COMS 4771 HW4 (Spring 2023)

Due: Sun Apr 23, 2023 at 11:59pm

This homework is to be done **alone**. No late homeworks are allowed. To receive credit, a typesetted copy of the homework pdf must be uploaded to Gradescope by the due date. You must show your work to receive full credit. Discussing possible approaches for solutions for homework questions is encouraged on the course discussion board and with your peers, but you must write your own individual solutions and **not** share your written work/code. You must cite all resources (including online material, books, articles, help taken from/given to specific individuals, etc.) you used to complete your work.

1 From distances to embeddings

Your friend from overseas is visiting you and asks you the geographical locations of popular US cities on a map. Not having access to a US map, you realize that you cannot provide your friend accurate information. You recall that you have access to the relative distances between nine popular US cities, given by the following distance matrix D:

Distances (D)	BOS	NYC	DC	MIA	CHI	SEA	SF	LA	DEN
BOS	0	206	429	1504	963	2976	3095	2979	1949
NYC	206	0	233	1308	802	2815	2934	2786	1771
DC	429	233	0	1075	671	2684	2799	2631	1616
MIA	1504	1308	1075	0	1329	3273	3053	2687	2037
CHI	963	802	671	1329	0	2013	2142	2054	996
SEA	2976	2815	2684	3273	2013	0	808	1131	1307
SF	3095	2934	2799	3053	2142	808	0	379	1235
LA	2979	2786	2631	2687	2054	1131	379	0	1059
DEN	1949	1771	1616	2037	996	1307	1235	1059	0

Being a machine learning student, you believe that it may be possible to infer the locations of these cities from the distance data. To find an embedding of these nine cities on a two dimensional map, you decide to solve it as an optimization problem as follows.

You associate a two-dimensional variable x_i as the unknown latitude and the longitude value for each of the nine cities (that is, x_1 is the lat/lon value for BOS, x_2 is the lat/lon value for NYC, etc.). You write down the an (unconstrained) optimization problem

$$\text{minimize}_{x_1,\ldots,x_9} \quad \sum_{i,j} \big(\|x_i - x_j\| - D_{ij}\big)^2,$$

where $\sum_{i,j}(\|x_i - x_j\| - D_{ij})^2$ denotes the embedding discrepancy function.

(i) What is the derivative of the discrepancy function with respect to a location x_i?

(ii) Write a program in your preferred language to find an optimal setting of locations $x_1, \ldots, x_9$. You must submit your code to receive full credit.

(iii) Plot the result of the optimization showing the estimated locations of the nine cities. (here is a sample code to plot the city locations in Matlab)

```
>> cities={'BOS','NYC','DC','MIA','CHI','SEA','SF','LA','DEN'};
>> locs = [x1;x2;x3;x4;x5;x6;x7;x8;x9];
>> figure; text(locs(:,1), locs(:,2), cities);
```

What can you say about your result of the estimated locations compared to the actual geographical locations of these cities?

2 Kernelized SVM vs Neural Networks: Theory and Empirics

In this question, you will get a glimpse into some of the similarities and differences of kernelized SVMs (KSVM) and neural networks (NN) using both theory and empirics.

Understanding KSVMs for Regression: In lecture we covered kernelized SVMs for classification, but here we will consider a formulation for regression. For classification, we aimed to find the maximum margin decision boundary between our two classes. For regression, we can first study the linear case with one dimensional inputs and outputs. We aim to learn a predictor of the form $f(x) = wx + b$ given a set of observations (x_i, y_i) for $i \in [N]$. We now want to constrain the possible choices of our parameters, namely we want $\forall i \in [N] : |y_i - (wx_i + b)| \leq \epsilon$, where ϵ is a hyperparameter. This ensures that all our residuals are within ϵ distance.

(a) Assuming there exists such a predictor, we can formulate our optimization problem as follows:

$$\min_{w,b} \frac{1}{2} w^2$$

$$\text{such that: } \forall i \in [N] : |y_i - (wx_i + b)| \leq \epsilon.$$

Notice that the constraints are making sure that our prediction is within distance of ϵ. What role does the objective play in this optimization?

(Hint: consider the formulation of ridge regression)

(b) Of course, it is not guaranteed that all predictions will be within ϵ distance. Thus, we can add in slack variables as follows:

$$\begin{aligned} \min_{w,b,\xi,\gamma} \frac{1}{2} w^2 + C \sum_{i=1}^{N} (\xi_i + \gamma_i) & \\ \text{such that: } y_i - (wx_i + b) \leq \epsilon + \xi_i & \qquad \forall i \in [N] \\ (wx_i + b) - y_i \leq \epsilon + \gamma_i & \\ \xi_i \geq 0 & \\ \gamma_i \geq 0. & \end{aligned}$$

Describe in words the role of ξ_i and γ_i.

(c) Show that the learned predictor will take the form $f(x) = \sum_{i=1}^{N}(\alpha_i - \beta_i)x_i x + b$ by setting up the Lagrangian and examining the stationary points with respect to w.

So far our SVM formulation was for linear regression, where the prediction is done by

$$f(x) = \sum_{i=1}^{N}(\alpha_i - \beta_i)x_i x + b.$$

Note that by taking $\delta_i = \alpha_i - \beta_i$, and replacing the (inner) product between x_i and x by any non-linear kernel, we can now have a *kernelized* version as

$$f_{\text{kern}}(x) = \sum_{i=1}^{N}\delta_i K(x_i, x) + b.$$

For the remainder of the question we will be focusing on the RBF kernel.

Approximation Power of RBF KSVMs: RBF kernels are known to fit any arbitrary complex functions. Here, we will show specifically that our KSVM model can approximate any continuous function over the interval $[-1, 1]$ with arbitrary precision[1].

Let Z be the set of all possible RBF KSVM regressors of the type $[-1, 1] \to \mathbb{R}$ on a non-empty dataset, that is, $Z := \{f_{\text{kern}}\}$.

(d) Prove that Z is an algebra, that is, it is closed under: (i) addition ($\forall f_1, f_2 \in Z : (f_1 + f_2) \in Z$), (ii) multiplication ($\forall f_1, f_2 \in Z : f_1 f_2 \in Z$, i.e. element wise multiplication), and (iii) scalar multiplication ($\forall f_1 \in Z : \forall c \in \mathbb{R} : cf_1 \in Z$).

For multiplication, you may assume $\sigma = 1$ as the scaling parameter of the RBF kernel to simplify calculations.

(e) Prove that Z can "isolate" each point in $[-1, 1]$, that is, for a fixed dataset, there exists distinct $x, y \in [-1, 1] : \exists f \in Z : f(x) \neq f(y)$.

(f) Prove that $\forall x \in [-1, 1] : \exists f \in Z : f(x) \neq 0$.

Parts d-f collectively imply RBF KSVMs can approximate any continuous function on the interval $[-1, 1]$. [2]

Approximation Power of 2-Layer Neural Networks: A 2-layer neural network (with one-dimensional input, one-dimensional output, and an N-width hidden layer) is defined as

$$f_{\text{NN}}(x) := \sum_{i=1}^{N}\alpha_i \sigma(w_i x + b_i),$$

where w_i, b_i and α_i are the network weights, and σ is the "activation" function (usually a *sigmoid*, *ReLU*, or *tanh*).

[1] The set of continuous functions on the interval $[-1, 1]$ is denoted $C([-1, 1])$.

[2] One can combine these results, see e.g. Stone–Weierstrass Theorem, to show universal approximability.

Here we will show that ReLU activated 2-Layer Neural Networks, i.e. f_{NN} can approximate any function from $[-1, 1] \to \mathbb{R}$, by showing that ReLU activations are "discriminatory".[3]

An activation function σ is called discriminatory iff $\forall \mu \in C([-1, 1]), \forall w, b \in \mathbb{R}$,

$$\int_{-1}^{1} \sigma(wx + b)\mu(x)dx = 0 \implies \mu(x) = 0.$$

(g) Prove that the linear activation function, that is, $\sigma(x) = x$ is *not* discriminatory.
(Hint: think of a discrete case first.)

(h) Assuming that all continuous functions of the form $\psi(x) = \begin{cases} 1 & \text{as } x \to \infty \\ 0 & \text{as } x \to -\infty \end{cases}$ are discriminatory, prove that ReLU is discriminatory.
(Hint: try to construct a ψ type of function out of ReLUs and prove by contradiction.)

Comparing Empirical Performance of KSVMs and NNs: If both KSVMs and NNs have universal approximation (as seen in previous parts), then why are NNs more used in practice?

While both KSVMs and NNs are powerful models that can work well on arbitrarily complex datasets, on simpler datasets, we want models that require fewer training samples to yield good prediction. Here we will compare the relative performance of KSVMs and NNs on increasingly complex datasets and study which model class adapts better.[4]

Download the dataset provided. It contains train and test samples of various sizes of (x, y) pairs of functions with increasing complexity. For this question, you may use any library you want.

(i) To get a feel for the data, create scatter plots (x vs. y) using 50 training samples and 1000 training samples for each function complexity. (There are a total of 10 plots.)

(j) Train an RBF KSVM regressor and a NN for each function complexity with varying number of training samples.

Suggestions: For the SVM, it is advised you use SciKitLearn's the built in Support Vector Regression function. The default settings should work fine, just ensure that you specify the right kernel. For the NN, it is advised you use PyTorch. Using a small neural network with 2 or 3 hidden layers and a dozen or a few dozen neurons per layer with the Adam optimizer and ReLU activation function should work well. To squeeze out good performance, try changing the number of epochs and the batch size first. Additionally, see this reference for more training tips: http://karpathy.github.io/2019/04/25/recipe/. You must use the MSE loss. For training sample sizes, consider sizes 50, 100, 300, 500, 750, and 1000.

You must submit your code to receive credit.

(k) For both KSVM and NN predictors, and each function complexity, plot the test MSE error for varying training sizes.

[3] See e.g. "Approximation by Superpositions of a Sigmoidal Function" by Cybenko for a proof of why discriminatory activation functions imply universal approximation.

[4] For a more theoretical analysis of this topic, please see https://francisbach.com/quest-for-adaptivity/ and references therein.

(l) What can you conclude about the adapatibility of KSVMs vs NNs? Is one model better than the other? Analyze how the prediction quality varies with function complexity and training size.