0 Instructions

Homework is due Tuesday, February 20, 2024 at 23:59pm Central Time. Please refer to `https://courses.grainger.illinois.edu/cs446/sp2024/homework/hw/index.html` for course policy on homeworks and submission instructions.

1 Soft-margin SVM: 4pts

In the lecture, we learned about the duality of hard-margin SVM. Given data examples $((\boldsymbol{x}_i, y_i))_{i=1}^N$, the primal form of hard-margin SVM is

$$\arg\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|_2^2, \quad \text{s.t. } y_i(\boldsymbol{w}^T\boldsymbol{x}_i) \geq 1 \quad \text{for all } i.$$

Its dual form is

$$\max_{\boldsymbol{\alpha}\geq 0} \sum_{i=1}^N \alpha_i - \frac{1}{2}\sum_{i=1}^N\sum_{j=1}^N \alpha_i\alpha_j y_i y_j (\boldsymbol{x}_i^T\boldsymbol{x}_j), \quad \text{s.t. } \alpha_i \geq 0 \quad \text{for all } i, \ \sum_{i=1}^N \alpha_i y_i = 0.$$

Now consider the soft-margin SVM. The primal form is

$$\arg\min_{\boldsymbol{w},\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^N \xi_i, \quad \text{s.t. } y_i(\boldsymbol{w}^T\boldsymbol{x}_i) \geq 1 - \xi_i \text{ and } \xi_i \geq 0 \quad \text{for all } i$$

Derive the dual form of the soft-margin SVM.

2 SVM, RBF Kernel and Nearest Neighbor: 6pts

1. (1pts) Suppose that given data examples $((\boldsymbol{x}_i, y_i))_{i=1}^N$, an optimal dual solution to the hard-margin SVM is $(\hat{\alpha}_i)_{i=1}^N$. Write the prediction on a new $\boldsymbol{x}$ as a function of $(\hat{\alpha}_i)_{i=1}^N$, $((\boldsymbol{x}_i, y_i))_{i=1}^N$ and $\boldsymbol{x}$.

   Hint: The prediction on $\boldsymbol{x}$ is $f(\boldsymbol{x}) = \hat{\boldsymbol{w}}^T \boldsymbol{x}$.

2. (1pts) Now we apply the kernel trick to the prediction function $f(\boldsymbol{x})$. Recall that the RBF kernel (Gaussian kernel) is

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}{2\sigma^2}\right).$$

   What is $f_\sigma(\boldsymbol{x})$, the prediction on $\boldsymbol{x}$ using RBF kernel?

3. (4pts) Denote $S \subset \{1, 2, \ldots, n\}$ as the set of indices of support vectors. Given an input $\boldsymbol{x}$, let $T := \text{argmin}_{i \in S} \|\boldsymbol{x} - \boldsymbol{x}_i\|_2$ denote the set of closest support vectors to $\boldsymbol{x}$, and let $\rho := \min_{i \in S} \|\boldsymbol{x} - \boldsymbol{x}_i\|_2$ denote this smallest distance. (In other words, $T := \{i \in S : \|\boldsymbol{x} - \boldsymbol{x}_i\|_2 = \rho\}$.) Prove that

$$\lim_{\sigma \to 0} \frac{f_\sigma(\boldsymbol{x})}{\exp\left(-\rho^2/2\sigma^2\right)} = \sum_{i \in T} \hat{\alpha}_i y_i.$$

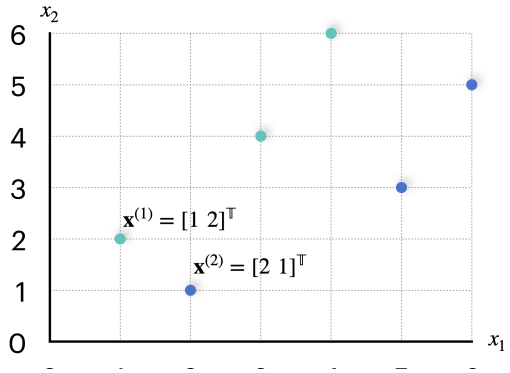
   Hint: Split up the sum over elements of S into two sums: one over $i \in T$ and one over $i \in S \setminus T$.

   Remark: In other words, when the bandwidth σ becomes small enough, the RBF kernel SVM is almost the 1-nearest neighbor predictor with the set of support vectors as the training set. Note that while nearest neighbors will not be introduced until Lecture 11, solving this problem does not require knowledge of them.

   Remark 2: The prediction function, $f_\sigma(\boldsymbol{x}) : \boldsymbol{x} \to \phi(\boldsymbol{x})^\top \overline{\mathbf{w}}$, depends only on the labels of the closest support vectors of $\boldsymbol{x}$, i.e., the constituents of the set T.

3 Decision Tree and Adaboost: 12 pts

The figure below shows a dataset $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{6}$ (where $\boldsymbol{x}^{(i)} \in \mathbb{R}^2$, $y^{(i)} \in \mathbb{R}$), containing six data points with two features x_1 and x_2. For example, $\boldsymbol{x}^{(1)} = [1\ 2]^\top$ and $\boldsymbol{x}^{(2)} = [2\ 1]^\top$. The label $y^{(i)}$ can take on the values 1 (blue) or -1 (green).



The decision tree defined in class can only support discrete-valued instances. Here, we extend this concept to general continuous spaces. A continuous-valued decision attribute need to be represented using a comparison operator ($\geq, <$). Specifically, for each round, unlike discrete-valued tree building that chooses only which feature to use as the current decision attribute, we will specify a feature (x_1 or x_2) and also **a threshold** τ, and then create **two descendant nodes** at the current node. For all data points in the current node, those below the threshold will be put into child node "$x_j < \tau$", and those above the threshold will be put into child node "$x_j \geq \tau$" ($j = 1$ or 2).

Note: We assume τ can only be integer values. Please describe the split rule as "$x_j \geq \tau$", such as $x_1 \geq 1$ or $x_2 \geq 2$. Do not use the answers like $x_1 \geq 2.5$ or $x_2 > 2$. And also make sure to describe what the predicted label is in two child nodes "$x_j < \tau$" and "$x_j \geq \tau$" ($j = 1$ or 2).

Note: Use $\log_2(\cdot)$ in the calculation of entropy.

1. (1pts) What is the sample entropy of $\mathcal{D}$? Show each step of your calculation.
2. (2pts) What is the maximum information gain if we split the root into two child nodes? what is the rule for this split? Show each step of calculating information gain. You do not need to prove the split.
3. (3pts) After the first split in 2., how do we further split child nodes based on maximum information gain? Please also give information gain for each split.

Adaboost. A decision stump is an one-level decision tree. It classifies cases with only one attribute. In this problem, you will run through $T = 2$ steps of AdaBoost with decision

stumps as weak learners on dataset D. For the sake of notation simplicity, we denote $\boldsymbol{x}^{(1)} = [1, 2]^\top, \boldsymbol{x}^{(2)} = [2, 1]^\top, \boldsymbol{x}^{(3)} = [3, 4]^\top, \boldsymbol{x}^{(4)} = [4, 6]^\top, \boldsymbol{x}^{[5]} = [5, 3]^\top, \boldsymbol{x}^{(6)} = [6, 5]^\top$.

4. (4pts) For each iteration $t = 1, 2$, compute the weights for each sample $\gamma_t \in \mathbb{R}^6$, the weighted error rate ϵ_t, the weight of the decisition stump α_t and the decision stump f_t.

   Note:

   - Please describe the split rule as $x_j \geq \tau$ or $x_j < \tau$ where τ is an integer.
   - If you find multiple solutions, giving one is okay.

5. (2pts) Following 4., write down the rule of the classifier you constructed with a formula. Does your solution classify each case correctly? Show your work.

   Note:

   - You may use the function sign in the decision rule. Where

$$\text{sign}(x) = \begin{cases} 1 & \text{for } x \geq 0, \\ -1 & \text{for } x < 0 \end{cases}$$

4 Learning Theory: 14pts

1. (2pts) **Generalization bound.** Imagine tossing a biased coin that lands heads with probability p and let our hypothesis h be the one that always guesses heads. Denote the true error rate as $R(h)$ $(R(h) = p)$ and the empirical error rate as $\hat{R}_S(h) = \hat{p}$, where $\hat{p}$ is the empirical probability of heads based on the training sample drawn (S), which is drawn i.i.d. At least how many samples are needed so that with a probability (i.e. confidence) of 95%, we have an accuracy of $|R(h) - \hat{R}_S(h)| \leq 0.05$?

2. **VC Dimensions.** In the lecture we learned about the VC dimension which captures some kind of complexity or capacity of a set of functions $\mathcal{F}$.

   Note: The straightforward proof strategy to show that the VC dimension of a set of classifiers is k is to first show that there exists a set of k points which is shattered by the set of classifiers. Then, show that any set of $k+1$ points cannot be shattered. You can do that by showing that for any set of $k+1$ points, there exists an assignment of labels which cannot be correctly classified using $\mathcal{F}$.

   Notation: The indicator function is defined as follows

$$\mathbb{1}\{\text{condition}\} = \begin{cases} 1 & \text{if condition is true} \\ -1 & \text{if condition is false} \end{cases}$$

   We will now find the VC dimension of some basic classifiers.

   (a) (4pts) **1D Affine Classifier**

   Let's start with a fairly simple problem. Consider $\mathcal{X} = \mathbb{R}$ and $\mathcal{Y} = \{0, 1\}$. Affine classifiers are of the form:

$$\mathcal{F}_{\text{affine}} = \{\mathbb{1}\{wx + w_0 \geq 0\} : \mathcal{X} \to \mathbb{R} \mid w, w_0 \in \mathbb{R}\},$$

   Show what is $VC(\mathcal{F}_{\text{affine}})$ and prove your result.

   Hint: Try less than a handful of points.

   (b) (4pts) **General Affine Classifier**

   We will now go one step further. Given some dimensionality $k \geq 1$, consider $\mathcal{X} = \mathbb{R}^k$ and $\mathcal{Y} = \{0, 1\}$. Affine classifiers in k dimensions are of the form

$$\mathcal{F}^k_{\text{affine}} = \{\mathbb{1}\{\boldsymbol{w}^\top \boldsymbol{x} + w_0 \geq 0\} : \mathcal{X} \to \mathbb{R} \mid \boldsymbol{w} \in \mathbb{R}^k, w_0 \in \mathbb{R}\}$$

   Show what is $VC(\mathcal{F}^k_{\text{affine}})$ and prove your result.

Hint: Note that $\boldsymbol{w}^\top \boldsymbol{x} + w_0$ can be written as $[\boldsymbol{x}^\top \ \ 1] \begin{bmatrix} \boldsymbol{w} \\ w_0 \end{bmatrix}$. Moreover, consider to put all data points into a matrix, e.g.,

$$\mathbf{X} = \begin{bmatrix} (\boldsymbol{x}^{(1)})^\top & 1 \\ (\boldsymbol{x}^{(2)})^\top & 1 \\ \vdots & \vdots \end{bmatrix}.$$

(c) (4pts) **Cosine classifier**

Consider the classifier based on the cosine function

$$\mathcal{F}_{\cos} = \{\mathbb{1}\{\cos(cx) \geq 0\} : \mathcal{X} \to \mathbb{R} \mid c \in \mathbb{R}\}$$

Show what is $VC(\mathcal{F}_{\cos})$ and prove your result.

5 Coding: SVM, 24pts

Recall that the dual problem of SVM is

$$\max_{\boldsymbol{\alpha}\in\mathcal{C}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j).$$

where the domain $\mathcal{C} = [0,\infty)^n = \{\boldsymbol{\alpha} : \alpha_i \geq 0\}$ for hard-margin SVM, and you already derive the domain for soft-margin SVM in Q1.

Equivalently, it can be formulated as a minimization problem

$$\min_{\boldsymbol{\alpha}\in\mathcal{C}} f(\boldsymbol{\alpha}) := \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{n} \alpha_i.$$

It can be solved by projected gradient descent, which starts from some $\boldsymbol{\alpha}_0 \in \mathcal{C}$ (e.g., $\mathbf{0}$) and updates as follows

$$\boldsymbol{\alpha}_{t+1} = \Pi_{\mathcal{C}}\boldsymbol{\alpha}_t - \eta\nabla f(\boldsymbol{\alpha}_t).$$

Here $\Pi_{\mathcal{C}}[\boldsymbol{\alpha}]$ is the *projection* of $\boldsymbol{\alpha}$ onto $\mathcal{C}$, defined as the closet point to $\boldsymbol{\alpha}$ in $\mathcal{C}$:

$$\Pi_{\mathcal{C}}[\boldsymbol{\alpha}] := \operatorname*{argmin}_{\boldsymbol{\alpha}'\in\mathcal{C}} \|\boldsymbol{\alpha}' - \boldsymbol{\alpha}\|_2.$$

If $\mathcal{C}$ is convex, the projection is uniquely defined.

1. (10pts) Implement an `svm_solver()`, using projected gradient descent formulated as above. See the docstrings in `hw2.py` for details.
2. (10pts) Implement an `svm_predictor()`, using an optimal dual solution, the training set, and an input. See the docstrings in `hw2.py` for details.
3. (4pts) On the area $[-5,5]\times[-5,5]$, plot the contour lines of the following kernel SVMs, trained on the XOR data. Different kernels and the XOR data are provided in `hw2_utils.py`. Learning rate 0.1 and 10000 steps should be enough. To draw the contour lines, you can use `hw2.svm_contour()`. **Include your plots in the report.**
   - The polynomial kernel with degree 2.
   - The RBF kernel with $\sigma = 1$.
   - The RBF kernel with $\sigma = 2$.
   - The RBF kernel with $\sigma = 4$.