CSE517a – Homework 3

M. Neumann

Feb 18 2019

- Please keep your written answers brief and to the point. Incorrect or rambling statements can hurt your score on a question.
- If your hand writing is not readable, we **cannot give you credit**. We recommend you type your solutions in LaTeX and compile a .pdf for each answer. **Start every problem on a new page!**
- This will be due THU **Feb 27 2020** at **10am** with an automatic 3-day extension.
  - If you want your homework to be graded before the midterm exam, then you will have to submit by this deadline. Do **not** use the extension.
  - If you decide to use the extension we cannot gurantee to grade your assignment before the midterm exam.
  - Once you have a submission before the posted deadline you may not update this submission anymore in the extension period.
- You may work in groups of at most 2 students.
- Submission instructions:
  - Start every problem on a **new page**.
  - Submissions will be exclusively accepted via **Gradescope**. Find instructions on how to get your Gradescope account and submit your work on the course webpage.

Problem 1 (*25 points*) Another Look at SVM

You suddenly have this vision, that it might be beneficial to pre-compute all inner-products in your data set. I.e. you store a $n \times n$ matrix $K_{ij} = x_i^\top x_j$

(a) (*10 pts*) Imagine you start the gradient descent procedure from above with initial weights $w^{(0)} = 0$ (this is just to help intuition). Take the gradient update of SVM (**hw1 p1(d)**) and show that after t gradient steps you can express the weight vector as $w^{(t)} = \sum_{i=1}^n \alpha_i^{(t)} x_i$ for some values $\alpha_i^{(t)}$. (HINT: each gradient step update the α's, if the update is correct you can always write $w^{(t)} = \sum_{i=1}^n \alpha_i^{(t)} x_i$ starting from update $t = 0$ to any update $t > 0$)

(b) (*10 pts*) Take this new definition $w = \sum_{i=1}^n \alpha_i x_i$ and substitute it into the loss function of SVM. You can now write the loss function as $\mathcal{L}(\alpha)$ where $\alpha = [\alpha_1 \ldots \alpha_n]^\top$ (no need to force the vector representation in your formula). Further, write the loss $\mathcal{L}(\alpha)$ only using the precomputed matrix entries K_{ij}.

(c) (*5 pts*) Can you derive a gradient descent update rule of SVM (**hw1, p1(d)**) with respect to α_i?

Problem 2 (*20 points*) Pairwise Euclidean Distances

Many machine learning algorithms (such as k-NN or the dual form of the SVM) access their input data primarily through pairwise distances. It is therefore important that we have a fast function that computes pairwise (Euclidean) distances of input vectors.

Assume we have two sets of input vectors $\{\mathbf{x}\}$ and $\{\mathbf{z}\}$ stored in two matrices $X = [\mathbf{x}_1, ..., \mathbf{x}_n] \in \mathbb{R}^{d\times n}$ where $\mathbf{x}_i \in \mathbb{R}^{d\times 1}$ and $Z = [\mathbf{z}_1, ..., \mathbf{z}_m] \in \mathbb{R}^{d\times m}$ where $\mathbf{z}_j \in \mathbb{R}^{d\times 1}$. Our objective is to calculate the Euclidean distances between *all* $\mathbf{x}_i$ and $\mathbf{z}_j$, denoted by the matrix $D \in \mathbb{R}^{n\times m}$ such that

$$D_{ij} = d(\mathbf{x}_i, \mathbf{z}_j) = \sqrt{(\mathbf{x}_i - \mathbf{z}_j)^T(\mathbf{x}_i - \mathbf{z}_j)}$$

Although there is an execution overhead per line in Python, matrix operations are extremely optimized and very fast with the numpy package. In the following you will transform the function above, so that it can be computed solely through matrix operations *without the use of any loops at all.*

(a) (*5 pts*) Show that the Gram matrix $G \in \mathbb{R}^{n\times m}$ (aka inner-product matrix) where

$$G_{ij} = \mathbf{x}_i^T \mathbf{z}_j$$

can be expressed in terms of matrix multiplication.

(b) (*10 pts*) Let us define two new matrices $S, R \in \mathbb{R}^{n\times m}$ where

$$S_{ij} = \mathbf{x}_i^T \mathbf{x}_i; R_{ij} = \mathbf{z}_j^T \mathbf{z}_j.$$

Show that the squared Euclidean distance matrix $D^2 \in \mathbb{R}^{n\times m}$, defined as

$$D_{ij}^2 = (\mathbf{x}_i - \mathbf{z}_j)^T(\mathbf{x}_i - \mathbf{z}_j)$$

can be expressed as a linear combination of S, R, and G. (HINT: It might help to first express D_{ij}^2 in terms of inner-products. HINT 2: Don't forget to show how you can compute S and R using numpy array operations.)

(c) (*5 pts*) Due to *numerical instability*, some of the entries in D may turn out to be less than `0`. How can you avoid this? Again use matrix operations in numpy.

Problem 3 (*30 points*) Valid Kernels, Kernel Construction

(a) (*10 pts*) Consider $\mathbf{x}_i \in \mathbb{R}^3$ and $\phi(\mathbf{x}_i) \in \mathbb{R}^D$ with

$$\phi(\mathbf{x}_i) = [1, \sqrt{2}(\mathbf{x}_i)_1, \sqrt{2}(\mathbf{x}_i)_2, \sqrt{2}(\mathbf{x}_i)_3, (\mathbf{x}_i)_1^2, (\mathbf{x}_i)_2^2, (\mathbf{x}_i)_3^2, \sqrt{2}(\mathbf{x}_i)_1(\mathbf{x}_i)_2, \sqrt{2}(\mathbf{x}_i)_1(\mathbf{x}_i)_3, \sqrt{2}(\mathbf{x}_i)_2(\mathbf{x}_i)_3]^T,$$

compute $\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$. What is D? Briefly argue why considering the inner products directly instead of explicitly computing the feature mappings is more efficient.

(b) (*5 pts*) For each of the following matrices show whether it is strictly positive definite, positive semidefinite, or neither:

$$A_1 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 0 \\ 1 & 0 & 2 \end{bmatrix}, \quad A_3 = \begin{bmatrix} 2 & 1 & -1 \\ 1 & 1 & 1 \\ -1 & 1 & 2 \end{bmatrix}.$$

(c) (*10 pts*) Show that the RBF Kernel corresponds to an inner product in an infinte dimensional space. (HINT: use Taylor expansion.)

(d) (*5 pts*) By using the **definition of a kernel**, verify that $\tilde{k}(\mathbf{x}, \mathbf{x}') = ck_1(\mathbf{x}, \mathbf{x}')$ is a valid kernel, given that $k(\mathbf{x}, \mathbf{x}')$ is a valid kernel and $c \geq 0$.

(e) **[ungraded]** Verify all other kernel construction rules as an additional practice.

HINT: to prove the rules there are a couple of different things to try. Note that not everything works for every rule.

- use the definition of a kernel (that is, construct a feature space, where $k(.,.)$ corresponds to the inner product)
- use the spectral decomposition $k_1(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x}) \phi_i(\mathbf{x}')$, where λ_i are the eigenvalues in the transformed feature space (cf. *Mercer's Theorem*)
- show that the Gram matrix corresponding to any arbitrary subset of elements in the input space is positive-semi definite by
  - showing that the quadratic form is non-negative for arbitrary vectors
  - showing that all eigenvalues are non-negative
- use already proven construction rules

Problem 4 (*25 points*) Kernelize the Perceptron Algorithm

Kernelize the perceptron algorithm for binary classification $y_i = \{+1, -1\}$.

```
(0) Initialize $\mathbf{w} = 0$
REPEAT until convergence:
    (1) Pick $(\mathbf{x}_i, y_i)$ randomly from $D$.
    (2) If $y_i \mathbf{w}^T \mathbf{x}_i \leq 0$ then make the update $\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i$, otherwise do nothing.
```

(a) (*5 pts*) Show that $\mathbf{w}$ can be written as a linear combination of the training inputs.

(b) (*10 pts*) Derive the kernelized classifier $h(\mathbf{x})$.

(c) (*10 pts*) State the kernelized version of the learning algorithm.