UNIVERSITY OF MICHIGAN
Department of Electrical Engineering and Computer Science
EECS 445 Introduction to Machine Learning
Winter 2019

Homework 2, Due: Tue. 02/26 at 11:59pm

Submission: Please upload your completed assignment by 11:59pm on Feb. 26, 2019 to Gradescope.

1 Support Vectors [8 pts]

1.1 SVM Primal [4 pts]

Suppose we are looking for a maximum hard margin (no slack variables) linear classifier *through the origin*, i.e., $b = 0$ with binary labels in $\{-1, 1\}$. In other words, we minimize $\frac{||\bar{\theta}||^2}{2}$ subject to $y^{(i)}\bar{\theta} \cdot \bar{x}^{(i)} \geq 1,\ i = 1, ..., n$

a) (2pts) Given a single training vector $\bar{x} = [a_1, a_2]^T$ with label $y = -1$, **what** is the $\bar{\theta}^*$ that satisfies the above constrained minimization?

Hint: *Try thinking about this problem geometrically.*

b) (1pt) Suppose we have two training examples, $\bar{x}^{(1)} = [-2, -1]^T$ and $\bar{x}^{(2)} = [-1, -1]^T$ with labels $y^{(1)} = 1$ and $y^{(2)} = -1$. **What** is $\bar{\theta}^*$ in this case, and **what** is the margin γ to the support vector?

c) (1pt) **How** would the classifier and the margin in the previous question change if the offset parameter b were allowed to be non-zero? What are $(\bar{\theta}^*, b^*)$ and γ in this case?

1.2 SVM Dual [4 pts]

Consider solving the dual SVM optimization problem in the simple case of only two 2-dimensional training examples $\bar{x}^{(1)}, \bar{x}^{(2)} \in \mathbb{R}^2$ with labels $y^{(1)} = 1$ and $y^{(2)} = -1$. For this problem, we assume in addition that $\left\|\bar{x}^{(1)}\right\| = \left\|\bar{x}^{(2)}\right\| = 1$ and that the kernel function is $K(\bar{x}, \bar{x}') = \bar{x} \cdot \bar{x}'$ (linear kernel). Note that we are including the offset parameter in this case. The goal is to maximize

$$\sum_{i=1}^{2} \alpha_i - \frac{1}{2} \sum_{i=1}^{2} \sum_{j=1}^{2} \alpha_i \alpha_j y^{(i)} y^{(j)} K(\bar{x}^{(i)}, \bar{x}^{(j)})$$

subject to $\alpha_i \geq 0$ and $\sum_{i=1}^{2} \alpha_i y^{(i)} = 0$. Express your answers in terms of $\bar{x}^{(1)}$ and $\bar{x}^{(2)}$.

a) (1pt) **What** are the resulting optimal values for the Lagrange multipliers α_1^* and α_2^*? (Hint: use the constraints first, write the optimization problem in terms of α_1^* alone). Make sure that you use the condition on the norm of the training examples.

b) (1pt) Using (a), write the optimal parameter values of $\bar{\theta}^*$ in terms of the training examples only ($\bar{x}^{(1)}$ and $\bar{x}^{(2)}$)

c) (1pt) **What** is the offset parameter b^*?

d) (1pt) **What** is the distance (margin γ) from the decision boundary to the support vector?

2 Soft Margin SVM Dual [2 pts]

Consider the dual formulation of a soft-margin SVM with regularization, where C is a regularization hyper-parameter.

$$\begin{aligned} \underset{\bar{\alpha}}{\text{maximize}} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y^{(i)} y^{(j)} \bar{x}^{(i)} \cdot \bar{x}^{(j)} \\ \text{subject to} \quad & \sum_{i=1}^{N} \alpha_i y^{(i)} = 0, \\ & 0 \leq \alpha_i \leq C \, , \forall i = 1, ..., N. \end{aligned}$$

i	$\bar{x}^{(i)}$	$y^{(i)}$	α_i
1	[-2.78,-2.08]	-1	0
2	[-5.20,1.99]	-1	0
3	[-2.69,4.26]	-1	0.05
4	[-2.25,-3.96]	-1	0
5	[-2.59,-3.36]	-1	0
6	[-1.17,1.44]	-1	0.05
7	[-1.94,-3.68]	-1	0.0159
8	[-1.19,-2.99]	-1	0.05
9	[-2.14,-0.22]	-1	0.05
10	[-3.12,-1.99]	-1	0

i	$x^{(i)}$	$y^{(i)}$	α_i
11	[0.68, 1.76]	1	0.05
12	[2.38, 0.52]	1	0
13	[-1.49, 1.65]	1	0.05
14	[1.74, 0.22]	1	0.05
15	[-0.13, -1.77]	1	0.05
16	[1.59, 5.35]	1	0
17	[1.65, 2.17]	1	0.0159
18	[2.56, -4.04]	1	0
19	[2.75, 0.73]	1	0
20	[2.34, 2.92]	1	0

a) (1pt) Given the table above, which points are the support vectors?

b) (1pt) Assuming the regularization hyperparameter $C = 0.05$, given the α_i for each of the points above, what are the equations to compute the optimal parameter values $\bar{\theta}^*$ and b^*? What are the resulting $\bar{\theta}^*$ and b^* for the data above?

3 Kernels [8 pts]

3.1 From feature mapping to kernel [2 pts]

We have two kernels, $K_1(\bar{x}, \bar{z}) = \bar{x} \cdot \bar{z}$ and $K_2(\bar{x}, \bar{z}) = ?$, where $\bar{x}, \bar{z} \in \mathbb{R}^2$. The first kernel is defined for us, but the second kernel is unknown. We can define a third kernel, which is the product of the first two: $K_3(\bar{x}, \bar{z}) = K_1(\bar{x}, \bar{z}) K_2(\bar{x}, \bar{z})$. The feature mapping corresponding to $K_3(\bar{x}, \bar{z})$ is given:

$$\phi(\bar{x}) = [x_1^3, \ \sqrt{2}x_1^2 x_2, \ x_1 x_2^2, \ x_1^2 x_2, \ \sqrt{2}x_1 x_2^2, \ x_2^3, \ \sqrt{6}x_1^2, \ 2\sqrt{3}x_1 x_2, \ \sqrt{6}x_2^2, \ 3x_1, \ 3x_2 \ ]$$

In other words, $K_3(\bar{x}, \bar{z}) = \phi(\bar{x})\phi(\bar{z})$.

Hint: *Group the first six terms, the next three, and the last two terms together, and simplify from there.*

(a) What is $K_2(\bar{x}, \bar{z})$?

3.2 Kernelizing logistic regression [6 pts]

Here is the algorithm of SGD for logistic regression.

```
1 Initialization: θ̄^(0) = 0̄
2 Repeat until convergence:
3     for i = 1, ..., N:
4         θ̄^(k+1) ← θ̄^(k) + η σ(−y^(i) θ̄^(k) · x̄^(i)) y^(i) x̄^(i)
5         k ← k + 1
6 θ̄ = θ̄^(k)
7 Calculate probability that x̄ is classified with label y = +1:
8 h(x̄) = σ(θ̄ · x̄) = 1/(1+exp(−θ̄·x̄))
```

We would like to kernelize this algorithm. This can be done by mapping each example $\bar{x} \in \mathbb{R}^d$ into a new feature vector $\phi(\bar{x}) \in \mathbb{R}^{d'}$(in many cases, $d' > d$) and reformulating the algorithm such that it only uses the kernel function $k(\bar{x}, \bar{x}')$ corresponding to $\phi(\bar{x}) \cdot \phi(\bar{x}')$. This will require a number of changes. In the algorithm above, we need to calculate $\bar{\theta} \cdot \bar{x}$. However, once we introduce a feature mapping $\phi : \mathbb{R}^d \to \mathbb{R}^{d'}$, $\bar{\theta}$ will lie in a d'-dimensional space, where d' may be very large. We want a new formulation that does not explicitly depend on $\bar{\theta}$.

(a) (2pts) We will derive a new version of the above algorithm, by first reformulating $\bar{\theta}$ as a combination of training examples $\bar{x}^{(i)}$ for $i = 1, ..., N$, *i.e.* $\bar{\theta} = \sum_{i=1}^{N} \alpha_i \bar{x}^{(i)}$.

Now consider a dataset with 3 data points, *i.e.* $N = 3$. Suppose we update the parameters for 9 times before convergence. These assumptions are only for part (a).
We DO NOT shuffle the points after each epoch. **Write down** the expression for α_i for $i = 1, 2, 3$ as a function of η, $\bar{x}^{(i)}$, $y^{(i)}$, and $\bar{\theta}^{(k)}$ for $k = 0, \cdots, 8$.

(b) These new parameters, $\bar{\alpha}$, will help us replace $\bar{\theta}$ in the algorithm. We can turn the to-be-trained parameters from $\bar{\theta} \in \mathbb{R}^d$ to $\bar{\alpha} \in \mathbb{R}^N$. Next, we will come up with the appropriate update for $\bar{\alpha}$ as outlined in the algorithm below.

```
1 Initialization: ᾱ^(0) = 0̄
2 Repeat until convergence:
3     for i = 1, ..., N:
4         <Update all N elements in ᾱ^(k+1) in terms of η, ᾱ^(k), y^(i), x̄^(i)>
5         k ← k + 1
6 ᾱ = ᾱ^(k)
7 Calculate probability that x̄ is classified with y = +1:
8 <Update compute h(x̄) in terms of ᾱ, x̄^(j) for j = 1, ··· , N>
```

Consider the following questions:

(i) (1pt) First, consider the parameter update **(line 4)**. We seek to update the parameters of our model. Based on the relationship between $\bar{\alpha}$ and $\bar{\theta}$, think about how many elements in $\bar{\alpha}$ need updating. **How** should we update the parameters? You may write more than one line of pseudocode.

(ii) (1pt) When we finish updating the parameters, we need to calculate the probability that a new example $\bar{x}$ is classified as $y = +1$ **(line 8)**. **How** can we express this probability in terms of $\bar{\alpha}$?

(c) (2pts) Now we can kernelize the algorithm from part (b): map each example $\bar{x}$ into a feature vector $\phi(\bar{x})$; reformulate the algorithm such that it only uses the kernel function $k(\bar{x}, \bar{x}')$ corresponding to $\phi(\bar{x}) \cdot \phi(\bar{x}')$. **Complete** the algorithm below to kernelize the logistic regression:

```
1 Initialization: ᾱ^(0) = 0̄
2 Repeat until convergence:
3     for i = 1,...,N:
4         # You can write more than one line here.
5         _____(1pt)
6         k ← k + 1
7 ᾱ = ᾱ^(k)
8 Calculate probability that x̄ is classified with y = +1:
9 h(x̄)=_____(1pt)
```

4 Entropy [2 pts]

Feature			Classification
Punctuality	Boarding Efficiency	Quality of Service	Satisfied with the flight
Delayed	Decent	Good	Yes
On Time	Decent	Poor	No
Delayed	Slow	Good	No
Delayed	Fast	Good	Yes
On Time	Slow	Great	No
Delayed	Fast	Poor	Yes
On Time	Decent	Good	Yes
On Time	Decent	Great	Yes
On Time	Slow	Poor	No

a) (1pt) Consider the table above, which maps variables pertaining to a customer's experience to whether or not they were satisfied with their flight. We consider a categorical representation (as opposed to an ordinal representation). **Which feature(s)** result in the highest conditional entropy $H(Y|X)$, where Y is the outcome and X is one of the features (Punctuality, Boarding Efficiency, and Quality of Service)?

b) (1pt) **Calculate** the information gain $IG(X, Y)$ for each of the features and the outcome. **Which feature(s)** result in the highest information gain?

5 Ensemble Methods [5 pts]

We will study the performance of two ensemble methods on the very popular MNIST dataset consisting of handwritten digits. This dataset contains 70000 samples (each a 28×28 grayscale image having 784 features) of handwritten digits, classified into 10 classes (0 through 9). Here, we will consider a subset of the data pertaining to four classes, which you can fetch using the provided `load_mnist(classes)` function. Please be aware that due to the relatively large dataset size, the code will take a longer time to run.

Within `HW2_ensemble.py`, the following functions have been implemented for you:

- `load_mnist(classes)`
- `get_avg_performance(X, y, m_vals, nsplits=50)`
- `plot_data(bagging_scores, random_forest_scores, m_range)`

It also contains the following function declarations that you will implement:

- `bagging_ensemble(X_train, y_train, X_test, y_test, n_clf=10)`
- `random_forest(X_train, y_train, X_test, y_test, m, n_clf=10)`

a) (2pts) **Implement** `random_forest(X_train, y_train, X_test, y_test, m, n_clf=10)` based on the specification provided in the skeleton code. Random forest consists of `n_clf`-many decision trees where each decision tree is trained independently on a bootstrap sample of the training data (for this problem, `n_clf=10`). For each node, we randomly select m features as candidates for splitting on.

Here, the final prediction of the bagging classifier is determined by a majority vote of these `n_clf` decision trees. In the case of ties, randomly sample among the plurality classes (i.e. the classes that are tied) to choose a label.

You should use the `sklearn.tree.DecisionTreeClassifier` class. Set `criterion='entropy'` to avoid the default setting. Also, see the `max_features` parameter within this class.

Note: Do not set the `max_depth` parameter. Remember that you are free to use helper functions to keep your code organized.

b) (1pt) **Implement** `bagging_ensemble(X_train, y_train, X_test, y_test, n_clf=10)` based on the specification provided in the skeleton code. Like random forest, a bagging ensemble classifier consists of `n_clf`-many decision trees where each decision tree is trained independently on a bootstrap sample of the training data. However, all features are considered at every split.

Again, the final prediction is determined by a majority vote of these `n_clf` decision trees.

c) (2pts) Now, we will compare the performance of these ensemble classifiers using `get_avg_performance()`. Measure the median performance across 50 random splits of the digits dataset into training (80%) and test (20%) sets, and **plot** the returned performance for each algorithm over the range of m values specified in the skeleton code (use the `plot_data()` function). See Figure 1 for an example of the plot (yours may differ due to randomness, but the general trend should be the same). How does the average test performance of the two methods compare as we vary m (the size of the randomly selected subset of features)?

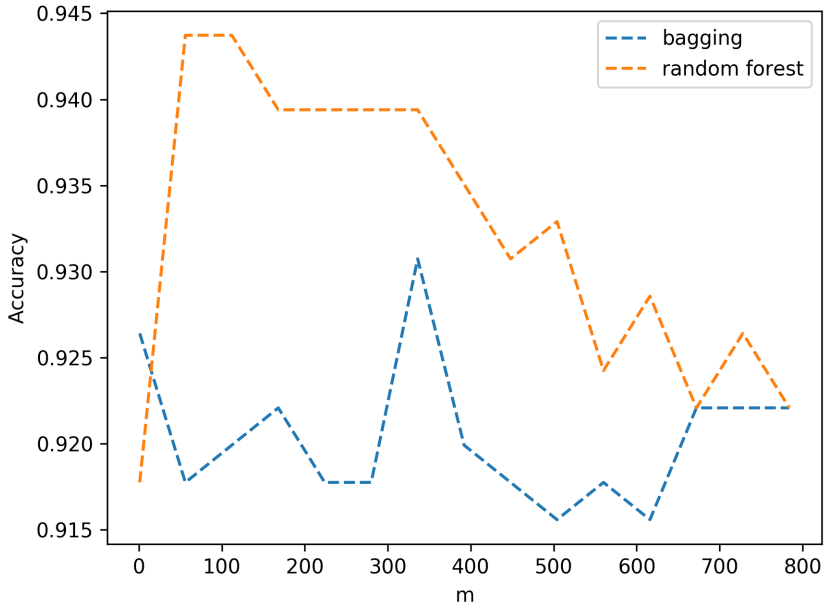

Figure 1: Plot of accuracy of random forest, bagging ensemble methods for $m = 1, 56, 112, ..., 784$

6 Boosting [5 pts]

Boosting combines a set of weak classifiers into a stronger ensemble classifier $h_M(\bar{x}) = \sum_{j=1}^{M} \alpha_j h(\bar{x}; \bar{\theta}_j)$, where $\alpha_j \geq 0$ are votes allocated to each base classifier ("decision stump") $h(\bar{x}; \bar{\theta}_j) = \text{sign}(\theta_1^{(j)} x_k + \theta_0^{(j)})$ described by a parameter vector $\bar{\theta}_j = \{k, \theta_1^{(j)}, \theta_0^{(j)}\}$ that encodes the co-ordinate, direction and location information. Finding a jointly optimum solution of $\bar{\theta}_j$ and α_j for all j is a hard problem and therefore, we take an iterative approach exemplified by the adaptive boosting (AdaBoost) algorithm:

Set $W_0(i) = \frac{1}{n}$ for $i = 1 \ldots n$
For $m = 1$ to M do:
    Find $h(\bar{x}; \bar{\theta}_m)$ that minimizes the weighted training error ϵ_m:
        $\epsilon_m = \sum_{i=1}^{n} W_{m-1}(i)[[y^{(i)} \neq h(\bar{x}^{(i)}; \bar{\theta}_m)]]$

    Given $\bar{\theta}_m$, compute α_m that minimizes weighted training loss:
        $\alpha_m = \frac{1}{2} \log(\frac{1-\epsilon_m}{\epsilon_m})$

    Update weights on all training examples
    For $i = 1$ to n do:
        $W_m(i) = c_m W_{m-1}(i) \exp\{-y^{(i)} \alpha_m h(\bar{x}^{(i)}; \bar{\theta}_m)\}$

Consider the labeled training points in Figure 2, where the •'s and ×'s denote negative and positive labels, respectively. We wish to apply Adaboost with decision stumps to solve the classification problem. In each boosting iteration, we select the stump that minimizes the weighted training error, breaking ties arbitrarily.

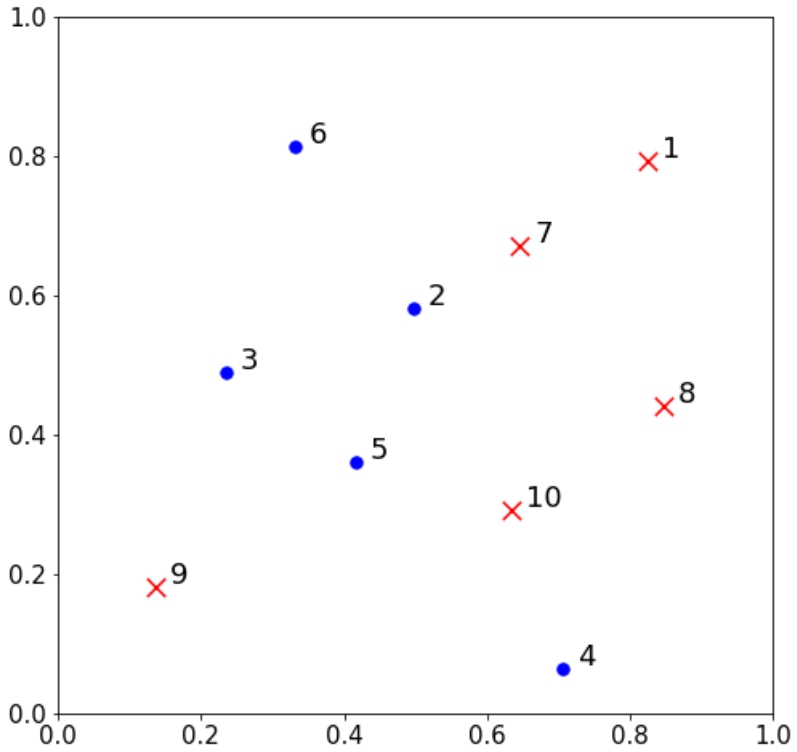

Figure 2: •-negative points and ×-positive points.

a) (1pt) In Figure 2, **draw** the decision boundary corresponding to the first decision stump that the boosting algorithm would choose. **Label** this boundary as `(1)`, and also **indicate** the `+/−` sides of the decision boundary. (Note: we have provided the data in the skeleton code associated with this problem, so that you can recreate the plot yourself if needed.)

b) (1pt) In the same Figure 2, **circle** the point(s) that have the largest weight after the first boosting iteration.

c) (1pt) **What is the weighted error** of the first decision stump after the first boosting iteration, i.e., after the points have been re-weighted?

d) (1pt) **Draw** the decision boundary corresponding to the second decision stump, again in Figure 2, and **label** it as `(2)` also **indicating** the `+/−` sides of the boundary.

e) (1pt) Will any of the points be misclassified by the combined classifier after the two boosting iterations? **Provide a brief justification**, no calculations are necessary (the point will be awarded for the justification, not whether your y/n answer is correct).

REMEMBER Submit your completed assignment by 11:59pm on Feb. 26, 2018 to Gradescope.