- We can represent θ as a linear combination of the $\phi(x^{(i)})$. We do this by letting:

$$\theta^{(i)} = \sum_{k=1}^{i} \alpha^{(k)} y^{(k)} \phi(x^{(k)})$$

  Where $\alpha^{(k)}$ is the number of times $x^{(k)}$ has been misclassified.

- Using our implicit representation of $\theta^{(i)}$, we get

$$h_{\theta^{(i)}}(\phi(x^{(i+1)})) = sign(\theta^{(i)^T} \phi(x^{(i+1)})) = sign\Big(\big(\sum_{k=1}^{i} \alpha^{(k)} y^{(k)} \phi(x^{(k)^T})\big)\phi(x^{(i+1))})\Big)$$

$$h_{\theta^{(i)}}(\phi(x^{(i+1)})) = sign\Big(\sum_{k=1}^{i} \alpha^{(k)} y^{(k)} \phi(x^{(k)^T})\phi(x^{(i+1))})\Big)$$

$$h_{\theta^{(i)}}(\phi(x^{(i+1)})) = sign\Big(\sum_{k=1}^{i} \alpha^{(k)} y^{(k)} K(x^{(k)}, x^{(i+1)})\Big)$$

  Thus, assuming that $K(x^{(k)}, x^{(i+1)})$ access is $O(1)$, then this operation is $O(m)$

- First, we evaluate the prediction $h_{\theta^{(i)}}(\phi(x^{(i+1)}))$, and if $h_{\theta^{(i)}}(\phi(x^{(i+1)})) \neq y^{(i+1)}$, then the update rule is $\alpha^{(i+1)} \leftarrow \alpha^{(i+1)} + 1$

- We have that $\phi(x^{(1)})^T = (1,0,0), \phi(x^{(2)})^T = (1,2,2), y^{(1)} = -1, y^{(2)} = 1$, which gives us a vector parallel to θ:
$$\theta_p^T = (\phi(x^{(1)}) - \phi(x^{(2)}))^T = (0,2,2)$$
- $$\frac{1}{2}||\theta_p|| = \frac{1}{2}\sqrt{0+4+4} = \sqrt{2}$$
- We know that $k\theta_p = \theta$ for some k, so we know that:
$$\sqrt{2} = \frac{1}{||\theta||} = \frac{1}{||k\theta_p||} = \frac{1}{\sqrt{8k^2}}$$
$$\sqrt{2} = \frac{1}{2k\sqrt{2}}$$
$$k = \frac{1}{4}$$
$$\theta^T = (0, 1/2, 1/2)$$
- Because the inequalities are tight, we get two equations with one variable to solve for, θ_0. The first equation is:
$$-1([0, \frac{1}{2}, \frac{1}{2}][1,0,0]^T + \theta_0) = 1$$
$$[0, \frac{1}{2}, \frac{1}{2}][1,0,0]^T + \theta_0 = -1$$
$$0 + \theta_0 = -1$$
Likewise, the second equation is
$$[0, \frac{1}{2}, \frac{1}{2}][1,2,2]^T + \theta_0 = 1$$
$$2 + \theta_0 = 1$$
$$\theta_0 = -1$$
- The equation for decision boundary is
$$y = \theta^T\phi(x) + \theta_0$$
$$y = (0, 1/2, 1/2)(1, x\sqrt{2}, x^2)^T - 1$$
$$y = \frac{x^2 + x\sqrt{2} - 1}{2}$$

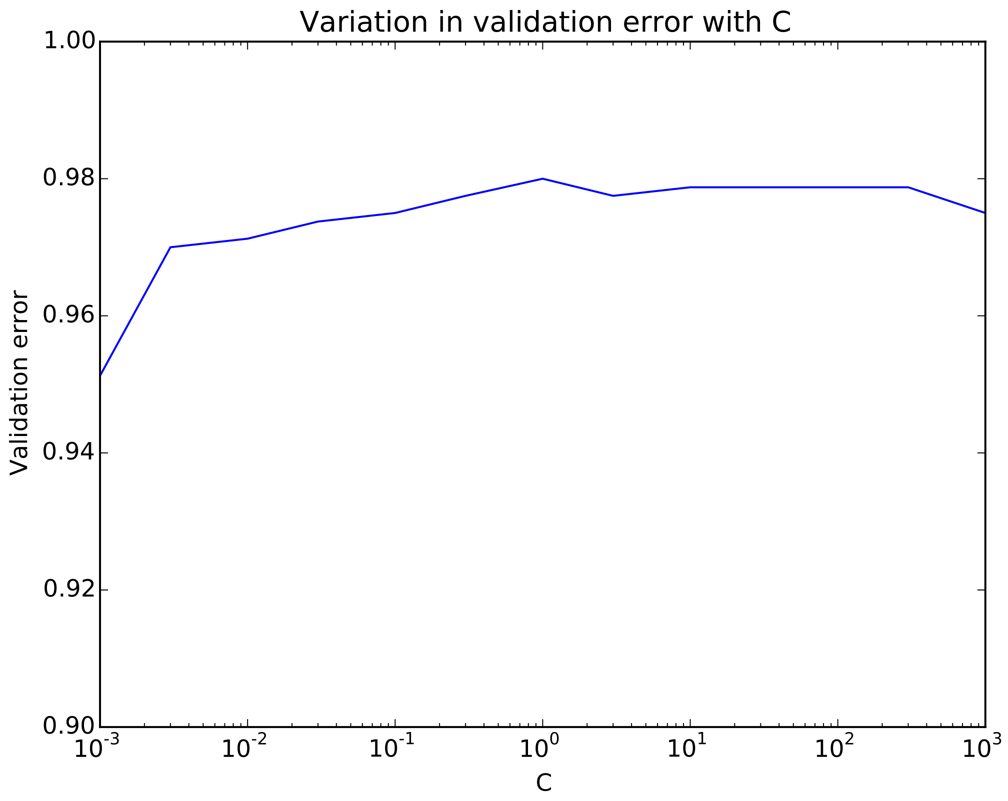
- Training SVM's for Spam Classifications:
  We reasoned that there was no reason to scale the features when we were running experiments without kernelizing, since the feature vector contained only 0's and 1's and thus was already scaled (and in a very meaningful way) and that there was no reason to not scale when kernelizing, since the post-kernlized data features would likely have scales that varied widely, which would affect the convergence time/accuracy of the model. Thus we didn't run any experiments with those configurations (no kernel + scaled or kernel + no scale). We ran an extensive hyper-parameter grid search (using a validation set to test the hyper-parameters) for 5 following different SVM configurations and received the following optimal hyperparamters and accuracies:

  - No Kernel, No Scaling: Training Accuracy (0.9935), Test Accuracy (0.984)
    * Hyperparameters: Learning Rate=1.0, Iterations=100, C=1
  - Gaussian Kernel, Scaling: Train Accuracy (0.8975), Test Accuracy (0.8975)
    * Hyperparameters: Learning Rate=0.01, Iterations=100, C=0.1, Sigma=10
  - Polynomial Kernel Degree 1, Scaling: Train Accuracy (0.918), Test Accuracy (0.921)
    * Hyperparameters: Learning Rate=0.01, Iterations=100, C=0.1, c=-10
  - Polynomial Kernel Degree 2, Scaling: Train Accuracy (0.96325), Test Accuracy (0.96)
    * Hyperparameters: Learning Rate=0.01, Iterations=100, C=0.1, c=-10
  - Polynomial Kernel Degree 3, Scaling: Train Accuracy (0.973), Test Accuracy (0.959)
    * Hyperparameters: Learning Rate=0.1, Iterations=100, C=0.1, c=-10

  For all the above models that used scaled features, we used Standard Scaling (Zero Mean, Unit Variance), as we figured that this should do a reasonable job. We also experimented with scaling the data from a vector of 0's and 1's to a vector of -.5's and .5's, thinking that this could improve the results of the polynomial kernels, but it didn't. We would've attempted to use other types of scaling; however, some of the other scaling methods (like log scaling) did not make much sense given the type of problem/type of data, and furthermore, the kernelized models (the only ones that used scaling) were not promised enough, compared to the no-kernel, no-scaling model, so we did not attempt these other types of scaling.

  With the best model, we achieved 99.35% accuracy on the training data and 98.4% accuracy on the test data, and we found that the top 15 predictors of spam were: out, replac, client, william, plu, wealth, domain, yourself, guess, benefit, aa, format, monitor, hermio, hous. Note that the best kernel models are the polynomial kernels with degree 2 or 3 and c=-10 (different from C).

  The hyperparameters for the optimal no-kernel, no-scaling model that we found were Learning Rate=1.0, Number of Iterations=100, and C=1. Here is a graph that backs up our hyperparameter selection for the No Kernel Model (i.e. the best model):



This graph plots the best accuracy (found over a grid search over the two other hyperparameters, the learning rate and the number of iterations) for every value of C. As we can see, the peak of the plot is at C=1, so C=1 is the best C value for the No Kernel SVM Configuration. The hyperparameters that give the best accuracy when C=1 are Learning Rate=1.0 and Iterations=100, so we have that the best hyperparamters are Learning Rate=1.0, Number of Iterations=100, and C=1, with No Kernel.