Midterm 1 Practice for COMP6321 Fall 2019

The questions in this practice midterm are suggestive only of the *style* and *difficulty* of questions that will be asked on on the real midterm. The length and the particular course content evaluated will be different.

Q1. [10 marks total] This question is about *logistic regression models*.

a) [2 marks] What kind of learning task is logistic regression used for?

> Binary classification

b) [1 mark] Can the optimal parameter vector $\boldsymbol{w}$ for a logistic regression problem be solved for 'directly'?

> No.

c) [2 marks] Is the decision boundary of logistic regression linear or non-linear within the feature space $\boldsymbol{\phi}$? Explain.

> Linear. The standard decision boundary is all points $\boldsymbol{\phi}$ where $\sigma(\boldsymbol{w}^T\boldsymbol{\phi}) = 0.5$. This is the same set of points as those satisfying $\boldsymbol{w}^T\boldsymbol{\phi} = 0$. This is linear in $\boldsymbol{\phi}$ so the decision boundary will be linear. Even if we chose a different decision threshold than 0.5, say γ, the decision boundary would be all points $\boldsymbol{w}^T\boldsymbol{\phi} = \sigma^{-1}(\gamma)$ which is still linear in $\boldsymbol{\phi}$ since $\sigma^{-1}(\gamma)$ is constant.

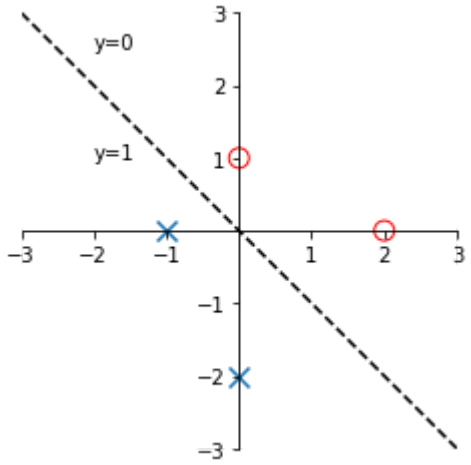
d) [3 marks] Assume you are given data set $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$. Write the *logistic regression* loss function with respect to this data set. For full marks include the feature transformation $\boldsymbol{\phi}(\cdot)$.

$$\ell_{\mathrm{LR}}(\boldsymbol{w}) = -\sum_{i=1}^{N} y_i \ln \hat{y}_i + (1 - y_i)\ln(1 - \hat{y}_i) \qquad \text{where } \hat{y}_i = \sigma(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)).$$

e) [2 marks] Assume you are given training set in matrix format $\begin{bmatrix}1 & x_1 & x_2\end{bmatrix}$ where X and $\boldsymbol{y}$ are

$$X = \begin{bmatrix}1 & -1 & 0\\ 1 & 0 & -2\\ 1 & 0 & 1\\ 1 & 2 & 0\end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix}1\\ 1\\ 0\\ 0\end{bmatrix}$$

Plot the data in two dimensions and draw the decision boundary that would result from applying logistic regression. Be sure to indicate which side corresponds to predicting $y = 1$.



Q2. [10 marks total] Assume we have samples $\{x_1, \ldots, x_N\}$ from a univariate normal distribution $\mathcal{N}(\mu, \sigma^2)$. The likelihood $p(x \mid \mu, \sigma)$ having observed a single point x is therefore

$$\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

a) [2 marks] The likelihood is a function of which variable(s)?

μ and σ

b) [2 marks] Write the likelihood $p(x_1, \ldots, x_N \mid \mu, \sigma)$ having observed all x_i jointly.

$$\prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right)$$

c) [2 marks] Write the negative log likelihood of $p(x_1, \ldots, x_N \mid \mu, \sigma)$.

$$
\begin{aligned}
&= -\ln \left[\prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2} \right) \right] \\
&= -\sum_{i=1}^{N} \ln \left[\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2} \right) \right] \\
&= N \ln \sqrt{2\pi}\sigma + \frac{1}{2\sigma^2} \sum_{i=1}^{N} (x_i - \mu)^2
\end{aligned}
$$

d) [2 marks] Write the gradient of the negative log likelihood of $p(x_1, \ldots, x_N \mid \mu, \sigma)$.

Differentiate with respect to μ:

$$
\begin{aligned}
&= \frac{\partial}{\partial \mu} \left[N \ln \sqrt{2\pi}\sigma + \frac{1}{2\sigma^2} \sum_{i=1}^{N} (x_i - \mu)^2 \right] \\
&= \frac{1}{2\sigma^2} \sum_{i=1}^{N} \frac{\partial}{\partial \mu} \left[(x_i - \mu)^2 \right] \\
&= \frac{1}{\sigma^2} \sum_{i=1}^{N} (\mu - x_i)
\end{aligned}
$$

Differentiate with respect to σ:

$$
\begin{aligned}
&= \frac{\partial}{\partial \sigma} \left[N \ln \sqrt{2\pi}\sigma + \frac{1}{2\sigma^2} \sum_{i=1}^{N} (x_i - \mu)^2 \right] \\
&= \frac{N}{\sigma} + \frac{\partial}{\partial \sigma} \left[\frac{1}{2\sigma^2} \right] \sum_{i=1}^{N} (x_i - \mu)^2 \\
&= \frac{N}{\sigma} - \frac{1}{\sigma^3} \sum_{i=1}^{N} (x_i - \mu)^2
\end{aligned}
$$

So the gradient is $\begin{bmatrix} \frac{1}{\sigma^2} \sum_{i=1}^{N} (\mu - x_i) \\ \frac{N}{\sigma} - \frac{1}{\sigma^3} \sum_{i=1}^{N} (x_i - \mu)^2 \end{bmatrix}$

e) [2 marks] Use your answer from part (d) to derive a maximum likelihood estimate of the normal distribution parameters.

We must solve the system of equations

$$0 = \frac{1}{\sigma^2}\sum_{i=1}^{N}(\mu - x_i)$$

$$0 = \frac{N}{\sigma} - \frac{1}{\sigma^3}\sum_{i=1}^{N}(x_i - \mu)^2$$

By inspection we can first solve for μ_{ML} independently of σ, and then use μ_{ML} to solve for σ_{ML}.

Solve for μ_{ML}:

$$0 = \frac{1}{\sigma^2}\sum_{i=1}^{N}(\mu - x_i)$$

$$\Rightarrow 0 = \sum_{i=1}^{N}(\mu - x_i)$$

$$\Rightarrow 0 = N\mu - \sum_{i=1}^{N} x_i$$

$$\Rightarrow \mu_{\text{ML}} = \frac{1}{N}\sum_{i=1}^{N} x_i = \bar{x} \text{ (the mean of the data)}$$

Solve for σ_{ML}:

$$0 = \frac{N}{\sigma} - \frac{1}{\sigma^3}\sum_{i=1}^{N}(x_i - \mu)^2$$

$$\Rightarrow N = \frac{1}{\sigma^2}\sum_{i=1}^{N}(x_i - \mu)^2$$

$$\Rightarrow \sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$$

$$\Rightarrow \sigma_{\text{ML}} = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N}}$$

Q3. [8 marks total] This question is about programming machine learning concepts with Numpy. You can assume that `import numpy as np` has already been run.

a) [2 marks] You are given the following incomplete function:

```
def linear_model_predict(X, w):
    """
    Returns predictions from linear model y(X, w) at each point X[i,:] using parameters w.
    Given X with shape (N,D) and w with shape (D,), returns predictions with shape (N,).
    """
```

Complete the function in the space below. (No need to copy the above function signature.) For full marks, your answer should be fully vectorized.

```
        return X @ w
```

b) [2 marks] You are given the following incomplete function:

```
def sigmoid(z):
    """Returns the element-wise logistic sigmoid of array z."""
```

Complete the function in the space below. (No need to copy the above function signature.) For full marks, your answer should be fully vectorized.

```
        return 1 / (1 + np.exp(-z))
```

c) [4 marks] You are given the following incomplete function:

```
def linear_regression_by_gradient_descent(X, y, w_init, learn_rate=0.05, num_steps=500):
    """
    Fits a linear model by gradient descent.

    Given X, y, and w_init with shapes (N,D), (N,), and (D,) respectively,
    returns a new parameter vector w that minimizes the squared error to the targets
    by running num_steps of gradient descent.
    """
```

The gradient of a linear model can be expressed mathematically as

$$\nabla \ell_{\mathrm{LS}}(\boldsymbol{w}) = (X^T X)\boldsymbol{w} - X^T \boldsymbol{y}$$

Complete the function in the space below. (No need to copy the above function signature.) For full marks, your answer should be fully vectorized.

```
    w = w_init
    for i in range(num_steps):
        grad = (X.T @ X) @ w - X.T @ y
        w = w - learn_rate*grad
    return w
```

Q4. [6 marks total] This question is about the assigned reading: the 2001 paper by Leo Breiman.

a) [2 marks] How are what Breiman calls "data models" typically validated? How are what he calls "algorithmic models" typically validated?

> *Data models* are typically validated by measuring "goodness-of-fit" and analyzing residuals. A model is either accepted or rejected based on that.
>
> *Algorithmic models* are typically validated by measuring their predictive accuracy, ideally on a held-out test set.

b) [3 marks] Give three examples of what Breiman calls "data models" and three examples of what he calls "algorithmic models"

> Data models: linear regression, logistic regression, Cox model.
>
> Algorithmic models: decision trees, support vector machines, neural networks.

c) [1 mark] What is Breiman's argument against doing dimensionality reduction?

It reduces the amount of information available to the model.

Q5. [8 marks total] This question is about the *K-means* clustering algorithm.

a) [4 marks] What is the optimization problem that the *K*-means clustering algorithm tries to solve? Express your answer mathematically, then explain what each symbol means.

The *K*-means algorithm is minimizing the following problem:

$$\begin{aligned} &\min_{\boldsymbol{r},\boldsymbol{\mu}} \sum_{i=1}^{N}\sum_{k=1}^{K} r_{ik} \left\|\boldsymbol{x}_i - \boldsymbol{\mu}_k\right\|^2 \\ &\text{s.t.} \sum_{k=1}^{K} r_{ik} = 1 \text{ for all } i = 1,\ldots,N \\ &\quad r_{ik} \in \{0,1\} \text{ for all } i = 1,\ldots,N \text{ and } k = 1,\ldots,K \end{aligned}$$

where N is the number of data points, $\boldsymbol{x}_i$ is the i^{th} data point, K is the number of clusters, $\boldsymbol{\mu}_k$ is the mean for cluster k, and r_{ik} indicates whether data point i is assigned to cluster k or not.

b) [1 mark] In the *K*-means optimization problem, the objective function has a name. What is it?

It is sometimes called the "distortion," a measure of how far (how distorted) each data point is from its mean.

c) [1 mark] Why is K-means considered a "coordinate descent" algorithm?

Because at each step it minimizes over a subset of its parameters (coordinates) at a time, holding the other parameters fixed.

d) [1 mark] Write the formula for computing the "assignment step" of the *K*-means algorithm

The assignment step assigns new values to the r_{ik} variables according to:

$$r_{ik} = \begin{cases} 1 & \text{if } k = \arg\min_j \left\|\boldsymbol{x}_i - \mu_j\right\|^2 \\ 0 & \text{otherwise} \end{cases}$$

e) [1 mark] Prove that your answer to (d) is an optimal assignment with respect to the current means.

Suppose for i^{th} data point the assignment step assigned $r_{ik} = 1$. If there were a j such that assigning $r_{ij} = 1$ would decrease the *K*-means objective, this would imlpy that $\left\|\boldsymbol{x}_i - \boldsymbol{\mu}_j\right\|^2 < \left\|\boldsymbol{x}_i - \boldsymbol{\mu}_k\right\|^2$. However, this is not possible because $k = \arg\min_j \left\|\boldsymbol{x}_i - \boldsymbol{\mu}_j\right\|^2$. Therefore the hard assignment step is optimal.

Q6. [3 marks] The probability of a point $x \in \mathbb{R}$ under the univariate normal distribution $\mathcal{N}(\mu, \sigma^2)$ is

$$p(x \mid \mu, \sigma) = \frac{1}{Z} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right) \text{ where } Z = \sqrt{2\pi}\sigma$$

Write the probability of a point $\boldsymbol{x} \in \mathbb{R}^D$ under the multivariate normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. (It is OK to just write Z for the new normalization constant without specifying its value.) Explain how your answer reduces to the univariate case when $D = 1$.

The probability of $\boldsymbol{x}$ under the multivariate normal distribution is

$$p(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{Z} \exp\left(-\frac{1}{2}\left\|\boldsymbol{x} - \boldsymbol{\mu}\right\|_{\boldsymbol{\Sigma}^{-1}} \left\|\boldsymbol{x} - \boldsymbol{\mu}\right\|\right)$$

When $D = 1$ we have $\|x - \mu\| = |x - \mu|$ and $\boldsymbol{\Sigma} = \left[\sigma^2\right]$ so the above reduces to

$$p(x \mid \mu, \sigma) = \frac{1}{Z} \exp\left(-\frac{1}{2}\left|x - \mu\right| \sigma^{-2} \left|x - \mu\right|\right) = \frac{1}{Z} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right)$$

which is the univariate case.

Q7. [6 marks total] This question is about the *expectation maximization* (EM) algorithm for fitting a K-component Gaussian mixture model to a data set $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^N$.

a) [4 marks] Write the formula for the density $p(\boldsymbol{X} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ of a Gaussian mixture model with parameters $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_K\}$, $\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K\}$, and $\boldsymbol{\Sigma} = \{\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K\}$.

$$p(\boldsymbol{X} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{i=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

b) [2 marks] The EM algorithm is derived from a probablistic model of how each $\boldsymbol{x}$ is generated. This model is based on a component selection vector $\boldsymbol{z}$. Prove that $p(z_k = 1 \mid \boldsymbol{x}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$, *i.e.* the probability that component k was used to generate data point $\boldsymbol{x}$, is equal to

$$\frac{\pi_k \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

Applying Bayes theorem gives

$$p(z_k = 1 \mid \boldsymbol{x}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{p(\boldsymbol{x} \mid z_k = 1, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})\, p(z_k = 1 \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{p(\boldsymbol{x} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})}$$

Now we simplify:

- $p(z_k = 1 \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = p(z_k = 1 \mid \pi_k) = \pi_k$, as the probability of choosing component k in a GMM is just π_k and does not depend on the means or covariances.
- $p(\boldsymbol{x} \mid z_k = 1, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = p(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{u}_k, \boldsymbol{\Sigma}_k)$, since once component k is selected the probability of $\boldsymbol{x}$ only depends on the mean and covariance for mixture component k.
- $p(\boldsymbol{x} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_j \pi_j \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ since this is just the density for a single point. Substituting these gives the result.

[Your answer would not have to be quite this detailed. If you don't know Bayes theorem, it is reviewed in any introductory text on probabilities, and follows directly from the rule of conditional probability $p(x, y) = p(x \mid y)p(y)$.]

Q8. [12 marks total] This question is about *support vector machines*. Suppose you are given the following training set for 2-class classification:

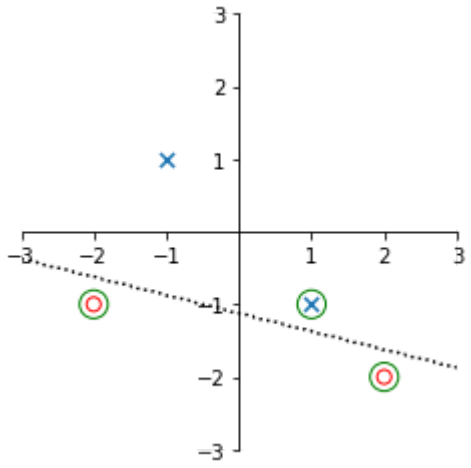
$$\boldsymbol{X} = \begin{bmatrix} -2 & -1 \\ 2 & -2 \\ 1 & -1 \\ -1 & 1 \end{bmatrix}, \quad \boldsymbol{t} = \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$$

a) [5 marks] For the above training set, write the quadratic program for the primal hard-margin SVM learning problem. You should not have symbols $\boldsymbol{x}$ or t in your answer: instead substitute the coefficients ($2w_1$, etc).

For this training set, the primal hard-margin SVM learning problem can be formulated as the following quadratic program:

$$\begin{aligned} \min_{\boldsymbol{w},b} \ & \frac{1}{2} \|\boldsymbol{w}\|^2 \\ \text{s.t.} \ & 1 \leq \ 2w_1 + \ w_2 - b \\ & 1 \leq -2w_1 + 2w_2 - b \\ & 1 \leq \ w_1 - \ w_2 + b \\ & 1 \leq -w_1 + \ w_2 + b \end{aligned}$$

b) [3 marks] Plot the data on a set of axes. Plot the optimal decision boundary (as closely as you can). Indicate which points are the support vectors.



c) [2 marks] The dual formulation of an SVM is defined in terms of kernels $k(\boldsymbol{x}, \boldsymbol{x}')$. Given kernel $k(\boldsymbol{x}, \boldsymbol{x}') = 1 + e^{-x_1 - x_1'} + e^{-x_2 - x_2'} + \cdots + e^{-x_D - x_D'}$, what feature space does this kernel correspond to? Explain.

That kernel corresponds to feature space

$$\boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} 1 & e^{-x_1} & e^{-x_2} & \cdots & e^{-x_D} \end{bmatrix}$$

because then taking $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\phi}(\boldsymbol{x})$ gives the kernel shown.

d) [2 marks] The dual formulation of hard-margin SVM is defined in terms of dual variables $\boldsymbol{\alpha} \in \mathbb{R}^N_{\geq 0}$. Write the formula for the SVM decision function $y(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}) + b$ in terms of the α_i, training labels t_i, training samples $\boldsymbol{x}_i$, kernel function $k(\cdot, \cdot)$, and the bias term b which you can assume is known. Show your steps.

The SVM decision function is $y(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}) + b$. As we derived in class, for any choice of dual variables $\boldsymbol{\alpha}$ we can write the optimal choice of primal weights as

$$\boldsymbol{w} = \sum_{i \in \mathcal{S}} \alpha_i t_i \boldsymbol{\phi}(\boldsymbol{x}_i)$$

where $\mathcal{S}$ is the set of support vectors (since $\alpha_i = 0$ for all $i \notin \mathcal{S}$). Substituting, we then have

$$\begin{aligned} y(\boldsymbol{x}) &= \left(\sum_{i \in \mathcal{S}} \alpha_i t_i \boldsymbol{\phi}(\boldsymbol{x}_i) \right)^T \boldsymbol{\phi}(\boldsymbol{x}) + b \\ &= b + \sum_{i \in \mathcal{S}} \alpha_i t_i \boldsymbol{\phi}(\boldsymbol{x}_i)^T \boldsymbol{\phi}(\boldsymbol{x}) \\ &= b + \sum_{i \in \mathcal{S}} \alpha_i t_i k(\boldsymbol{x}_i, \boldsymbol{x}) \end{aligned}$$

which is our final expression for the decision function.

Q9. [6 marks total] This question is about *boosting*.

a) [2 marks] What makes boosting an "ensemble" method?

Boosting is an "ensemble" method because it trains multiple models on the training set and then makes a prediction by combining the predictions of each model.

b) [2 marks] When the AdaBoost algorithm is training $y_r(\boldsymbol{x})$, the r^{th} weak learner, does it compute the current ensemble's predictions $y(\boldsymbol{x}_i)$ over the training set, where $y(\boldsymbol{x}) = \sum_{j=1}^{r-1} \alpha_j y_j(\boldsymbol{x})$? Explain.

No. Unlike gradient boosting, AdaBoost does not explicitly compute the training predictions $y(\boldsymbol{x}_i)$ of the current ensemble. Instead it updates the sample weights w_i in such a way that when $y_r(\boldsymbol{x})$ is trained it will likely "boost" performance.

c) [2 marks] In the AdaBoost from lecture, if the weak learners are always "better than random guessing" then each sample weight w_i will either stay the same or will increase. *True or False?* Explain.

False. If there are any misclassified training cases, then their weight will increase, but the normalization step will then *decrease* the weights of all correctly-classified training samples.