CS/ECE/ME 532
Homework 7: Convexity and the SVM

1. **Verifying convexity**. Prove that the following functions are convex.

   a) The sum of two convex functions: $f(\boldsymbol{x}) = g(\boldsymbol{x}) + h(\boldsymbol{x})$ where g and h are convex.

   b) A positive quadratic form: $f(\boldsymbol{x}) = \boldsymbol{x}^\mathsf{T}\boldsymbol{P}\boldsymbol{x}$, where $\boldsymbol{P} \succ 0$.

   c) The pointwise maximum of several affine functions: $f(\boldsymbol{x}) = \max\limits_{i\in\{1,\dots,m\}} (\boldsymbol{a}_i^\mathsf{T}\boldsymbol{x} + b_i)$

   d) The definition of convexity we saw in class may also be extended to functions that take a matrix as an argument, e.g. $f : \mathbb{R}^{m\times n} \to \mathbb{R}$. Prove that $f(\boldsymbol{X}) = \|\boldsymbol{X}\|_2$ (the induced 2-norm) is convex.

2. **Gradient Descent and Stochastic Gradient Descent**. Suppose we have training data $\{\boldsymbol{x}_i, y_i\}_{i=1}^m$, with $\boldsymbol{x}_i \in \mathbb{R}^n$ and y_i is a scalar label. Derive gradient descent and SGD algorithms to solve the following ℓ_1-loss optimization:

$$\min_{\boldsymbol{w}\in\mathbb{R}^n} \sum_{i=1}^m \left| y_i - \boldsymbol{x}_i^\mathsf{T}\boldsymbol{w} \right| .$$

   a) Simulate this problem as follows. Generate each x_i as random points in the interval $[0, 1]$ and generate $y_i = w_1 x_i + w_2 + \epsilon_i$, where w_1 and w_2 are the slope and intercept of a line (of your choice) and $\epsilon_i =$ `randn`, a Gaussian random error generated in Matlab. With $m = 10$. Repeat this experiment with several different datasets (with different random errors in each case).

   b) Implement the GD or SGD algorithm for the ℓ_1-loss optimization. Compare the solution to this optimization with the LS line fit.

   c) Now change the simulation as follows. Instead of generating ϵ_i as Gaussian, now generate the errors according to a Laplacian (two-sided exponential distribution) using `laprnd(1,1)`. Compare the LS and ℓ_1-loss solution compare in this case. Repeat this experiment with several different datasets (with different random errors in each case).

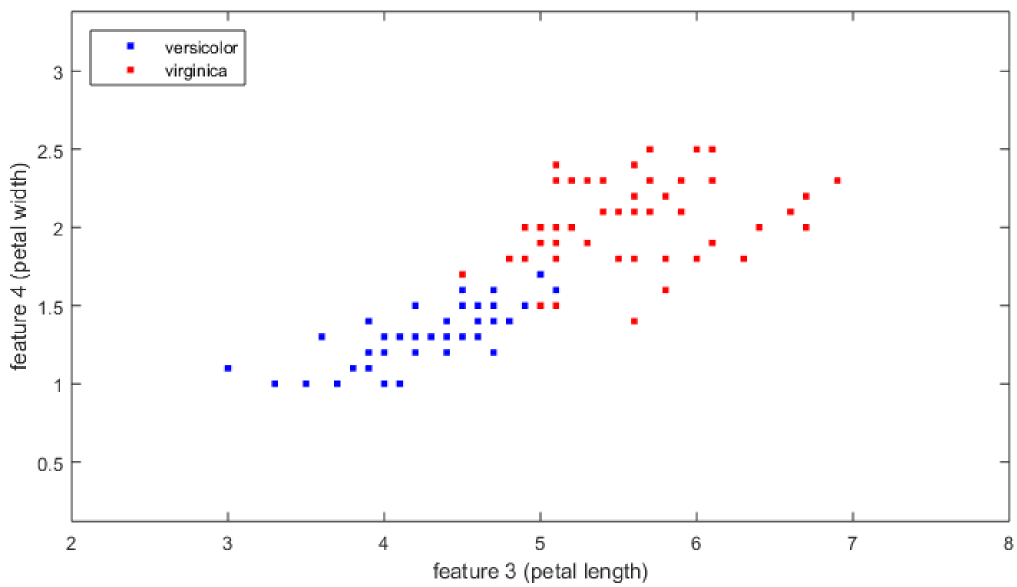
3. **Error Bounds using Hinge Loss**. State the SGD algorithm for solving the hinge-loss optimization

$$\min_{\boldsymbol{w}} \sum_{i=1}^m f_i(\boldsymbol{w}) \qquad \text{where:} \qquad f_i(\boldsymbol{w}) = \left(1 - y_i \boldsymbol{x}_i^\mathsf{T}\boldsymbol{w}\right)_+ .$$

   a) Derive a bound on the average error $\frac{1}{T}\sum_{t=1}^T \left(f_{i_t}(\boldsymbol{w}_t) - f_{i_t}(\boldsymbol{w}^*)\right)$ using Theorem 1 from the lecture notes (on moodle). Assume that $\boldsymbol{w}_1 = \boldsymbol{0}$ and $\|\boldsymbol{w}^*\| \le 1$, and that the features are normalized so that $\|\boldsymbol{x}_i\| \le 1$ for all i. Assume a constant stepsize of $\gamma = 1/\sqrt{T}$ as in Corollary 1.

   b) How many iterations are required to guarantee that the average error is less than 0.01?

4. **Classification and the SVM**. Revisit the `iris` data set from Homework 3. For this problem, we will use the 3rd and 4th features to classify whether an iris is *versicolor* or *virginica*. Here is a plot of the data set for this restricted set of features.



We will look for a linear classifier of the form: $x_{i3}w_1 + x_{i4}w_2 + w_3 \approx y_i$. Here, x_{ij} is the measurement of the j^{th} feature of the i^{th} iris, and w_1, w_2, w_3 are the weights we would like to find. The y_i are the labels; e.g. $+1$ for *versicolor* and -1 for *virginica*.

a) Reproduce the plot above, and also plot the decision boundary for the least squares classifier.

b) This time, we will use a regularized SVM classifier with the following loss function:

$$\underset{\boldsymbol{w}}{\text{minimize}} \quad \sum_{i=1}^{m} (1 - y_i \boldsymbol{x}_i^\mathsf{T} \boldsymbol{w})_+ + \lambda(w_1^2 + w_2^2)$$

Here, we are using the standard hinge loss, but with an ℓ_2 regularization that penalizes only w_1 and w_2 (we do not penalize the offset term w_3). Solve the problem by implementing gradient descent of the form $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \gamma \nabla f(\boldsymbol{w}_t)$. For your numerical simulation, use parameters $\lambda = 0.1$, $\gamma = 0.003$, $\boldsymbol{w}_0 = \boldsymbol{0}$ and $T = 20{,}000$ iterations. Plot the decision boundary for this SVM classifier. How does it compare to the least squares classifier?

c) Let's take a closer look at the convergence properties of $\boldsymbol{w}_t$. Plot the three components of $\boldsymbol{w}_t$ on the same axes, as a function of the iteration number t. Do the three curves each appear to be converging? Now produce the same plots with a larger stepsize ($\gamma = 0.01$) and a smaller stepsize ($\gamma = 0.0001$). What do you observe?