum_{i=1}^{n} (-g_m(\mathbf{x}_i) - h(\mathbf{x}_i))^2$$
   (c) Find β_m to minimize the loss
   $$\beta_m = \arg\min_\beta \sum_{i=1}^{n} L(y_i, \hat{f}_{m-1}(\mathbf{x}_i) + \beta h_m(\mathbf{x}_i))$$
   (d) Update $\hat{f}$
   $$\hat{f}_m(\mathbf{x}) = \hat{f}_{m-1} + \beta_m h_m(\mathbf{x})$$
3. Output $\hat{f}_m$ for regression or the sign of $\hat{f}_m$ for classification

1. Based on this general procedure, write the precise steps required to approximate a function using the squared error loss $L(y, F) = (y - f)^2/2$ to perform regression.

2. Analyze what the algorithm does with this loss function and explain it in a few sentences.

Problem 6 (SSVM):

As seen in the tutorial, the primal formulation of structural SVM is the following:

$$\begin{aligned} &\min_{w,\xi} \frac{\lambda}{2}||w||^2 + \frac{1}{n}\sum_{i=1}^{n} \xi_i \\ &\text{s.t. } \langle w, \phi(x_i, y_i) - \phi(x_i, y) \rangle \geq L(y_i, y) - \xi_i \ \forall i, \ \forall y \in \mathcal{Y}(x_i). \end{aligned} \tag{1}$$

To simplify the notation, denote $\psi_i(y) := \phi(x_i, y_i) - \phi(x_i, y)$, $L_i(y) := L(y_i, y)$ and $\mathcal{Y}_i = \mathcal{Y}(x_i)$. Then use Lagrange method to derive the dual formulation.