# EECS545 Machine Learning
# WN22 Midterm

April 7, 2022

## Kindly don't share or post these solutions.

You have 2 hours to complete this Midterm Exam. You may use your book and notes to complete the exam. However, the use of any electronic devices, including phones and laptops are strictly prohibited. You are also not allowed to discuss any portion of the exam with any other student while the exam is being proctored. The total points of the midterm is 100. Please manage your time to receive as many points as possible. Note that Section 3 (Short answer questions) may take longer than the first 2 sections.

There are a total of **29 questions** (Pg. 2 - 15). Additional pages have been provided for scratch work (Pg. 16 - 18). Please let us know if any pages are missing from your exam.

| Sections | Pages | Questions | Points |
|---|---|---|---|
| 1. Multiple choice and T/F without explanations | 2 - 5 | 1 - 14 | /42 |
| 2. Multiple choice and T/F with explanations | 6 - 8 | 15 - 22 | /16 |
| 3. Short answer questions | 9 - 15 | 23 - 29 | /42 |
| Total | 2 - 15 | 29 | /100 |

Name: _____

Uniqname: _____

I attest that I have not given or received aid in this examination, and that I have done my share and taken an active part in seeing to it that others as well as myself uphold the spirit and letter of the University of Michigan College of Engineering Honor Code. Specifically for this examination, I attest that I have not used a laptop, a smartphone, any electronic device, or any online materials.

Signed: _____

## Multiple choice with no explanations

**Write the options only [e.g. (a), (d)] in the box provided below each question.**

1. **[4 points]** Assume that you are given data matrix $\mathbf{X} \in \mathbb{R}^{N \times M}$ where each row contains a data point $\left(\mathbf{x}^{(i)}\right)^T$. Assume that the data is centered so that $\frac{1}{N}\sum_{i=1}^{N} \mathbf{x}^{(i)} = \mathbf{0}$ and the covariance matrix $\frac{1}{N}\mathbf{X}^T\mathbf{X}$ has distinct eigenvalues. Which of the following are true? **Choose all options that apply.**

   (a) If you use PCA to project $d$-dimensional points down to $j$ principal coordinates, and then you run PCA again to project those $j$-dimensional coordinates down to $k$ principal coordinates, with $d > j > k$, you always get the same result as if you had just used PCA to project the $d$-dimensional points directly down to $k$ principle coordinates.

   (b) If you rotate each of the sample points in feature space ($\mathbf{x}^{(i)} \to U\mathbf{x}^{(i)}$, $U$ is an orthogonal matrix) before performing PCA, the principal component directions and the associated eigenvalues do not change.

   (c) If you rotate each of the sample points in feature space ($\mathbf{x}^{(i)} \to U\mathbf{x}^{(i)}$, $U$ is an orthogonal matrix) before performing PCA, the principal component directions change but the associated eigenvalues remain the same.

   (d) If you perform an invertible linear transform of the sample points in feature space ($\mathbf{x}^{(i)} \to A\mathbf{x}^{(i)}$, $A$ is an invertible matrix) before performing PCA, the principal component directions do not change.

   **Solution:** (a), (c).
   (a): True because all eigenvalues are distinct. (c): Rotating all points rotates the principal components by the same amount. Mathematically, $X^{new} = XU^T$ and new covariance matrix is $\frac{1}{N}U\mathbf{X}^T\mathbf{X}U^T = UV\Lambda V^T U^T = (UV)\Lambda(UV)^T$

2. **[2 points]** Assume that a binary classifier estimates the probability $p(y = 1|\mathbf{x}) = \sigma(\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c)$ where $\mathbf{A}$, $\mathbf{b}$ and $c$ are parameters of the model and $\sigma$ is the sigmoid function. Assume $\mathbf{A}$ is restricted to be a **positive semi-definite matrix**. The decision boundary of the classifier (**choose one option only**)

   (a) Is always linear

   (b) Can be linear or non-linear

   (c) Is always non-linear

   **Solution:** (b). Is linear when $\mathbf{A} = \mathbf{0}$

3. **[4 points]** Which of the following ideas can be used for better convergence while training a neural network? **Choose all options that apply.**

   (a) Keep a running momentum that is updated at each step by the current gradient, and use this momentum for updating parameters.

   (b) Use a decaying learning rate instead of a fixed learning rate.

   (c) Add a scalar factor $\lambda$ to the loss function with $0 < \lambda < 1$ to ensure numerical stability.

   (d) Use optimizers like Adagrad or RMSProp which utilize adaptive learning rates

**Solution:** (a), (b), (d)
(c) is false because adding a scalar factor does not affect the gradient of the loss function - which means that the convergence is not affected

4. **[4 points]** Assume that your model is overfitting the training data, then which of the following methods can you use to avoid that? **Choose all options that apply.**

   (a) Increase the complexity of the existing model.

   (b) Increase the quantity of training data.

   (c) Introduce some form of regularization.

**Solution:** (b), (c)
(a) is false because increasing complexity increases the chances of overfitting.

5. **[4 points]** Which of the following factors contribute to difficulty in finding the optimal weights for a deep neural network model? **Choose all options that apply.**

   (a) The output of the network is typically a highly nonlinear function of the input.

   (b) Typical loss functions (like mean squared error or cross entropy) are non-convex with respect to weights of the network.

   (c) It's impossible to obtain the gradient of loss with respect to to the weights of the network.

   (d) SGD can get stuck in the many local minima and saddle points present in the loss landscape.

**Solution:** (a), (b), (d)
(c) is false because we can calculate the gradient of loss wrt weights; that's how we perform back-propagation.

6. **[2 points]** Which of the following about the closed-form solution to linear regression with least square loss is true? **Choose one option only.**

   (a) Computing the closed-form solution is computationally more expensive compared to gradient descent on very large and very high-dimensional datasets.

   (b) The closed-form solution may not give the exact optimum.

   (c) With L2 regularization, a unique closed-form solution may not exist.

   (d) The closed-form solution will always be faster to compute than the iterative solution.

**Solution:** (a): Closed form requires matrix multiplication and inversion which is $\approx O(n^3)$ if matrix size is $n \times n$. Iterative solution is $O(n^2)$ per step.

7. **[2 points]** Suppose for a dataset with $n$ examples we run classification using K-nearest neighbors, with $K = n$. In this case, K-nearest neighbors is... **(choose only one answer.)**

   (a) Very good – error is not more than twice the optimal error

   (b) Reasonable – $n$ is not a bad choice for $k$

   (c) Somewhat poor – $K = n$ leads to somewhat high bias

   (d) Very poor – Leads to a naive classifier

**Solution:** (d)
We have a majority classifier when $K = n$, which is a naive classifier.

8. **[2 points]** Which of the following is a disadvantage of Newton's method compared to gradient descent? **Choose only one answer.**

   (a) Newton's method usually takes more iterations than gradient descent to converge

   (b) Newton's method takes more time to compute on an individual iteration than gradient descent

   (c) Newton's method does not always find the optimum for a convex function

**Solution:** (b)
Need to perform matrix inversion (which is expensive) in Newton's method but not in gradient descent.

9. **[2 points]** What is the purpose of using cross validation instead of just a training/test split (without a cross-validation partition)? **Choose only one answer.**

   (a) Avoid overfitting hyperparameters to test set.

   (b) Utilize more data in the training phase.

   (c) Avoid using too much data in the test phase.

**Solution:** (a)

10. **[2 points]** What is one benefit of using $K$-fold cross validation over ordinary (1-fold) cross validation? **Choose only one answer.**

    (a) Regularizes the objective and reduces overfitting.

    (b) Accuracy increases to 100% as $K$ increases to the size of the dataset.

    (c) Helps with choosing better / best hyperparameters out of multiple possible sets of hyperparameters.

**Solution:** (c)

11. **[4 points]** Consider polynomial regression by optimizing the least-squares objective function with regularization:

$$\tilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(\mathbf{w}^T\phi(x^{(i)}) - y^{(i)})^2 + \lambda\|\mathbf{w}\|^2$$

where $x^{(i)}$ is scalar and $\phi(x^{(i)})_j = (x^{(i)})^j$.
**Choose all options that apply**:

   (a) Increasing $\lambda$ increases estimator bias and decreases estimator variance.

   (b) As $\lambda$ increases, the curve formed by polynomial regression will become flatter.

   (c) Best practice indicates that we should keep tuning $\lambda$ until we find a final value $\lambda^*$ that minimizes error on the test set.

   (d) It is not valid to use the $L1$ norm in place of the $L2$ norm in the regularized objective function.

**Solution:** (a), (b)

12. **[3 points] Choose all conditions** that guarantee the convergence of the perceptron algorithm:

    (a) When we kernelize the perceptron algorithm with Gaussian (RBF) kernel.

    (b) When every data point with positive label cannot be represented as the convex combination of the data points with negative labels.

    (c) When one feature has the same sign with the label, where the label takes value -1 or +1.

**Solution:** (a), (c)
We are looking for the sufficient conditions for linear separability. For (a), RBF kernel corresponds to the infinite dimensional features, where finite dataset is always separable. For (c), using that one feature with threshold at 0, we can separate dataset.
(b) is not sufficient. For example, positively labeled points uniformly distributed in the ring of radius 1 cannot be represented by the convex combination of negative labeled points lie inside the circle of radius 1. But the dataset is not linearly separable. To make it linearly separable, we would need symmetric condition with positive and negative labeles interchanged.

13. **[3 points]** After training a linear SVM classifier, you observe that the test error is high while the training error is low.

    (a) **[1 points]** True or false: Changing the kernel from linear to quadratic is likely to help improve test performance.

    **Solution:** False, Overfitting.

    (b) **[1 points]** True or false: Increasing the value of parameter $C$ is likely to help improve test performance.

    **Solution:** False, Decreasing $C$ will improve margin and that might help.

    (c) **[1 points]** True or false: Performing feature selection, i.e., selecting fewer number of informative features might help improve test performance.

    **Solution:** True. Reduce overfitting chance.

14. **[4 points] Choose all true statements**:

    (a) When the regularization parameter C increases in soft-margin SVM, the number of support vectors will increase.

    (b) When the regularization parameter C increases in soft-margin SVM, the total sum of slack variables will decrease.

    (c) For a fixed linearly-separable dataset, the minimum objective value for soft-margin SVM is at least as small as the minimum objective value for hard-margin SVM.

    (d) If $\mathbf{w}_1$ and $\mathbf{w}_2$ have the same objective value for soft-margin SVM, their average $\bar{\mathbf{w}} = (\mathbf{w}_1 + \mathbf{w}_2)/2$ has the smaller objective value compared to $\mathbf{w}_1$ and $\mathbf{w}_2$.

**Solution:** (b), (c), (d)
(a), (b): Increasing $C$ would penalize more for the slack variables, decreasing the total sum of slack variables at optimum. The number of support vectors is not directly correlated.
(c): Since the solution for the hard-margin SVM (0 slack variables) is still a feasible point for the soft-margin SVM problem, having more freedom on slack variable can only decrease the minimum for the soft-margin SVM.
(d): The soft-margin SVM is a convex optimization problem with strictly convex objective. Thus, the average of two distinct points would always be feasible and decrease the objective.

## Multiple choice with explanations

**Clearly state your choice and then follow it up with a 1 - 2 sentence(s) explanation for each question.**

15. **[2 points]** True or false: Logistic regression learns a non-linear decision boundary because the logistic function is non-linear.

**Solution:** False. Logistic regression learns a linear decision boundary in the feature space. In other words, when you use $x$ as features, then the decision boundary is a linear function of $x$. If you use non-linear function $\phi(x)$ as features, the decision boundary can be non-linear function of $x$, but this is not because the logistic function is non-linear.

16. **[2 points]** Training with more training data usually improves generalization. Suppose we have a training set containing $N$ training cases. If we duplicate the training cases $L$ times each so that the training set now contains $LN$ cases (but still only $N$ unique cases), will this improve the generalization performance of a kernelized linear regression (in any case, assume that the regularization (e.g., $L2$) hyperparameter will be carefully tuned on the validation data)?
Note that this is similar to a practice exam question, except we are using *kernelized* linear regression instead of logistic regression.

**Solution:** No. Naively duplicating data still doesn't help since the objective function will be increased by a constant factor. So, the optimal solution would remain the same as in the case of using original data.

17. **[2 points]** Consider a problem where you want to use a high-dimensional features (where there may be some correlation between the features). Between logistic regression and naive Bayes, which is a better choice?

**Solution:** Logistic regression. The naive Bayes assumes conditional independence of features given class labels. This may be a too strong assumption when there is non-trivial correlation between features.

18. **[2 points]** For any two documents $x$ and $z$, define $K(x, z)$ to equal the number of unique words that occur in both $x$ and $z$ (i.e., the size of intersection of the sets of words in the two documents $x$ and $z$. Is this function a kernel? If so, please show how to construct the corresponding feature vector $\phi(x)$ from a document $x$. If not, please justify your answer.

**Solution:** Yes. $\phi(x)$ as a binary vector whose $i$-th entry is 1 when the document x contains the $i$-th word and 0 if it doesn't.

19. **[2 points]** True or false:

The VC dimension of a class of parameterized non linear functions is equal to the number of parameters it contains.

**Solution:** False
There are many counterexamples. VC dim can be lower or higher than the number of parameters in the model.
Multi layer neural networks ($VC \geq WL\log(W/L)$)
SVM with RBF kernel can have a VC Dimension of $\infty$ (or something else, it is dependent on number of support vectors).
For any $t \in \mathbb{R}$, define a function $f_t(x) = \text{sign}(\sin(tx))$ and let $\mathcal{H} = \{f_t | t \in \mathbb{R}\}$. You can show that this class can shatter any dataset of any size (choose $t$ high enough so that we sufficiently frequent oscillations). This class has $\infty$ VC-dimension with only one parameter.
VC dim of this overparameterized family is 1:

$$\begin{aligned}\mathcal{H} &= \{f(x) = \text{sign}\left(b\,\text{sign}(ax^2) + c\right) | a, b, c \in \mathbb{R}\} \\ &= \{f(x) = \text{sign}\left(b\,\text{sign}(a) + c\right) | a, b, c \in \mathbb{R}\} \\ &= \{f(x) = \text{sign}\left(c\right) | c \in \mathbb{R}\}\end{aligned}$$

VC dim of this overparameterized family is 2:

$$\begin{aligned}\mathcal{H} &= \{f(x) = \text{sign}\left(c\,\text{sign}(ax + b)\right) | a, b, c \in \mathbb{R}\} \\ &= \{f(x) = \text{sign}\left(ax + b\right) | a, b \in \mathbb{R}\}\end{aligned}$$

Note that for linear classifiers $\mathcal{H} = \{f(x) = \text{sign}\left(\mathbf{w}^T\mathbf{x} + b\right) | \mathbf{w} \in \mathbb{R}^d, b \in R\}$ the number of parameters and the VC dimension are both $d + 1$.

20. **[2 points]** True or false: Naive Bayes classifier and GDA (Gaussian Discriminant Analysis) are generative models. Why?

**Solution:** True, can estimate joint density $p(x, y)$

21. **[2 points]** True or false: $k(\mathbf{x}, \mathbf{y}) = k_1(\mathbf{x}, \mathbf{y}) - k_2(\mathbf{x}, \mathbf{y})$ is a kernel if $k_1$ and $k_2$ are defined as $k_1(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y} + 1)^2$ and $k_2(x, y) = \mathbf{x}^T\mathbf{y}$.

**Solution:** True. $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y})^2 + \mathbf{x}^T\mathbf{y} + 1$ with each term in the sum being a kernel.

22. **[2 points]** True or false: When training a neural network, it is better to initialize weights to zeros for better convergence.

**Solution:** False: The idea is that there is no symmetry breaking. Concretely, if all nodes have the same value of 0, then all the nodes will have the same gradient vector composed of 0's, which will not allow us to update the weights at any learning rate.

## Short Answer questions

**Write your answer in the space provided below each question.**

23. **[2 points]** Recall Laplacian smoothing adds one "imaginary count" to each word, so that

$$P(word = j|spam) = \mu_j^s = \frac{N^{spam}}{\sum_j N_j^{spam}}$$

$$P(word = j|not\ spam) = \mu_j^{ns} = \frac{N^{nonspam}}{\sum_j N_j^{nonspam}}$$

becomes

$$P(word = j|spam) = \mu_j^s = \frac{N^{spam} + 1}{\sum_j N_j^{spam} + M}$$

$$P(word = j|not\ spam) = \mu_j^{ns} = \frac{N^{nonspam} + 1}{\sum_j N_j^{nonspam} + M}$$

for $M$ being the size of the word vocabulary. In 1-2 sentences, explain why this is done.

**Solution:** Avoids zero product for words that show up as only spam / only nonspam

24. Consider a binary classification CNN that classifies $224 \times 224$ RGB images into ten categories. The CNN has the following layers that are arranged sequentially:

- `Conv2D (outChannels=16, inChannels=3, kernel=(5, 5), stride=1, padding=2)`
- `ReLU()`
- `MaxPool2D (kernel=(2, 2), stride=2)`
- `Conv2D (outChannels=64, inChannels=16, kernel=(3, 3), stride=1, padding=0)`
- `ReLU()`
- `MaxPool2D (kernel=(2, 2), stride=2)`
- `Flatten()`
- `Linear (input_dim=?, output_dim=?)`
- `Softmax()`

Note that the network will be trained with a cross-entropy loss.

(a) **[6 points]** What would be the shape of the weight matrix for the final linear layer (before the softmax)?

Output shape after the first convolution: _____.

Output shape after the second convolution: _____.

Shape of the weight matrix in the linear layer: _____.

**Solution:** Output shape after the first convolution: $(16, 224, 224)$.
Output shape after the second convolution: $(64, 110, 110)$.
Shape of the weight matrix in the linear layer: $(64 * 55 * 55) \times 10$.

(b) **[4 points]** Consider now that we directly pass a $256 \times 256$ RGB image to this network. Does the network still produce a valid output without an error? If yes, please state why the network can handle images of different sizes. If no, please state the layer where the network fails and what you can do to ensure that the network works for variable input sizes. [Hint: You may want to think about adding operations before or after the network as well]

**Solution:** The network fails at the first fully-connected layer due to a dimension or shape mismatch. In order to ensure that the network works with different sized images, you can add a pre-processing step in the pipeline that resizes inputs to $224 \times 224$. This could be full-image resizing or any relevant cropping of the image. Alternatively, you can also add a Global Average Pooling layer to the network to ensure that images of any size can work with the network.

25. **[2 points]** One form of the Naïve Bayes classifier can be considered to be a special case of the Gaussian Discriminant Analysis. What should be the form of the covariance matrix for the GDA to consider it as a case of Naïve Bayes?

**Solution:** A GDA classifier can be considered to be a form of the Naïve Bayes classifier if the Naïve Bayes condition holds:
$P(x_1, ..., x_M|C_k) = \prod_{j=1}^{M} P(x_j|C_k)$
2 This can happen when the off-diagonal elements of the covariance matrix are zero. Therefore, the covariance matrix can be any **diagonal matrix**.

26. **[2 points]** Give a condition on the prior $p(\theta)$ which ensures that the MAP estimate $\arg\max_\theta p(\theta|\mathcal{D})$ and the ML estimate $\arg\max_\theta p(\mathcal{D}|\theta)$ are the same for all datasets $\mathcal{D}$ and likelihood $p(\mathcal{D}|\theta)$.

**Solution:** A sufficient condition for ML and MAP estimates to be same ($\arg\max_\theta p(\theta)p(D|\theta) = \arg\max_\theta p(D|\theta)$) is when $p(\theta)$ be independent of $\theta$ i.e, $p(\theta) = c \ \forall\theta$. Note that $c$ is not necessarily 1. Keyword: Uniform prior or $p(\theta) = c$ or $p(\theta_1) = p(\theta_2) = \cdots$ ... **[2 points]**
$p(\theta) = 1 \ldots$ **[1 points]**
Otherwise (e.g.: $p(\theta) = p(\mathcal{D})$) **[0 points]** (Note that the question asks to find a condition which would hold for all likelihood functions).

27. Assume we have data $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1\ldots N}$ where $\mathbf{x}^{(i)} \in \mathbb{R}^d$ and $y^{(i)} \in \{-1, 1\}$. Note that the labels are now $\{-1, 1\}$ which is different from the convention followed in lecture. In this case, the (regularized) logistic regression objective function that is minimized can be written as

$$L(\mathbf{w}, b) = \frac{1}{N} \sum_{i=1}^{N} \left( \phi \left( y^{(i)} (\mathbf{w}^T \mathbf{x}^{(i)} + b) \right) \right) + \lambda \|\mathbf{w}\|^2$$

where $\phi(t) = \log(1 + \exp(-t))$. Let $\mathbf{X} \in \mathbb{R}^{N \times d}$ so that each row of $\mathbf{X}$ is $(\mathbf{x}^{(i)})^T$.

(a) **[4 points]** Calculate the gradient of the objective with respect to the weights $\mathbf{w}$.

$$\nabla_{\mathbf{w}} L(\mathbf{w}, b) = \_\_\_\_\_$$

We can use the above expression to write the gradient descent update rule for $\mathbf{w}$ (with learning rate $\eta$) as follows:

$$\mathbf{w}_{t+1} = \mathbf{w}_t (1 - 2\lambda\eta) - \frac{\eta}{N} \mathbf{X}^T \mathbf{g}$$

where $\mathbf{g} \in \mathbb{R}^N$. Write the expression for the $\mathbf{g}_i$, the $i^{th}$ element of $\mathbf{g}$.

$$\mathbf{g}_i = \_\_\_\_\_$$

**Note:** You are not expected to simplify $\phi'(\cdot)$.

**Solution:** $\nabla_{\mathbf{w}} L(\mathbf{w}, b) = \frac{1}{N} \sum_{i=1}^{N} \left( y^{(i)} \phi' \left( y^{(i)} (\mathbf{w}^T \mathbf{x}^{(i)} + b) \right) \mathbf{x}^{(i)} \right) + 2\lambda \mathbf{w} \ldots$ **[2 points]**

$$\begin{aligned} \mathbf{w}_{t+1} &= \mathbf{w}_t - \eta \nabla_{\mathbf{w}} L(\mathbf{w}_t, b_t) \\ &= \mathbf{w}_t (1 - 2\lambda\eta) - \frac{\eta}{N} \sum_{i=1}^{N} \left( y^{(i)} \phi'(y^{(i)} (\mathbf{w}_t^T \mathbf{x}^{(i)} + b_t)) \mathbf{x}^{(i)} \right) \\ &= \mathbf{w}_t (1 - 2\lambda\eta) - \frac{\eta}{N} \mathbf{X}^T \mathbf{g} \end{aligned}$$

where $\mathbf{g}_i = y^{(i)} \phi'(y^{(i)} (\mathbf{w}_t^T \mathbf{x}_i + b_t)) \ldots$ **[2 points]**.

(b) **[4 points]** We can show that if $\mathbf{w}_t = \mathbf{X}^T \boldsymbol{\alpha}_t$ for some $\boldsymbol{\alpha}_t \in \mathbb{R}^N$ then $\mathbf{w}_{t+1}$ obtained by the update rule in part (a) can also be expressed as $\mathbf{w}_{t+1} = \mathbf{X}^T \boldsymbol{\alpha}_{t+1}$ for some $\boldsymbol{\alpha}_{t+1} \in \mathbb{R}^d$. Write the expression for $\boldsymbol{\alpha}_{t+1}$ in terms of $\boldsymbol{\alpha}_t$, $\mathbf{g}$ and other constants. Note that $\mathbf{g}_i$ (the $i^{th}$ element of $\mathbf{g}$) has to be written only in terms of $\boldsymbol{\alpha}_t$, $\mathbf{X}$, $\mathbf{x}^{(i)}$ and $y^{(i)}$ and $b_t$.

**Solution:**

$$\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t (1 - 2\lambda\eta) - \frac{\eta}{N} \mathbf{g} \ldots \textbf{[4 points]}$$

where

$$\mathbf{g}_i = y^{(i)} \phi' \left( y^{(i)} (\boldsymbol{\alpha}_t^T \mathbf{X} \mathbf{x}^{(i)} + b_t) \right) \ldots \textbf{[Extra points]}$$

and thus

$$\mathbf{g} = \left[ y^{(1)} \phi' \left( y^{(1)} (\boldsymbol{\alpha}_t^T \mathbf{X} \mathbf{x}^{(1)} + b_t) \right), \ldots, y^{(n)} \phi' \left( y^{(n)} (\boldsymbol{\alpha}_t^T \mathbf{X} \mathbf{x}^{(n)} + b_t) \right) \right]^T$$

28. Consider a probabilistic model with random variables $z \in \{0, 1\}$ and $x \in \{0, 1, 2, \ldots, M\}$. The joint distribution of the two variables is given by the following:

$$\begin{aligned} z &\sim \text{Bernoulli}(\phi) \\ x|z=1 &\sim \text{Binomial}(M, p) \\ x|z=0 &\sim \text{Binomial}(M, t) \end{aligned}$$

The parameters of this model are $\phi, p, t \in [0, 1]$. Suppose $z$ is a latent (unobserved) random variable. Our training set is therefore of the form $\{x^{(1)}, \ldots, x^{(N)}\}$. Derive an EM algorithm to find the optimal parameters of the model.

Recall that if $X \sim \text{Binomial}(N, p)$, $P(X = k) = \binom{N}{k} p^k (1-p)^{N-k}$ where $\binom{N}{k} = \frac{N!}{(N-k)!k!}$ is the binomial coefficient.

(a) **[2 points]** The E-step involves computing the following quantity for all $n \in \{1, 2, \ldots, N\}$:

$$q^{(n)} = p(z^{(n)} = 1|x^{(n)}) = \_\_\_\_\_$$

Fill in the blank. The expression should be in terms of the parameters of the model, the data and possibly other constants.

**Solution:**

$$q^{(n)} = p(z^{(n)} = 1|x^{(n)}) = \frac{\phi \binom{M}{x^{(n)}} p^{x^{(n)}} (1-p)^{M-x^{(n)}}}{\phi \binom{M}{x^{(n)}} p^{x^{(n)}} (1-p)^{M-x^{(n)}} + (1-\phi) \binom{M}{x^{(n)}} (t)^{x^{(n)}} (1-t)^{M-x^{(n)}}}$$

(Note that we can cancel out the binomial coefficient) ... **[2 points]**

(b) **[4 points]** The M-step involves the following optimization:

$$\arg\max_{p,t,\phi} \sum_{n=1}^{N} \left( q^{(n)} \left( \_\_\_\_\_ \right) + (1 - q^{(n)}) \left( \_\_\_\_\_ \right) \right)$$

Fill in the blanks. The final expression should be in terms of the parameters of the model, the training data and possibly other constants.

**Solution:**

$$\arg\max_{p,t,\phi} \sum_{n=1}^{N} q^{(n)} \log p(x^{(n)}, z^{(n)} = 1) + (1 - q^{(n)}) \log p(x^{(n)}, z^{(n)} = 0) \ldots \textbf{[2 points]}$$

$$= \arg\max_{p,t,\phi} \sum_{n=1}^{N} q^{(n)} \left( \log \phi + \log \binom{M}{x^{(n)}} + x^{(n)} \log p + (M - x^{(n)}) \log(1-p) \right)$$
$$+ (1 - q^{(n)}) \left( \log(1-\phi) + \log \binom{M}{x^{(n)}} + x^{(n)} \log t + (M - x^{(n)}) \log(1-t) \right) \ldots \textbf{[2 points]}$$

(Note that we can get rid of the binomial coefficients)
Not necessary to write the first equation, if the second equation is right full 4 points will be awarded. Also, not necessary to write the logarithm of products as a sum of logarithms in the second step.
It was asked to write the final expression in terms of parameters, so vague expressions (e.g. in terms of Binomial$(x^{(n)}; M, p)$ will not get full credit.

(c) **[4 points]** M-step: Using the expression derived in part (b), write the update rule for $p$.

**Solution:**

$$\sum_{n=1}^{N} \frac{q^{(n)}x^{(n)}}{p} - \sum_{n=1}^{N} \frac{q^{(n)}(M - x^{(n)})}{1-p} = 0 \ldots \textbf{[2 points]}$$

$$\implies \sum_{n=1}^{N} q^{(n)}x^{(n)} - p\sum_{n=1}^{N} q^{(n)}x^{(n)} = pM\sum_{n=1}^{N} q^{(n)} - p\sum_{n=1}^{N} q^{(n)}x^{(n)}$$

$$\implies p = \frac{\sum_n q^{(n)}x^{(n)}}{M\sum_n q^{(n)}} \ldots \textbf{[2 points]}$$

29. Consider a simplified scalar RNN model: $x_t \in [-B, B] \ \ \forall t \in \{1, 2, \cdots, T\}$, $h_0 = 0$ and

$$h_t = ax_t + bh_{t-1} \quad \forall t \in \{1, 2, \cdots, T\}.$$

Here, the scalars $a$ and $b$ form the parameters of the RNN.

Assume we train the RNN with a squared loss $L(y, \hat{y}) = \frac{1}{2}(\hat{y} - y)^2$ where $\hat{y} = h_T$ (the last hidden state). Note that a single loss is applied at the end of the sequential process.

(a) **[4 points]** Calculate the gradient $\dfrac{\partial L}{\partial a}$.

Note that the first step involved in calculating this gradient would be

$$\frac{\partial L}{\partial a} = \frac{\partial L}{\partial h_T}\frac{\partial h_T}{\partial a} = (y - h_T)\frac{\partial h_T}{\partial a}$$

by the chain rule. Can you expand the second term in the expression?

$$\begin{aligned}\frac{\partial L}{\partial a} &= \frac{\partial L}{\partial h_T}\frac{\partial h_T}{\partial a} \\ &= (y - h_T)\frac{\partial h_T}{\partial a} \\ &= \underline{\qquad\qquad\qquad\qquad\qquad\qquad}\end{aligned}$$

**Solution:**

$$\frac{\partial L}{\partial a} = \frac{\partial L}{\partial h_T}\frac{\partial h_T}{\partial a} = (y - h_T)\frac{\partial h_T}{\partial a}$$

$$= (y - h_T)\left(x_T + b\frac{\partial h_{T-1}}{\partial a}\right) \ldots \textbf{[1 points]}$$

$$= (y - h_T)\left(x_T + bx_{T-1} + b^2\frac{\partial h_{T-2}}{\partial a}\right) \textbf{[1 points]}$$

$$= \cdots$$

$$= (y - h_T)\left(x_T + bx_{T-1} + b^2x_{T-2} + \cdots + b^kx_{T-k} + \cdots b^{T-1}x_1\right) \ldots \textbf{[2 points]}$$

All other cases ... **[0 points]**

(b) **[2 points]** Observe the expression of the gradient $\frac{\partial L}{\partial a}$ and write a condition which can cause exploding gradient.

**Solution:** $|b| > 1$. Note that the $x_t$'s are bounded ($[-B, B]$).
Acceptable: High values of $b$ Not acceptable: Conditions on $x_t$s or $h_t$s.

(c) **[2 points]** Observe the expression of the gradient $\frac{\partial L}{\partial a}$ and write a condition which can cause vanishing gradient.

**Solution:** $|b| < 1$, Acceptable: Low values of $b$.
Not acceptable: Conditions on $x_t$s or $h_t$s.