# CPSC 540 Assignment 3 (due October 1st)

Ridge Regression, Kernels, and $\ell_1$-Regularization

Please put your name and student number on the assignment, and staple the submission together.

## 1 Generalized Ridge Regression

Recall that the ridge regression esitmator is the solution to

$$\mathop{\arg\min}_{w \in \mathbb{R}^d} \frac{1}{2}\|Xw - y\|^2 + \frac{\lambda}{2}\|w\|^2,$$

which can be written as

$$w = (X^TX + \lambda I)^{-1}X^Ty.$$

In class, we showed that this is equivalent to a MAP estimator under the assumption

$$y_i|x_i, w \sim \mathcal{N}(w^Tx_i, 1), \quad w_j \sim \mathcal{N}(0, \lambda^{-1}),$$

where we have parameterized the Gaussian distributions in terms of the precision parameter $\lambda = 1/\sigma^2$. If have prior knowledge about the scale of the variables (e.g., some $w_j$ are expected to be small while some are expected to be large), we might consider using a different $\lambda_j$ for each variable $w_j$. In other words, we would assume

$$y_i|x_i, w \sim \mathcal{N}(w^Tx_i, 1), \quad w_j \sim \mathcal{N}(0, \lambda_j^{-1}).$$

Derive the MAP estimate for $w$ under these assumptions. (Hint: you will need to use a diagonal matrix $\Lambda$, where the entries along the diagonals are the corresponding $\lambda_j$.)

## 2 Equivalence of Ridge Regression Estimators

When deriving the kernelized form of ridge regression, we used the equivalence

$$\hat{X}(X^TX + \lambda I)^{-1}X^TY = \hat{X}X^T(XX^T + \lambda I)^{-1}Y.$$

Use a variant of the matrix inversion lemma (MLAPP Corollary 4.3.1) to show this equality holds. You may find it helpful to use the identity $(\alpha A)^{-1} = \frac{1}{\alpha}A^{-1}$, where $\alpha \neq 0$ and $A$ is invertible.

## 3 Using a Validation Set for Kernel Selection

The function *kernelDemo* loads and displays a training data set $\{X, y\}$ and a corresponding test data set $\{Xtest, ytest\}$. Subsequently, after each time you press a key, it shows the result of using the polynomial kernel with a different order $m$ of the polynomial. (Notice that the fit gets better as we increase the order, but eventually the polynomial becomes unstable and we overfit). After this, the demo will cycle through various values of $\lambda$ and $\sigma$ for the radial basis function (RBF) kernel.

In this demo, the test errors are reported but in practice you will need to estimate the test error from the training data alone. Make the following modifications to the demo:

1. Instead of training on the full training set and testing on the test set. Train of the first half of the training data and test on the second half of the training data (this second half is be called the 'validation' set).
2. The 'hyper-parameters' that this demo searches over are the order $m$ for the polynomial kernel, and the parameters $\{\lambda, \sigma\}$ for the RBF kernel. For each kernel, keep track of the hyper-parameter(s) that yields the smallest error on the validation set (use absolute error on the validation set).
3. Once you have the best value(s) of the hyper-parameter(s) for each kernel, compute the absolute error (still training on only the first half of the data) on the test set.

Hand in your code and report the best value of $m$ (polynomial kernel) and $\{\lambda, \sigma\}$ (RBF kernel) as well as their test error. (Please remove the plotting sections from your code.)

# 4 Least Squares with $\ell_1$-Regularization

In class we discussed the $\ell_1$-regularized least squares estimator,

$$\operatorname*{arg\,min}_{w \in \mathbb{R}^d} \frac{1}{2}\|Xw - y\|^2 + \lambda \|w\|_1,$$

where $\|w\|_1$ is the $\ell_1$-norm of $w$, $\|w\|_1 = \sum_{i=1}^d |w_i|$. Derive the form of the distribution for the likelihood and prior that lead to this MAP estimate, as we did in class for $\ell_2$-regularized least squares.

Re-write the non-smooth optimization problem as a linearly-constrained smooth optimization problem, as we did in class for $\ell_1$-regression (this should result in a "quadratic program", meaning that objective function is a quadratic and the constraints are linear).

# 5 Shooting Algorithm

The $\ell_1$-regularized least squares algorithm is sometimes called the LASSO (least absolute shrinkage and selection operator), and one of the first algorithms proposed for solving this problem is the 'shooting' algorithm (Algorithm 13.1 of MLAPP). On each iteration of this algorithm, it chooses one variable and sets it to its optimal value given the values of the other variables, an example of *coordinate descent*. Normally, coordinate descent doesn't work for non-smooth problems but here it works because the non-smooth term is *separable* (meaning that it can be written as the sum of a function applied to each variable, $\lambda\|w\|_1 = \sum_{j=1}^d \lambda |w_j|$). The algorithm can be started from any initialization, and any strategy for choosing the variable to update works as long as you guarantee that they are all updated periodically. A common termination criterion is that none of the values changed by more than a small tolerance $\epsilon$ in their last update.

Write a function *shooting(X,y,lambda)* that implements the shooting algorithm, and hand in this function. You do not have to derive/understand the the update for each variable.

We call the plot of the MAP values of $w$ as a function of $\lambda$ the *regularization path*. The function *regPathDemo* shows part of the regularization path when using $\ell_2$-regularization for the *housing* data set (in this dataset $y$ is the value of that a home sold for and $X$ are features that may affect its market value). Modify this function to use your shooting function instead of the $\ell_2$-regularized MAP estimate. Hand in your plot of this part of the $\ell_1$-regularization path.