1: Margins

(1) Consider the following input to the XOR function in two dimensions (x_1, x_2) and the input mapped into the space $(x_1, x_1 x_2)$.

x_1	x_2	label	$x_1 x_2$
-1	-1	-1	1
-1	1	1	-1
1	-1	1	-1
1	1	-1	1

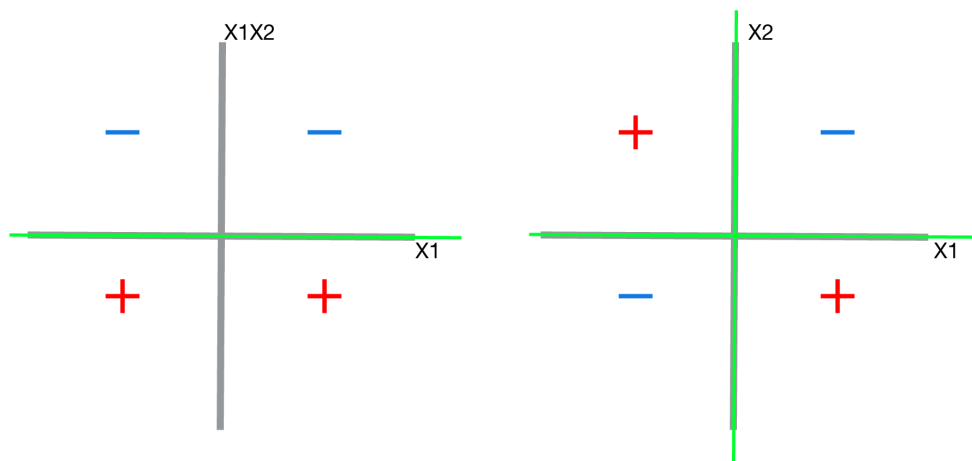Table 1: Inputs to the XOR function in two dimensions and mapping.

Figure 1: The separating line producing the maximal margin in both spaces.

The XOR function in $x_1, x_1 x_2$ space has a maximum margin of 1. This is achieved with the separating line $x_1 x_2 = 0$.

Translating the line $x_1 x_2 = 0$ back to the original x_1, x_2 Euclidean space will be the lines $x_1 = 0$ and $x_2 = 0$. This is because $x_1 x_2 = 0$ is true when at least one of x_1 or x_2 is 0. Which is satisfied by the lines running along the x1-axis and the x2-axis.
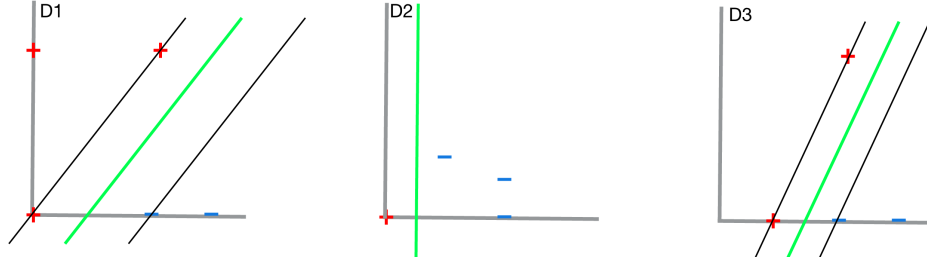
Maximum margin of 1.

Figure 2: Geometry of the three training sets.

(2a) D_1 **-** Points x_1 and x_3 form the line $y - x = 0$ which has a slope of 1. A line with slope 1 that passes through point x_5 is given by $y - x + 1 = 0$. The parallel separating line between these two lines will give the maximum margin. It intersects the x-axis at $(0.5, 0)$, has a slope of 1 and is given by the formula $y - x + 0.5 = 0$. We can use the distance formula for a line and a point to determine the margin between this separating line and the nearest points.

$$distance(ax + by + c = 0, (x_0, y_0)) = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}} \tag{1}$$

The separating line is defined by $a = -1$, $b = 1$, and $c = 0.5$.

$$distance(-x + y + 0.5 = 0, (0,0)) = \frac{0.5}{\sqrt{2}}$$

$$distance(-x + y + 0.5 = 0, (1,1)) = \frac{0.5}{\sqrt{2}}$$

$$distance(-x + y + 0.5 = 0, (1,0)) = \frac{0.5}{\sqrt{2}}$$

$$maximum\ \gamma = \frac{1}{2\sqrt{2}}$$

D_2 **-** The separating line that produces the maximum margin is evident from the geometry of the training points. As seen in figure 2, the line $x = 0.25$ separates the points with the maximum margin.

The separating line is defined by $a = 1$, $b = 0$, and $c = -0.25$.

$$distance(x - 0.5 = 0, (0,0)) = 0.25$$

$$distance(x - 0.5 = 0, (\frac{1}{2}, \frac{\sqrt{3}}{2})) = 0.25$$

$$maximum\ \gamma = 0.25$$

D_3 **-** Points x_3 and x_4 form the line $y - 2x + 1 = 0$ which has a slope of 2. A line with slope 2 that passes through point x_5 is given by $y - 2x + 2 = 0$. The parallel separating line between these two lines will give the maximum margin. It intersects the x-axis at $(\frac{3}{4}, 0)$, has a slope of 2 and is given by the formula $y - 2x + 1.5 = 0$.

2

The separating line is defined by $a = -2$, $b = 1$, and $c = 1.5$.

$$distance(-2x + y + 1.5 = 0, (\frac{1}{2}, 0)) = \frac{0.5}{\sqrt{5}}$$

$$distance(-2x + y + 1.5 = 0, (1, 1)) = \frac{0.5}{\sqrt{5}}$$

$$distance(-2x + y + 1.5 = 0, (1, 0)) = \frac{0.5}{\sqrt{5}}$$

$$\boxed{maximum\ \gamma = \frac{1}{2\sqrt{5}}}$$

(2b)

$$\boxed{Perceptron\ Mistake\ Bound\ = (\frac{R}{\gamma})^2} \tag{2}$$

D_1 - From part a we found $\gamma = \frac{1}{2\sqrt{2}}$. To find R, we find the training point furthest away from the origin as the line from the origin to this point defines the radius of the training examples. Using the Pythagorean Theorem, we determine x_7 is the furthest point from the origin with a distance of 1.5.

$$PMB(\gamma, R) = PMB(\frac{1}{2\sqrt{2}}, 1.5) = (\frac{R}{\gamma})^2 = 18$$

$\boxed{\text{Mistake bound of 18}}$

D_2 - We found $\gamma = 0.25$. The furthest point from the origin is point x_8 with a distance of $\frac{\sqrt{5}}{2}$.

$$PMB(\gamma, R) = PMB(0.25, \frac{\sqrt{5}}{2}) = (\frac{R}{\gamma})^2 = 20$$

$\boxed{\text{Mistake bound of 20}}$

D_3 - We found $\gamma = \frac{1}{2\sqrt{5}}$. The furthest point from the origin is the point x_7 with a distance of 1.5.

$$PMB(\gamma, R) = PMB(\frac{1}{2\sqrt{5}}, 1.5) = (\frac{R}{\gamma})^2 = 45$$

$\boxed{\text{Mistake bound of 45}}$

(2c) I rank the ease of learning of the datasets: D_1, D_2, D_3 based off their mistake bound. The smaller the mistake bound, the smaller the number of mistakes that will be made on the training set. Thus making it easier to learn the true hypothesis h that splits the training set.

2: Kernels

(1a) Given two kernels K_1 and K_2 we will prove the product of these two kernels is itself a kernel.

$$K(\mathbf{x}, \mathbf{z}) = K_1(\mathbf{x}, \mathbf{z})K_2(\mathbf{x}, \mathbf{z})$$

A kernel is defined as a dot product in high dimensional space.

$$K_1(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T\phi(\mathbf{z}) \; and \; K_2(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T\phi(\mathbf{z})$$

Bringing this all together with the original K we are proving to be a kernel.

$$K(\mathbf{x}, \mathbf{z}) = K_1(\mathbf{x}, \mathbf{z})K_2(\mathbf{x}, \mathbf{z})$$

$$K(\mathbf{x}, \mathbf{z}) = (\phi(\mathbf{x})^T\phi(\mathbf{z}))(\phi(\mathbf{x})^T\phi(\mathbf{z}))$$

$$K(\mathbf{x}, \mathbf{z}) = (\sum_i f_i(x)f_i(z))(\sum_j g_j(x)g_j(z))$$

Rearranging terms to form a single dot product in a high dimensional space.

$$K(\mathbf{x}, \mathbf{z}) = \sum_{i,j} f_i(x)g_j(x)f_i(z)g_j(z)$$

$$K(\mathbf{x}, \mathbf{z}) = \sum_{i,j} h_{ij}(x)h_{ij}(z)$$

$$K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T\phi(\mathbf{z})$$

Thus proving that K, the product of two kernels K_1 and K_2 is itself a kernel.

(1b) To start, we will prove that $K(\mathbf{x}, \mathbf{z}) = \alpha K_1(\mathbf{x}, \mathbf{z}) + \beta K_2(\mathbf{x}, \mathbf{y})$.

We know that a kernel is a dot product in a high dimensional space. As such, if we want to scale the dot product by a scalar, we scale each blown-up vector by the square root of that scalar. That is, if we want to scale a product then we simply scale the factors by the square root of that scalar.

$$K(\mathbf{x}, \mathbf{z}) = (\sqrt{\alpha}\phi(x)\sqrt{\alpha}\phi(z)) + (\sqrt{\beta}\phi(x)\sqrt{\beta}\phi(z))$$

Rearrange terms to form a single dot product.

$$K(\mathbf{x}, \mathbf{z}) = (\sqrt{\alpha}\phi(x)\sqrt{\beta}\phi(x))(\sqrt{\beta}\phi(z)\sqrt{\alpha}\phi(z))$$

Thus we have proven that the sum of two scaled kernels is also a kernel.

Returning to the original objective - proving $K(\mathbf{x}, \mathbf{z}) = P(K_1(\mathbf{x}, \mathbf{z}))$. We know that a polynomial of kernels will be composed of: the sum of scaled kernels and the product of kernels. In part **(1a)**, we proved that the product of two kernels is a kernel. And we just proved that the sum of two scaled kernels is also a kernel.

Therefore we have proven that $K(\mathbf{x}, \mathbf{z}) = P(K_1(\mathbf{x}, \mathbf{z}))$.

(2) Incomplete - ran out of time.

(3) Incomplete - ran out of time.

3: Support Vector Machines

Note: In my submitted source code there is a 'traces' directory. This contains traces of all of the experiments I ran to collect the following data.

(1) A SVM using Stochastic sub-gradient descent was trained on the `handwriting` dataset. The hyperparameters $C = 1$, $\gamma_0 = 0.01$, and $epoch = 10$ were used.

The classifier was evaluated using the *training* and *test* `handwriting` datasets with the following results:

Data Set	Mistakes	Example Count	Accuracy
handwriting training	1	1,000	0.999
handwriting testing	8	593	0.987

Table 2: Classification accuracy on the training and test set.

(2) Splits for continuous features: The `madelon` dataset has continuous features. I decided to make the features binary features. To determine their value, I took the mean value a feature had across all training examples. Then, depending on which side of the mean each feature value landed, the feature value was replaced by 0 for less than and 1 for greater than or equal to the mean.

Cross-validation: To determine the best hyperparameters C and γ_0 I ran 5-fold cross validation on various combinations of C and γ_0. For all the cross validations trials an epoch of 20 was used.

C	Initial Learning Rate	Accuracy
2	0.5	0.527
2	0.01	0.539
2	0.001	0.532
0.5	0.5	0.536
0.5	0.01	0.533
0.5	0.001	0.522
0.25	0.5	0.538
0.25	0.01	0.53
0.25	0.001	0.519
0.125	0.5	0.537
0.125	0.01	0.542
0.125	0.001	0.527
0.0625	0.5	0.526
0.0625	0.01	0.514
0.0625	0.001	0.545
0.03125	0.5	0.516
0.03125	0.01	0.521
0.03125	0.001	0.548

Table 3: Results of 5-fold cross-validation.

The results of the cross-validation produced a best C of 0.03125 and a γ_0 of 0.001 using an epoch of 10. These hyperparameters were used to train the SVM classifier and it was evaluated on both the *training* and *test* set.

Data Set	Mistakes	Example Count	Accuracy
madelon training	641	2,000	0.679
madelon testing	248	600	0.587

Table 4: Classification accuracy on the training and test set.

(3) In addition to the accuracy of of the two classifiers from part a and part b, I calculated the precision, recall and F_1 score on each dataset.

Data Set	Precision	Recall	F_1 Score
handwriting training	1.0	0.998	0.999
handwriting testing	1.0	0.974	0.987
madelon training	0.679	0.68	0.679
madelon testing	0.586	0.59	0.588

Table 5: Precision, recall and F_1 score of previous classifiers.

6

4: Ensemble of Decision Trees

Note: In my submitted source code there is a 'traces' directory. This contains traces of all of the experiments I ran to collect the following data.

(1) Hyperparameters: I ran cross validation on the `handwriting` training set to determine good hyperparameters for this experiment. The hyperparameter to determine for the ensemble of decision trees is m or the number of random samples used when training each decision tree. N and k are fixed for this experiment.

For the meta-SVM classifier, I determined the best γ_0, C and epoch to be used on features produced by the ensemble of decision trees.

m	N	k	γ_0	C	epoch
2,000	5	8	0.1	4	15

Table 6: Hyperparameters for the ensemble and meta-SVM classifiers

Using these chosen hyperparameters, 5 decision trees were trained on k random features each. The results of these trees were used to train as the training examples for a meta-SVM classifier. This trained meta-SVM classifier was evaluated on both the *training* and *test* `handwriting` datasets.

Data Set	Mistakes	Examples Count	Accuracy	Precision	Recall	F_1 Score
handwriting training	257	1,000	0.743	0.844	0.651	0.735
handwriting test	127	593	0.786	0.884	0.673	0.764

Table 7: Results of meta-SVM classifier on the `handwriting` dataset.

(2a) Splits for continuous features: The `madelon` dataset has continuous features. I decided to make the features binary features. To determine their value, I took the mean value a feature had across all training examples. Then, depending on which side of the mean each feature value landed, the feature value was replaced by 0 for less than and 1 for greater than or equal to the mean.

Hyperparameters: I ran cross validation on the `madelon` training set to determine good hyperparameters.

m	k	γ_0	C	epoch
500	11	0.1	4	5

Table 8: Hyperparameters for the ensemble and meta-SVM classifiers

Using these hyperparameters, I varied N through the values 10, 30, and 100 to train the classifier. For each choice I evaluated the classifiers performance using the *training* and *test* `madelon` dataset.

Data Set	N	Mistakes	Examples Count	Accuracy	Precision	Recall	F_1 Score
madelon training	10	999	2,000	0.5	0.5	0.996	0.666
madelon test	10	301	600	0.498	0499	0.993	0.664
madelon training	30	998	2,000	0.501	0.501	0.611	0.551
madelon test	30	293	600	0.512	0.51	0.617	0.558
madelon training	100	980	2,000	0.51	0.507	0.678	0.58
madelon test	100	280	600	0.533	0.525	0.7	0.6

Table 9: Results of meta-SVM classifier on the `madelon` dataset.

I determined the best choice for N is 100 out of the given choices. The classifier using $N = 10$ achieved a better F_1 score but it predicted true on every label. When N was small on this dataset, I found the classifier always predicted true. I found pretty poor performance on the `madelon` dataset using an ensemble of decision trees and a meta-SVM classifier. I am experiencing baseline performance.

(2b) Additional values of N: I found that $N = 100$ had the best performance in part a. I tried the additional values 5, 15, and 60 to see how they perform. Other than N, I used the same hyperparameters as used in part a.

m	k	γ_0	C	epoch
500	11	0.1	4	5

Table 10: Hyperparameters for the ensemble and meta-SVM classifiers

As in part a, I trained the classifier and evaluated it's performance on the *training* and *test* `madelon` datasets.

Data Set	N	Mistakes	Examples Count	Accuracy	Precision	Recall	F_1 Score
madelon training	5	1,031	2,000	0.485	0.487	0.57	0.525
madelon test	5	304	600	0.493	0494	0.597	0.541
madelon training	15	1,021	2,000	0.49	0.473	0.187	0.268
madelon test	15	297	600	0.505	0.513	0.2	0.288
madelon training	60	994	2,000	0.503	0.512	0.123	0.198
madelon test	60	313	600	0.478	0.409	0.093	0.151

Table 11: Results of meta-SVM classifier on the `madelon` dataset.

Experimenting with 6 different values of N, I determined N = 100 is the best choice for the `madelon` dataset.