9.520 Problem Set 2

Due April 25 , 2011

Note: there are five problems in total in this set.

Problem 1 In classification problems where the data are unbalanced (there are many more examples of one class than of the other one) a common strategy is to *weight* the loss function so that the errors in one class are counted more than those of the other class. In the case of RLS, this corresponds to solving the following problem

$$\min_{w\in\mathbb{R}^d}\left\{\sum_{i=1}^{n}\gamma_i\big(\langle w,x_i\rangle-y_i\big)^2+\lambda\|w\|^2\right\}$$

where $\sum_{i=1}^{n}\gamma_i=1$ and $\gamma_i>0$ for all $i=1,\ldots,n$.

(a) Derive the explicit form of the minimizer w^* of the above problem.

(b) Consider the case where we have a weighted loss function and also an offset $b\in\mathbb{R}$,

$$\min_{w\in\mathbb{R}^d,\, b\in\mathbb{R}}\left\{\sum_{i=1}^{n}\gamma_i\big(\langle w,x_i\rangle+b-y_i\big)^2+\lambda\|w\|^2\right\}$$

where $\{\gamma_i : i=1,\ldots,n\}$ are the weights of a convex combination (that is, $\sum_{i=1}^{n}\gamma_i=1$ and $\gamma_i>0$ for all $i=1,\ldots,n$). Using the results in PSET 1 and the above result, derive the explicit form of the minimizers w^*, b^* for the problem above.

(c) What do you think could be a good (optimal?) way to choose the weights?
Note that this last question is somewhat open-ended. There is not necessarily a right or wrong answer. We are just interested in seeing what ideas you have.

Problem 2 In this problem we will consider a special case of sparsity-based regularization, and show that the solution can be written in closed form. Consider the minimization problem:

$$\min_{w\in\mathbb{R}^d}\left\{\sum_{i=1}^{n}\big(\langle w,x_i\rangle-y_i\big)^2+2\lambda\|w\|_1\right\}$$

where $\langle w,x_i\rangle=\sum_{j=1}^{d}w^jx_i^j$ and $\|w\|_1=\sum_{j=1}^{d}|w^j|$. Assume that

$$\sum_{i=1}^{n}x_i^jx_i^k=\delta_{j,k}\qquad(\text{i.e.}\ \sum_{i=1}^{n}x_ix_i^T=I_{n\times n})$$

(a) Show that the minimizer w_* of the above problem can be written component-wise in closed form as

$$w_*^j = S_{\lambda n}(y^j)$$

where $y^j = \sum_{i=1}^n y_i x_i^j$ and

$$S_{\lambda n}(y^j) = y^j \max\{0, 1 - \frac{\lambda}{|y_j|}\}. \tag{1}$$

(Hint: Solve by computing the partial derivative of the functional w.r.t. a component w^j and setting it equal to zero. Note that you can compute the derivative of $|x|$ splitting in 3 cases: $x > 0$, $x < 0$ and $x = 0$. Also note that $\text{sign}(a) = a/|a|$.)

(b) Discuss what would change if we consider

$$\min_{w \in \mathbb{R}^d,\, b \in \mathbb{R}} \left\{ \sum_{i=1}^n \left(\langle w, x_i \rangle + b - y_i \right)^2 + 2\lambda \|w\|_1 \right\}.$$

Problem 3 Here we will consider the problem of transductive learning. We will derive the explicit solution to two different learning algorithms and compare them on a toy data set. We assume to be given a set of labeled examples $(x_i, y_i)_{i=1}^{\ell}$ and a set of unlabeled examples $(x_i)_{i=1}^{u}$. Let $m = \ell + u$. We assume that the indices of the input points are ordered so that the first ℓ points have labels.
The goal is to predict the label of such an unlabeled set[1].
We consider two related schemes to do this.

First, we need to introduce a few concepts. We will assume a $m \times m$ (symmetric) weight matrix W to be given, such that the similarity between two input points x_i, x_j is given by $W_{i,j}$. Then, the "smoothness" of a function on the input points can be written as

$$R(f) = \frac{1}{2} \sum_{i,j=1}^{m} W_{i,j} (f(x_i) - f(x_j))^2.$$

(a) Prove that

$$R(f) = \mathbf{f}^T L \mathbf{f} = R(\mathbf{f})$$

where $L = D - W$, D is the diagonal matrix such that $D_{ii} = \sum_{j=1}^m W_{i,j}$ and $\mathbf{f} \in \mathbb{R}^m$ with $\mathbf{f}^i = f(x_i)$, $i = 1, \ldots, m$.

(b) Consider the algorithm

$$\min_{\mathbf{f} \in \mathbb{R}^m} R(\mathbf{f}),$$

subject to the constraint

$$f(x_i) = y_i, \quad i = 1 \ldots, \ell.$$

Find the vector $\mathbf{f}_*$ which solves the above problem.

[1] In contrast to semi-supervised learning where we might be interested in predicting labels of previously-unseen points.

(c) Consider the algorithm

$$\min_{\mathbf{f}\in\mathbb{R}^m} R(\mathbf{f}) + \lambda \sum_{i=1} (y_i - \mathbf{f}^i)^2.$$

Find the vector $\mathbf{f}_*^\lambda$ which solves the above problem.

(d) Design and perform an experiment to compare the behavior of the two algorithms on the two moon datasets.

(Hint for item b: write L as a block matrix with block $L_{\ell,\ell}, L_{u,\ell}, L_{\ell,u}, L_{u,u}$
Hint for item c: write the vector of labels as $J\mathbf{y}'$ where J is the diagonal matrix with ℓ ones on the diagonal and then zeros, and $\mathbf{y}' = (y_1, \ldots, y_\ell, 0, \ldots, 0) \in \mathbb{R}^m$.

Problem 4 In this problem, we will look at gradient descent first as a regularized algorithm based on early-stopping, and then as a tool for solving RLS problems. We also recall the successive approximation scheme associated to a contractive map.

Recall that a contractive map T is such that $\|T(c) - T(c')\|_2 \leq L\|c - c'\|_2$ for all c, c', with $L < 1$. By the Banach fixed-point theorem (see e.g. [1] or [2]), every contractive map has a unique fixed point: a point $c^* \in \mathbb{R}^n$ such that

$$c^* = T(c^*),$$

Then, since $\mathbb{R}^n$ is complete, it is easy to show that the iteration

$$c^{(i+1)} = T(c^{(i)}) \tag{2}$$

with $c^{(0)} = 0$, converges to the fixed point c^* for $i \to \infty$ (where superscripts denote iterates).

We have seen that the solution of ERM on a RKHS can be written as $f = \sum_{i=1}^n c_i K(\cdot, x_i)$ where the x_i belong to the training set. Then the ERM problem can be written as

$$\min_{c\in R^n} \|Y - \mathbf{K}c\|_2^2 \tag{3}$$

where $\mathbf{K}$ is the $(n \times n)$ kernel matrix and c, Y are the (n-dimensional) vectors of coefficients and labels respectively.

Let the operator-norm $\|\mathbf{K}\|$ of a matrix be the maximum absolute eigenvalue of K. By assuming that $n \geq \|\mathbf{K}\| =$ (for instance, this is always true for the Gaussian kernel), we can ensure convergence of the iterative procedures described next.

(a) Prove that c^* minimizes the empirical risk in (3), if and only if it satisfies $c^* = T(c^*)$ with

$$T(c) := c - \frac{1}{n}(\mathbf{K}c - Y). \tag{4}$$

(b) Implement in Matlab the iteration (2) for the particular map given in (4). Construct your own toy dataset, and show empirically that if the number of iterations is chosen appropriately (that is, if we choose a good early stopping point), we can avoid overfitting. Use the Gaussian kernel with sigma equal to the average distance between the points in the training set. We could use other kernels of course, but the Gaussian kernel will conveniently allow us to define a stable learning rule of the form chosen in (4).

(c) Now recall the Tikhonov regularization problem

$$\min_{c\in\mathbb{R}^n}\{\|Y-\mathbf{K}c\|_2^2+\lambda c^t\mathbf{K}c\}. \tag{5}$$

Show that c_λ^* solves the problem (5) if and only if it also satisfies $c_\lambda^* = T(c_\lambda^*)$ with

$$T_\lambda(c) := c - \frac{1}{n+\lambda}((\mathbf{K}+\lambda I)c - Y). \tag{6}$$

(d) Implement in Matlab the iteration (2) for the map (6), and evaluate its behavior on the same dataset you used in (b). Discuss the (empirical) interplay between the number of iterations and the parameter λ, and provide evidence (plots etc.) to support your claims. For all experiments, use the Gaussian kernel with sigma chosen to be the average distance between the points in the training set.

Problem 5 In this excercise you will implement and use regularized least squares with an unregularized bias term and weighted loss function, in an artificial classification problem with unbalanced classes. You should use your solution for Problem 6 from HW1 in this problem.

Weighted RLS. Modify your function `rlsTrain`, from the previous problem set, to take a new parameter `gamma`, which is a vector of weights to use in weighting the loss function, and to use an unregularized bias term. More specifically:

- `rlsTrain(Ytrain,Xtrain,whichKernel,useBias,gamma,whichValidation)` takes six inputs:
  - `Ytrain` the training labels
  - `Xtrain` the training inputs
  - `whichKernel` the kernel to use, e.g. `'linear'`; for this problem, *your function need only implement the weighted loss and bias term for the linear kernel*
  - `useBias` true if you will use an unregularized bias term, false otherwise;
  - `gamma` the vector of weights on the corresponding terms in the loss function, of the same length as the number of training inputs
  - `whichValidation` the type of cross-validation to use in choosing λ; this can be `'loo'` or `'kfold'`

  and returns four outputs

- `coeffs` the optimal RLS coefficients
- `b` the optimal RLS bias
- `lambdas` a vector of values tried for the regularization parameter λ
- `cverr` a vector of error values on the cross-validation sets, one for each value of λ in `lambdas`

Validation. In HW1, we chose λ using leave-one-out cross validation. Here we will do k-fold cross validation. In `rlsTrain`, implement the following:

- When `'kfold'` cross-validation is specified:
  - Split the training set into k pairs of training and validation sets, according to a justifiable scheme. *Be sure to explain your scheme for choosing the validation splits.*
  - Record the squared error for each λ, incorporating the weights (`gamma`) when appropriate.
- Choose λ that minimizes the CV error.
- Throw an error on attempting to do leave-one-out CV when using weights or unregularized bias. (The expression for the leave-one-out predicted values, given weights and/or bias, differs from the one given in lecture.)

Weights. If n, n_-, n_+ are the total number of points, and the number of points in class -1 and 1, respectively, you can choose the weights for the weighted loss to be $\gamma_+ = n_-/n$ for class 1 and $\gamma_- = n_+/n$ for class -1.

Experiments. We want to see the effect of the un-regularized bias term and the weights on the loss function, individually and combined, when the classes are unbalanced. The dataset is linked on the course website. The dataset is drawn from a distribution in which the probability of drawing a point from the negative class is 0.05.

(a) You should produce one table of the following form:

		misclassified %		
	best λ	(pos. class)	(neg. class)	(all)
Linear RLS				
Linear RLS w/bias term				
Linear RLS w/weights				
Linear RLS w/weights & bias				

The last three columns give the percentages of positive/negative/all test examples misclassified.

(b) You should also produce one figure:

- Plot the test set and the decision boundaries for the different variants of RLS (with/without bias, with/without weights). Use the `legend` function to label the different decision boundaries. Matlab's `contour` function might be helpful here.

(c) Describe in one or two sentences the effects of the weighting scheme and bias on:

i. the decision boundaries;

ii. the misclassification error.

References

[1] J. Hunter and B. Nachtergaele, *Applied Analysis*, World Scientific, 2001. (available online – see ch.3: http://www.math.ucdavis.edu/∼hunter/book/pdfbook.html)

[2] A. N. Kolmogorov and S. V. Fomin. *Elements of the Theory of Functions and Functional Analysis*, Dover Publications, 1999.

[3] V. N. Vapnik. Statistical Learning Theory. Wiley, 1998.