CS 542: Machine Learning **Summer 2019**

Problem Set 4

Lecturer: Prof. Peter Chin *Due: June 26, 2019*

◇ Please submit your solutions via Gradescope by 23:59PM on the due date. The submission will be under one assignment, and must include

- Written solutions, typed or handwritten, pdf or images.
- A written report, standalone or as part of a Jupyter notebook.
- Code, either by itself as a pdf or image, or as part of a notebook.

◇ Late policy: there will be a penalty of 10% per day, up to three days late. After that no credit will be given.

1. (40 points) Written Problems

(a) (8 points) The linear regression error function (3.12) can be written

$$E_D(\mathbf{w}) = \frac{1}{2}\left(\mathbf{t} - \Phi\mathbf{w}\right)^{\mathrm{T}}\left(\mathbf{t} - \Phi\mathbf{w}\right).$$

Prove that $\nabla_{\mathbf{w}} E_D(\mathbf{w}) = -\Phi^{\mathrm{T}}\left(\mathbf{t} - \Phi\mathbf{w}\right)$.

(b) (8 points) Bishop 6.2

(c) (8 points) Bishop 7.3

(d) (8 points) Bishop 8.11

(e) (8 points) Bishop 8.13

2. (30 points) Support Vector Machine

We've attached a dataset, `MNIST_data.mat`, containing a sample of the famout MNIST benchmark[1]. Your report must provide summaries of each method's performance and some additional details of your implementation. Compare the relative strengths and weaknesses of the methods based on both the experimental results and your understanding of the algorithms.

You can load the data with `scipy.io.loadmat`, which will return a Python dictionary containing the test and train data and labels.

The purpose of this assignment is for you to implement the SVM. You are not allowed to import an SVM from, for instance, `scikit-learn`. You may, however, use a library (such as `scipy.optimize.minimize` or `cvxopt.solvers.qp`) for the optimization process.

[1] `http://yann.lecun/com/exdb/mnist`

Here are your instructions:

(a) Develop code for training an SVM for binary classification with nonlinear kernels. You'll need to accomodate non-overlapping class distributions. One way to implement this is to maximize (7.32) subject to (7.33) and (7.43). It may be helpful to redefine these as matrix operations. Let $\mathbb{1} \in \mathbb{R}^{N\times 1}$ be the vector whose entries are all 1's. Let $\mathbf{a} \in \mathbb{R}^{N\times 1}$ have entries a_i. Let $\mathbf{T} \in \mathbb{R}^{N\times N}$ be a diagonal matrix with $\mathbf{T}_{ii} = t_i$ on the diagonal. Then we can reformulate the objective to be

$$\text{maximize} \quad \tilde{L}(\mathbf{a}) = \mathbb{1}^{\mathrm{T}}\mathbf{a} - \frac{1}{2}\mathbf{a}^{\mathrm{T}}\mathbf{TKTa}$$

subject to

$$\begin{aligned} \mathbb{1}^{\mathrm{T}}\mathbf{a} &\preceq C \\ \mathbb{1}^{\mathrm{T}}\mathbf{a} &\succeq 0 \\ \mathbf{a}^{\mathrm{T}}\mathbf{t} &= 0. \end{aligned}$$

The "$\preceq$" symbol here means element-wise comparison. This formulation is very close to what `cvxopt` expects.

(b) Develop code to predict the $\{-1, +1\}$ class for new data. To use the predictive model (7.13) you need to determine b, which can be done with (7.37).

(c) Using your implementation, compare multiclass classification performance of two different voting schemes:

   i. one versus rest

   ii. one versus one

(d) The parameter $C > 0$ controls the tradeoff between the size of the margin and the slack variable penalty. It is analogous to the inverse of a regularization coefficient. Include in your report a brief discussion of how you found an appropriate value.

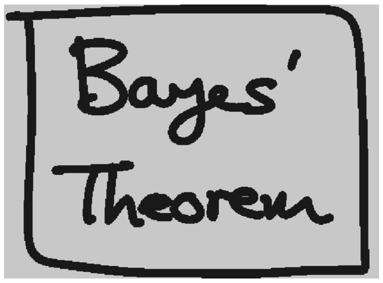
(e) In addition to calculating percent accuracy, generate multiclass confusion matrices[2] as part of your analysis.

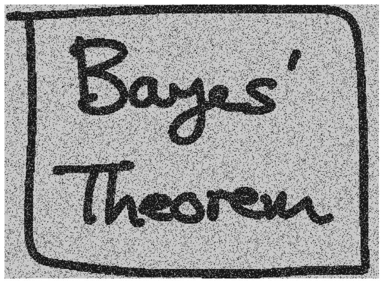
3. **(30 points) Denoising with a Graphical Model**

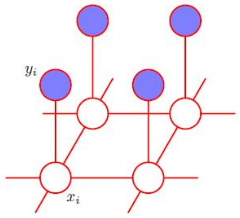
Tip: Solve Bishop 8.13 before implementing this problem.

We will implement the example from Bishop Section 8.3.3, applying a simple graphical model to remove noise from an image. We've taken a binary image and randomly flipped 10% of the pixels to the other value. For pixel i, we observe $y_i \in \{-1, +1\}$. Our goal is to recover the original (unobserved) pixel value x_i for each pixel.

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


(a) Clean image


(b) Noisy image



(c) MRF

We explicitly define an objective function to minimize. Borrowing from statistical physics, this is often called the "energy function." Pick parameters h, β, and η, and the energy function is

$$E(\mathbf{x}, \mathbf{y}) = h \sum_{i \text{ pixels}} x_i - \beta \sum_{\substack{\{i,j\} \\ \text{neighbors}}} x_i x_j - \eta \sum_{i \text{ pixels}} x_i y_i \tag{4.1}$$

Most observations will be unflipped, so we have a term $-\eta x_i y_i$ with $\eta \geq 0$ to encourage that. We also believe neighboring pixels will tend to have the same value, so we have $-\beta x_i x_j$ with $\beta \geq 0$ for every set of neighbors. We define "neighbors" here to be pixels that share a side, so most pixels have four neighbors. We include a term h to allow us to prefer one type of pixel over the other.

We can turn (4.1) into a probability distribution by writing

$$p(\mathbf{x}, \mathbf{y}) \propto \exp\{-E(\mathbf{x}, \mathbf{y})\}$$

This is what we'd like to maximize, so we can equivalently try to minimize the energy function. We'll do this optimization with an algorithm called "iterated conditional modes," which is a type of coordinate-wise descent algorithm. It's very simple:

(a) For all pixels i, initialize $x_i \leftarrow y_i$

(b) Loop over i, either randomly or in a fixed order. For each i, check if flipping x_i would decrease the energy function. If so, flip it.

(c) Stop when no more pixels can be flipped.

Some questions to consider: Will the algorithm always converge? If so, will we always find the lowest energy? Will the lowest energy always correspond to the original image? Include an intuitive discussion of these points in your report, as well as your best recovered image.

Implementation Details: We have included the noisy image `Bayes_noisy.png` and the original `Bayes_true.png` for comparison. They can be read with Scipy (`from scipy import misc`) using `img = misc.imread('Bayes_noisy.png')`. The pixel values are 25 and 200, so you'll need to translate between these and the ± 1 formulation in the book. Bishop suggests reasonable starting parameters on page 390, but you should try others.