

CS195-5: Introduction to Machine Learning
Fall 2006

Problem Set #3 : Apricot

Out: October 16, 2006

Due: October 25, 2006, 11am EDT

Instructions

In order to identify problem sets to the automated submission system we will assign code names to each problem set. This one will be called “apricot”.

How and what to submit? Please submit your solutions using the automated system in the following way.

1. Create a directory (say, `ps3`) in which you will work on your solutions. All files mentioned below are assumed to reside in that directory.
2. Typeset the written parts of your solution in \LaTeX . The final results should be a document in PDF (preferable) or in PostScript. please name this document `apricotsolution.pdf` or `apricotsolution.ps`, according to the format. **Important: please start the solution of every problem with `\newpage`.**
3. Create Matlab code (extension `.m`) and data (`.mat`) files needed.
4. Create a file named `README`, and include in it a short description of all Matlab files you are submitting.
5. **From the directory `ps3`**, run
`/course/cs195-5/bin/cs195-5handin apricot`

You can resubmit your work as many times as needed, and each subsequent version will override the previous one.

Late submissions: there will be a penalty of 25 points for any solution submitted past the deadline until 11am on Friday, October 27. No submissions will be accepted past then.

What is the required level of detail? When asked to derive something, please clearly state the assumptions, if any, and strive for balance: justify any non-obvious steps, but try to avoid superfluous explanations. When asked to plot something, please include the figure as well as the code used to plot it; if multiple entities appear on a plot, make sure that they are clearly distinguishable (by color or style of lines and markers). When asked to provide a brief explanation or description, try to make your answers concise, but do not omit anything you believe is important.

1 Regularization

Problem 1 [15 points]

Show that the solution for ridge regression problem

$$\hat{\mathbf{w}}_{ridge} = \underset{\mathbf{w}}{\operatorname{argmax}} - \sum_{i=1}^N (y_i - \mathbf{w}^T \mathbf{x}_i)^2 - \lambda \sum_{j=0}^d w_j^2$$

is given by $\hat{\mathbf{w}}_{ridge} = (\lambda \mathbf{I} + \mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$.

End of problem 1

Advice: Note that the penalty above includes, for simplicity, the constant term coefficient w_0 .

Recall how we derived the solution to unregularized least-squares in Lecture 2, and follow the derivation while correcting for the new penalty term. Reviewing the matrix identities tutorial (posted with Lecture 2) may be useful.

In the next problem we will look at the effect of L_2 regularization on the bias-variance tradeoff in regression. The experiment will involve data drawn from the model

$$y = \sin(x) + \nu$$

where ν is additive Gaussian noise drawn from $\nu \sim \mathcal{N}(\nu; 0, 0.1)$. The experiment will be based on the code available on the Web as `regularizationCode.m`; you will need to fill in a few missing pieces, and also to download the auxiliary function `genData.m` that generates random sets of data drawn from the above distribution, with $x \in [-1, 1]$.

Problem 2 [25 points]

Fill in the missing parts in the provided code, run it, and turn in the resulting

figure with four lines. Explain the behavior of the lines. In particular, explain the relationship between the plot of bias²+variance and the plot of the test error (try to provide both a quantitative and a qualitative explanation).

End of problem 2

Advice: Recall the decomposition of expected loss for regression in Lecture 9.

Finally, let us take a look at the effect of regularization on classifiers. Suppose now we are working with a quadratic logistic regression in 2D, i.e.,

$$p(1|\mathbf{x}) = 1 / (1 + \exp(-w_0 - w_1x_1 - w_2x_2 - w_3x_1^2 - w_4x_2^2)).$$

Problem 3 [20 points]

Consider the following regularized logistic regression objective:

$$\operatorname{argmax}_{\mathbf{w}} \sum_{i=1}^N \log p(y_i|\mathbf{x}_i) - \lambda (w_3^2 + w_4^2).$$

Modify the logistic regression code from Problem Set Nectarine to reflect this new objective. Train logistic regression on the data in `lrDataApricot.mat` with the following values of λ : 0 (no regularization), 5, 20, 100, 500. Plot the decision boundaries for each value of λ . What can you say about the effect of the regularization (with increasing λ) on the geometry of the decision boundary?

End of problem 3

Advice: One modification you will need is of course adding the penalty-related term to the gradient of the objective. Another change involves the Hessian. Note that the computation of Hessian in `logisticRegression.m` from PS Nectarine already includes a regularization term \mathbf{I} in order to ensure numerical stability. Change that expression to

```
hess = -(eye(d)+diag([zeros(1,3) lambda*ones(1,2)]))+ Z'*X);
```

Finally, please change the tolerance (threshold on change in error) to `1e-4` to ensure that the gradient descent doesn't stop prematurely.

You can use the function `plotLRcont.m`, available from the Web, to plot the boundaries. Type `help plotLRcont` to see an explanation on its arguments.

2 Support Vector Machines

Problem 4 [20 points]

Let (x_i, y_i) , $i = 1, \dots, N$ be the training examples and their labels, and α_i , $i = 1, \dots, N$ the Lagrange multipliers found by solving the SVM optimization problem. Recall that the SVM classifier is given by

$$\hat{y}(\mathbf{x}) = \text{sign} \left(w_0 + \sum_{\alpha_i > 0} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} \right).$$

Write down the expression for the optimal term w_0 .

End of problem 4

Advice: By definition of SVM, the value of w_0 is the one that maximizes the margin. There is a number of slightly different ways of deriving this. They all rely on the definition of margin and the definition of support vectors, and while they are all equivalent in theory, in practice some may be more robust numerically than others. Try to find the most robust solution.

If there is no test data available, we may want to estimate the risk of the classifier by computing the leave-one-out cross-validation error. That is,

$$\epsilon_N = \frac{1}{N} \sum_{i=1}^N L(\hat{y}_{-i}(\mathbf{x}_i), y_i)$$

where L is the zero-one loss function, and \hat{y}_{-i} is the prediction by the SVM trained on all the data *except* the i -th example. Turns out we can say something about this estimate even without explicitly computing it!

Problem 5 [20 points]

Suppose that after training an SVM on a data set of N examples, we end up with m support vectors (i.e., m out of N examples have non-zero coefficients). Let ϵ_N be the leave-one-out estimate of SVM risk on the same training data set. Show that

$$\epsilon_N \leq \frac{m}{N}.$$

End of problem 5