CSCI 6971/4971: Homework 3

Assigned Monday March 11 2018. Due at beginning of class Monday March 25 2019.

Remember to typeset your submission, and label it with your name. Coding should be done in Python+NumPy+Sklearn. Attach a printout of your code; label all plots appropriately and structure and comment your code neatly and appropriately.

Hand in a physical submission. Do not send me electronic copies of your writeup or code unless I ask for them.

1. [60 points] **Multi-target linear regression for classification of the MNIST image dataset**

   Consider the multitarget ridge regression problem

   $$\mathbf{X}_\star = \text{argmin}_{\mathbf{X}} \ \frac{1}{n}\|\mathbf{A}\mathbf{X} - \mathbf{B}\|_F^2 + \lambda\|\mathbf{X}\|_F^2,$$

   that takes a feature matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$ and a matrix $\mathbf{B} \in \mathbb{R}^{n\times m}$ whose columns represent m different targets. The solution to this system is given by

   $$\mathbf{X}_\star = (\mathbf{A}^T\mathbf{A} + n\lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{B}.$$

   (a) Write a function `(X, numIters, timeInside) = FactorizedRegularizedCG(A, Y, gamma, maxIters, eps)` that takes feature matrix $\mathbf{A}$, target matrix $\mathbf{Y}$, regularization constant γ, maximum number of iterations `maxIters`, and convergence tolerance ϵ and computes the solution to

   $$\mathbf{X}_\star = (\mathbf{A}^T\mathbf{A} + \gamma\mathbf{I})^{-1}\mathbf{Y} \tag{1}$$

   using CG[1]. *Do not explicitly compute* $\mathbf{A}^T\mathbf{A} + \gamma\mathbf{I}$, because when d is truly large, this matrix cannot be stored.

   Return $\mathbf{X}$, the number of iterations of CG needed for all the targets to converge, and the time spent in the function (in milliseconds). Document the function with comments explaining the problem it solves, the inputs, return arguments, termination criteria, and any other relevant information.

   Consider the ith target as resolved when the ith residual is small relative to the norm of that target:

   $$\|(\mathbf{A}^T\mathbf{A} + \gamma\mathbf{I})\mathbf{x}_i - \mathbf{y}_i\|_2 \leq \varepsilon\|\mathbf{y}_i\|_2,$$

   where $\mathbf{x}_i$ and $\mathbf{y}_i$ are the ith columns of $\mathbf{X}$ and $\mathbf{Y}$; continue to apply CG on *all* the targets until either *all* the targets have been resolved or the maximum number of iterations is reached.

   (b) Download the MNIST[2] training dataset from `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html#mnist` and load these with `sklearn.datasets.load_svmlight_file` to obtain[3] the feature matrix $\mathbf{A}$ and a target vector $\mathbf{b}$ that records the class (for this dataset, from 0 to 9) to which the instances belong.

   (c) Write a function `B = OneHotEncoding(b, numClasses)` that returns a matrix $\mathbf{B}$ with `numClasses` columns, whose ith row is $\mathbf{e}_{j+1}^T$ if $b_i = j$.

   Set $\mathbf{B} =$ `OneHotEncoding`$(\mathbf{b}, 10)$.

[1] See, e.g., `https://en.wikipedia.org/wiki/Conjugate_gradient_method` for an algorithm listing. Usually the CG algorithm is stated for a single right-hand side. To apply it to multiple right-hand sides, apply the algorithm to each target vector simultaneously. For efficiency, be sure to use vectorization and matrix-based operations as much as possible: e.g., compute the residual matrix as a matrix multiply and a matrix subtract instead of looping over each target vector and computing their residuals sequentially.

[2] See `https://en.wikipedia.org/wiki/MNIST_database`

[3] We really should augment the raw features by concatenating a constant 1 to the feature vectors in order to allow the model to learn a linear intercept if needed, but for simplicity, we do not do this.

(d) Compute solutions to the MNIST multitarget ridge regression problem for predicting the classes of the MNIST images, using `FactorizedRegularizedCG` with `maxIters = 500`, `eps = 1e-14`, and varying λ among $\{0.0065, 0.065, 0.65, 6.5, 65, 650\}$ (note this range is for λ, not γ). For each of these regularization values, report the RMSE training error

$$\frac{1}{\sqrt{n}}\|\mathbf{A}\mathbf{X}_\star - \mathbf{B}\|_F,$$

the training misclassification rate (in percentage), and the training time (in seconds), in the form

```
lambda = %f:  RMSE = %f, Misclassification Rate(%)) = %f, Training
                        Time(s) = %f
```

To calculate the training misclassification rate, first convert the predicted targets $\widehat{\mathbf{B}} = \mathbf{A}\mathbf{X}_\star$ into a class prediction; to do so, assign each instance to the class whose indicator vector is closest to its predicted class vector. Specifically, let $\widehat{\mathbf{b}}^i$ indicate the ith row of $\widehat{\mathbf{B}}$, then we say instance i is predicted to be in class j if the largest entry of $\widehat{\mathbf{b}}^i$ is its $j+1$th entry.

(e) Comment on your observations.

2. [40 points] **Nonlinear learning using Random Fourier Features**

Now we would like to (potentially greatly) decrease the misclassification rate by using nonlinear features.

(a) The canonical way of fitting a nonlinear model, by using kernel ridge regression, requires forming an $n \times n$ kernel matrix. In the case of the MNIST dataset, this is not feasible on most laptop machines. Assuming that we stored the kernel matrix in double precision (8 bytes per entry), how many gigabytes of RAM would be required to store a kernel matrix?

(b) Instead, we will use the Random Fourier Feature Map approach: we will replace the infinite-dimensional feature map $\phi_{\text{RBF}} : \mathbb{R}^d \to \mathbb{R}^\infty$ implicitly used in Gaussian kernel ridge regression[4] with a random finite-dimension approximate feature map $\mathbf{z}_{\text{RFF}} : \mathbb{R}^d \to \mathbb{R}^D$.

Specifically, given a bandwidth parameter σ^2 and a number of nonlinear features D, we define the random feature map

$$\mathbf{z}_{\text{RFF}}(\mathbf{x}) = \sqrt{\frac{2}{D}}[\cos(\omega_1^T\mathbf{x} + b_1), \ldots, \cos(\omega_D^T\mathbf{x} + b_D)]^T,$$

where the frequency vectors $\omega_i \in \mathbb{R}^d$ are independent samples from the $\mathcal{N}(0, 1/(2\sigma^2)\mathbf{I}_d)$ distribution, and the shift vector $\mathbf{b} \in \mathbb{R}^D$ consists of independent samples from the Uniform$(0, 2\pi]$ distribution.

Note that $\mathbf{z}_{\text{RFF}}$ is a random function: its value depends on the randomly sampled frequency vectors and offsets. One can show that with high probability this function does a good job of approximating ϕ_{RBF} (in a certain sense). The advantage of using $\mathbf{z}_{\text{RFF}}$ is that it is finite-dimensional, so we can explicitly solve the regression problem instead of using the kernel formulation. In particular, our matrix of nonlinear features $\mathbf{Z}$ is in $\mathbb{R}^{n\times D}$, and we can use ridge regression as in the problem above, this time on $\mathbf{Z}$ instead of $\mathbf{A}$.

Write a function `Z = computeRFF(A, W, b)` that takes raw features $\mathbf{A} \in \mathbb{R}^{n\times d}$, frequencies $\mathbf{W} \in \mathbb{R}^{D\times d}$ whose rows are the frequency vectors, and offsets $\mathbf{b} \in \mathbb{R}^D$, and computes

[4] Using the kernel function $\kappa_{\text{RBF}}(x) = \exp(-\|x-y\|_2^2/(2\sigma^2))$.

a matrix $\mathbf{Z} \in \mathbb{R}^{n \times D}$ of nonlinear features, whose rows consist of the application of $\mathbf{z}_{\mathrm{RFF}}$ to the corresponding rows of $\mathbf{A}$. Be careful about making this function efficient: vectorize as much as possible, and avoid for loops.

(c) Use `FactorizedRegularizedCG` as above to solve the kernel ridge regression problem for MNIST using $\mathbf{Z}$ as the features, with $D = 10^4$ features, `maxIters = 500`, `eps=1e-14`, and $\lambda = 0.001$. Search (by hand, if you like) over the values of σ^2 to find one that gives the lowest misclassification rate you can achieve. Hint: try searching logarithmically over a range of $\sigma^2 = .1$ to 4000 and refining your search by e.g., binary search.

Report on your observations (and the value of σ^2 you ultimately used), and compare the results to those from the previous problem.

(d) Write a function `Ascaled = RescaleToUnitInterval(A)` that takes raw features $\mathbf{A} \in \mathbb{R}^{n \times d}$ and returns $\mathbf{A}/255$. Apply this function to the raw features to obtain `Ascaled` and repeat the process in the preceeding subproblem to achieve the lowest misclassification rate that you can. The results should demonstrate the importance of preprocessing your raw features before using non-linear methods.

Report on your observations (and the value of σ^2 you ultimately used), and compare the results to those from the previous problem and the preceeding subproblem.