BUSI97287: Advanced Machine Learning Summer 2021 Imperial College Business School
Martin Haugh

Assignment 4

Due: 11.59pm Sunday 30^{the} May 2021

Rules

1. This is a group assignment. (There are approximately 3 people per group and by now you should know your assigned group.)
2. While `R` is the default package / programming language for this course you are free to use `R` or `Python` for the programming components of this assignment.
3. Within each group **I strongly encourage each person to attempt each question by his / herself first** before discussing it with other members of the group.
4. Students should **not** consult students in other groups when working on their assignments.
5. Late assignments will **not** be accepted and all assignments must be submitted through the Hub with one assignment submission per group. Your submission should include a PDF report with your answers to each question as well as any relevant code. Make sure your PDF clearly identifies each member of the group by CID and name.

1. **SVMs: Scaling the Inputs & Cross-Validation (20 marks)**
Suppose you have successfully trained an SVM with 10,000 training points and a Gaussian kernel where the values of C and σ were selected via cross-validation. You are then given an additional 40,000 training points and so you wish to retrain your SVM using the entire 50,000 training points that you now have. However, you wish to avoid the heavy computational expense associated with repeating the cross-validation exercise and so you therefore simply choose the SVM using the values of C and σ that you obtained with the earlier 10,000 training points. Do you see any potentially major problem with this? If so, what is it? Is there an easy fix?

Solution: There is a potentially major problem with doing this. To see it, recall that the SVM objective is to solve the optimisation problem

$$\min_{\mathbf{w},\boldsymbol{\xi},b} \ \frac{1}{2}\mathbf{w}^\top\mathbf{w} \ + \ C\sum_{i=1}^{n}\xi_i \tag{1}$$

subject to various constraints that we don't need to list here. There are two terms in the objective in (1): the first term, $\frac{1}{2}\mathbf{w}^\top\mathbf{w}$, does not depend on n whereas the second term, $C\sum_{i=1}^{n}\xi_i$, clearly scales linearly with n. Let C^* be the optimal value of C that we found from cross-validation when $n = 10,000$. If we now use C^* in (1) when $n = 50,000$ then the objective is putting a lot more weight on margin violations, i.e. training points where $\xi_i > 0$, than it did when $n = 10,000$. This relative re-weighting of the two terms in the objective in (1) will result in a potentially very different optimal classifier $(\mathbf{w}^*, b^*)$) which may generalize poorly due to the sub-optimal nature of C^* for the $n = 50,000$ data-set.

We can fix this by scaling appropriately. For example, we could always use $(C\sum_{i=1}^n \xi_i)/n$ instead of $C\sum_{i=1}^n \xi_i$ in (1). Then the optimal C found from cross-validation when $n = 10,000$ should also be a good choice when $n = 50,000$. In the specific problem above, this amounts to simply dividing C^* by 5 and then solving (1) with $n = 50,000$ without bothering to cross-validate again.

---

2. **Classifying Breast Tumours with SVMs (50 marks)**
In this question we will use the SVM functionality of the `e1071` library in `R` to build a classifier that predicts whether or not a breast tissue sample is malignant or benign. We will use the *BreastCancer* data-set which is part of the `mlbench` library / package in `R`/ (Further details can be found at `https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(original)`.)

   (a) Load the data-set and perform a brief summary EDA (exploratory data-analysis) of the data. You can and should remove the first column (Id) of the data-set since this just denotes the record number of the particular row / data-point. There are then 10 independent variables with the final column (Class) denoting the dependent variable that we wish to predict. How many row's have 1 or more NA values? You should remove these rows thereby leaving a data-set with 683 data-points. **(10 marks)**

   (b) Randomly assign 75% of your data to the training set and the remainder of your data to the test set. Then use cross-validation on your training set to build your classifier. Then fit an SVM with a **linear kernel** where the cost parameter C is chosen via cross-validation on the training set? Having found a good choice of C, refit your SVM on the entire training set. How does it perform on the test set? **(10 marks)**

   (c) Repeat part (c) but this time using a **polynomial kernel of degree 2**. (Again any parameters should be chosen via cross-validation before fitting on the entire training set. This also applies to parts (d) and (e) below.) **(10 marks)**

   (d) Repeat part (d) but this time using a **polynomial kernel of degree 3**. **(10 marks)**

   (e) Repeat part (e) but this time using a **Gaussian kernel**. **(10 marks)**

   Solution: The code for this question can be found in the `R` Notebook *SVM_breast_cancer.rmd*.

---

3. **Kernel K-Means Clustering (20 marks)**
Show that the K-means clustering algorithm can be kernelized, i.e. is amenable to the kernel trick.

Solution: Let $\phi(\mathbf{x}) \in \mathbb{R}^M$ be a feature vector where $\mathbf{x} \in \mathbb{R}^m$. (Typically $M >> m$ with possible $M = \infty$.) Recall in K-means clustering that the k^{th} cluster center satisfies

$$\mathbf{m}_k^{(t)} := \underset{\mathbf{m}\in\mathbb{R}^M}{\text{argmin}} \sum_{j\in\mathcal{C}_k^{(t)}} \|\phi(\mathbf{x}_j) - \mathbf{m}\|_2^2 = \frac{1}{|\mathcal{C}_k^{(t)}|} \sum_{j\in\mathcal{C}_k^{(t)}} \phi(\mathbf{x}_j)$$

for $k = 1, \ldots, K$ where K is the number of clusters and $|\mathcal{C}_k^{(t)}|$ is the number of points (in iteration t) assigned to cluster k. The cluster assignment of data-point i in iteration $t+1$ of the algorithm is

$$\begin{aligned} \mathcal{C}^{(t+1)}(i) &= \underset{k}{\text{argmin}} \left\| \phi(\mathbf{x}_i) - \mathbf{m}_k^{(t)} \right\|_2^2 \\ &= \underset{k}{\text{argmin}} \left\{ K(\mathbf{x}_i, \mathbf{x}_i) - \frac{2}{|\mathcal{C}_k^{(t)}|} \sum_{j\in\mathcal{C}_k^{(t)}} K(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{|\mathcal{C}_k^{(t)}|^2} \sum_{j,\ell\in\mathcal{C}_k^{(t)}} K(\mathbf{x}_j, \mathbf{x}_\ell) \right\} \quad (2) \end{aligned}$$

where $K(\mathbf{x}_i, \mathbf{x}_j) := \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$. (You should be sure to have derived (2) but it only takes a few lines of algebra and is omitted here.) The importance of (2) is that the cluster assignment can be computed without reference to ϕ. In particular, only the kernel function $K(\cdot, \cdot)$ is required! The kernel K-means clustering algorithm proceeds by computing (2) for all training points $i = 1, \ldots, N$ in iteration $t+1$, and then computing the $|\mathcal{C}_k^{(t+1)}|$'s before proceeding to iteration $t+2$. We continue iterating until convergence (to a local minimum) occurs.

Remark: You might protest that ϕ is required to compute the $\mathbf{m}_k^{(t)}$'s and indeed this is true. But you don't actually need the $\mathbf{m}_k^{(t)}$'s to do K-means clustering.

Kernel K-means clustering can be implemented in `R` (with various kernels possible) via the `kkmeans` function from the `kernlab` library. Further details are available at

`https://www.rdocumentation.org/packages/kernlab/versions/0.9-25/topics/kkmeans`.

---

4. **Kernel Ridge Regression (20 marks)**
Recall that ridge regression solves the problem

$$\min_{\boldsymbol{\beta}} \left\{ \frac{1}{2} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \frac{\lambda}{2} \boldsymbol{\beta}^\top \boldsymbol{\beta} \right\}$$

and it has the optimal solution

$$\hat{\boldsymbol{\beta}}_{\text{R}} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_d)^{-1} \mathbf{X}^\top \mathbf{y} \quad (3)$$

where $\mathbf{X}$ is $(n \times d)$, $\mathbf{y}$ is $(n \times 1)$ and $\boldsymbol{\beta}$ is $(d \times 1)$. You can assume that the data has been pre-processed so that there is no intercept parameter β_0. Show that we can apply the kernel trick to ridge regression.

Hint: You will need the following matrix identity. Let $\mathbf{P}$, $\mathbf{B}$ and $\mathbf{R}$ denote matrices of conformable sizes. Then

$$\left(\mathbf{P}^{-1} + \mathbf{B}^\top \mathbf{R}^{-1} \mathbf{B}\right)^{-1} \mathbf{B}^\top \mathbf{R}^{-1} = \mathbf{P}\mathbf{B}^\top \left(\mathbf{B}\mathbf{P}\mathbf{B}^\top + \mathbf{R}\right)^{-1}. \tag{4}$$

Use (4) to show that our prediction at a new point, $\mathbf{x}^{\text{new}}$ say, only depends on the kernel function and not the feature vector that corresponds to the kernel function.

Solution: Take $\mathbf{P} = \lambda^{-1}\mathbf{I}_d$, $\mathbf{B} = \mathbf{X}$ and $\mathbf{R} = \mathbf{I}_n$ in (4) to obtain

$$\begin{aligned} \hat{\boldsymbol{\beta}}_{\text{R}} &= \lambda^{-1}\mathbf{X}^\top \left(\lambda^{-1}\mathbf{X}\mathbf{X}^\top + \mathbf{I}_n\right)^{-1} \mathbf{y} \\ &= \mathbf{X}^\top \left(\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I}_n\right)^{-1} \mathbf{y}. \end{aligned}$$

Our prediction $\hat{y}(\mathbf{x}^{\text{new}})$ at a new point, $\mathbf{x}^{\text{new}}$, will therefore be

$$\hat{y}(\mathbf{x}^{\text{new}}) = \hat{\boldsymbol{\beta}}_{\text{R}}^\top \mathbf{x}^{\text{new}} = \mathbf{y}^\top \left(\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I}_n\right)^{-1} \mathbf{X}\,\mathbf{x}^{\text{new}} \tag{5}$$

Note that (5) is now amenable to the kernel trick as it only depends on $\mathbf{X}$ via inner products. We therefore have the following more general version of (5)

$$\hat{\boldsymbol{\beta}}_{\text{R}}^\top \mathbf{x}^{\text{new}} = \mathbf{y}^\top \left(\mathbf{K} + \lambda\mathbf{I}_n\right)^{-1} \mathbf{k}(\mathbf{x}^{\text{new}})$$

where we recall $\mathbf{K} = \boldsymbol{\phi}\boldsymbol{\phi}^\top$ is the $n \times n$ Gram matrix with

$$\mathbf{K}_{ij} := \boldsymbol{\phi}(\mathbf{x}_i)^\top \boldsymbol{\phi}(\mathbf{x}_j) =: k(\mathbf{x}_i, \mathbf{x}_j)$$

and $\mathbf{k}(\mathbf{x}^{\text{new}})$ is the $n \times 1$ vector with i^{th} element $k(\mathbf{x}_i, \mathbf{x}^{\text{new}})$.