Introduction to Machine Learning via Statistical Learning Theory

Problem set 2: Experimentation

Due March 13th

Problem 1: Surrogates to the Mis-Classification Loss

Given a training set $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ and a feature mapping $\phi : \mathcal{X} \to \mathbb{R}^d$, we might want to find the linear classifier minimizing the number of miss-classification errors. That is, find $w \in \mathbb{R}^d$ minimizing the empirical error:

$$\hat{R}_{01}(w) = \sum_{i=1}^{m} loss_{01}(w'\phi(x_i); y_i) \tag{1}$$

$$loss_{01}(z, y) = \begin{cases} 1 & \text{if } sign(z) \neq y \\ 0 & \text{if } sign(z) = y \end{cases} \tag{2}$$

Unfortunately, this is an NP-hard problem. In class, we discussed replacing the mis-classification loss with the "hinge-loss":

$$loss_{\text{hinge}}(z, y) = \max(0, 1 - yz). \tag{3}$$

Minimizing $\hat{R}_{\text{hinge}}(w) = \sum_{i=1}^{m} loss_{\text{hinge}}(z, y)$ can be cast as a linear program and solved efficiently (see the supplied routine `hingereg.m`). We will also consider a slightly different loss function. The "logistic loss" is defined as:

$$loss_{\log}(z, y) = \log(1 + e^{-yz}) \tag{4}$$

The logistic loss can be derived and justified from a statistical modeling perspective, but here we take it just as a surrogate for the non-convex non-smooth mis-classification loss. Unlike the hinge-loss, the logistic loss is differentiable, and thus somewhat easier to minimize using generic techniques and without access to a good linear-programming solver (it should be noted that this is not the simplest smooth surrogate one might use, but nevertheless it is very popular because of its statistical interpretation).

1. Plot the mis-classification loss $loss_{01}(z, 1)$, the hinge-loss $loss_{\text{hinge}}(z, 1)$ and the logistic loss $loss_{\log}(z, 1)$ for values of z between -4 and 4. Submit this plot. Also view the plot when the logistic loss is scaled by a factor $1/\log(2)$, and over a large range (say -50 to 50), both with and without scaling the logistic loss. There is no need to submit these additional plots.

2. Prove that for any distribution and any w:

$$R_{01}(w) \leq R_{\text{hinge}}(w)/\log(2) \tag{5}$$

where R_{01} and R_{hinge} are the expected mis-classification and hinge losses.

3. *[Optional]* Prove that for any distribution and any w:

$$R_{01}(w) \leq R_{\log}(w)/\log(2) \tag{6}$$

where $R_{\log}$ is the expected logistic loss.

The bounds (5) and (6) justify minimizing the hinge-loss or the logistic loss as a surrogate of the mis-classification loss. If we can ensure low hinge-loss or logistic loss, we can also ensure a low mis-classification loss.

However, minimizing these surrogate losses is not guaranteed to find a linear predictor that is even approximately optimal with respect to the mis-classification error.

4. Using the supplied routine `build.m`, create a small sample set of labeled points in $\mathbb{R}^2$ (i.e. $x_i \in \mathbb{R}^2$ with $\phi(x_i) = x_i \in \mathbb{R}^2$) for which $\inf_w \hat{R}_{01}(w) \leq 0.25$ but if we minimize the empirical hinge loss (or similarly the logistic loss):

$$\hat{w}_{\text{hinge}} = \arg\min_w \hat{R}_{\text{hinge}}(w) \tag{7}$$

we get many more mis-classification errors: $\hat{R}_{01}(\hat{w}_{\text{hinge}}) \geq 0.75$. Use the supplied routine `hingereg.m` to solve the minimization problem (7). Use the supplied routine `showlinear.m` to plot the sample set once with the decision boundary obtained by (7) and once with a linear decision boundary minimizing the empirical mis-classification error $\hat{R}_{01}$. Submit both plots.

Also experiment with using the supplied routine `logisticreg.m` to minimize the empirical logistic loss,

$$\hat{w}_{\log} = \arg\min_w \hat{R}_{\log}(w), \tag{8}$$

rather then hinge loss, and observe the behavior is similar (there is no need to submit any additional plots).

5. Is it possible to repeat the above exercise with a linearly separable sample set? I.e. does there exists a (finite) sample set for which $\inf_w \hat{R}_{01}(w) = 0$ by minimizing the hinge loss or logistic loss yields a predictor $\hat{w}$ with $\hat{R}_{01}(\hat{w}) > 0$? How does the optimum of (8) behave when $\inf_w \hat{R}_{01}(w) = 0$?

Problem 2: Model Selection

In this problem we will investigate using models of different complexity to learn a synthetic problem. Use the matlab command `load prob2` to load the data sets for this problem.

The 'source distribution' for this problem is a synthetically generated distribution of labeled points in $\mathbb{R}^2$ which you can view using `showdata(allx,ally)`. We will learn using predictors of the form $h(x) = \text{sign}\, f(x[1], x[2])$ where $f(x[1], x[2])$ is a polynomial in the two coordinates of x, of degree at most p. I.e. using hypothesis classes:

$$\mathcal{H}_p = \{x \to \text{sign}\, f(x[1], x[2]) \mid f \text{ is a polynomial of degree} \leq p\}$$

1. What is the VC-dimension of $\mathcal{H}_p$.
2. Using the training set of 100 labeled points given by `train100x` and `train100y`, for each p, use the supplied routines `polyexpand.m` and `hingereg.m` to find the predictor of degree at most p minimizing the empirical hinge loss. Plot the following as a function of p:
   (a) The training (empirical) hinge loss
   (b) The training (empirical) mis-classification error
   (c) The generalization hinge loss over the entire "source distribution" given by `allx` and `ally` (note that we treat here the source distribution as a (very large) discrete distribution fully specified by `allx` and `ally`).
   (d) The generalization mis-classification error on the entire "source distribution" given by `allx` and `ally`.
   (e) The bound on the generalization mis-classification error that can be obtained from the training mis-classification error.

   Repeat the above also for the larger training set of 350 points given by `train350x` and `train350y`. Submit the figures and the code used to generate them.
3. Using the supplied routine `showdata.m` (see the help documentation for the routine), plot the entire distribution, the training data, and the decision boundary for some of the values of p. Submit 3-4 of these plots that you find helpful in understanding the fitting, and over-fitting.
4. Using the entire "source distribution" `allx`,`ally`, plot the following on a single figure as a function of p:
   (a) The approximation error with respect to the hinge loss (note that this does not depend on the sample size!).
   (b) The estimation error with respect to the hinge loss, using the sample of size 100.
   (c) The estimation error with respect to the hinge loss, using the sample of size 350.

   Submit the figure and the code using to generate it.

You may choose to use logistic instead of hinge loss in this problem if you prefer.

Problem 3: Dependence of the Sample Complexity on the Dimensionality

In this problem you will validate experimentally the dependence of the number of samples required for learning on the dimensionality of the problem. We will consider the hypothesis class of linear predictors over $\mathbb{R}^d$, for varying dimensionality d, and ask how many samples m are necessary in order to ensure a low generalization error, even if there exists a zero-error predictor in the class (i.e. even if the source distribution is linearly separable).

In order to ensure the source distribution is linearly separable, you will generate the data synthetically.

First generate a d-dimensional random weight vector using the matlab command `w0 = randn(d,1)` (this generates a weight vector whose components are i.i.d. Gaussian, and so its direction is uniformly distributed). Then generate a random training set such that x is uniform over $\{-1,+1\}^d$ (e.g. using `x = sign(randn(m,d))`) and $y = \operatorname{sign} w_0' x$. Then use the supplied routine `findsep.m` to find a linear predictor $h_w(x) = \operatorname{sign} w'x$ with zero mis-classification error. Finally, generate a large (e.g. one thousand points) independent test set drawn from the same distribution and use it to evaluate the generalization error (in terms of the mis-classification loss) of the learned predictor.

For each dimensionality $d = 1..50$, repeat the above procedure several times, and for varying training set sizes, in order to find the training set size which ensures a generalization error of at most 5% with probability at least 90% over the training set.

1. Plot the required sample set size as a function of d. On the same figure, also plot the bound on the sample set which ensures a generalization error of at most 5% with probability at least 90%. Explain how you calculated the bound. Submit this figure and the code used to generate it.

   You may also want to experiment with different requirements on the generalization error and the probability of failure, and perhaps plot the generalization error as a function of sample size and/or dimensionality. However, there is no need to submit these.

2. Use the data in the plot to find a model for the required sample size as a function of the dimensionality. E.g. try to fit a linear, polynomial or perhaps linear times logarithmic, function to the data. Discuss the relationship between the required sample size and the dimensionality suggested by the experiment.

3. What does the experiment tell us about our learning bound? In what way is it tight? In what way is it loose? Does the experiment show that the bound can be improved?