CS689: Machine Learning - Fall 2021

Final Exam

Dec 15, 2021 Name: _____

Instructions: Write your name on the exam sheet. No electronic devices may be used during the exam. You may consult your paper notes and/or a print copy of texts during the exam. Sharing of notes/texts during the exam is strictly prohibited. Show your work for all derivation questions. A suitable explanation must be provided for all questions that ask for one to earn full credit. Attempt all problems. Partial credit may be given for partially incorrect or incomplete answers. If your answer to a question spans two pages, please make sure to indicate that at the end of the first page. If you have questions at any time, please raise your hand.

Problem	Topic	Page	Points	Score
1	Probabilistic Regression	1-2	10	
2	Probabilistic Logistic Regression	3-4	10	
3	Count Regression	5-6	10	
4	Exponential Mixtures	7-8	10	
5	Prediction with Mixtures	9-10	10	
6	Semi-Supervised Factor Analysis	11-12	10	
7	Non-Linear Factor Analysis	13-14	10	
8	De-Noising Autoencoders	15-16	10	
9	Implementation	17-18	10	
10	Modeling	19-20	10	
Total:			100	

1. (*10 points*) **Probabilistic Regression** Consider the generalized probabilistic regression model shown below:

$$p(Y = y|X = x, \theta) = \mathcal{N}(y; w_0 + w_1 x, \exp(v_0 + v_1 x)^2)$$

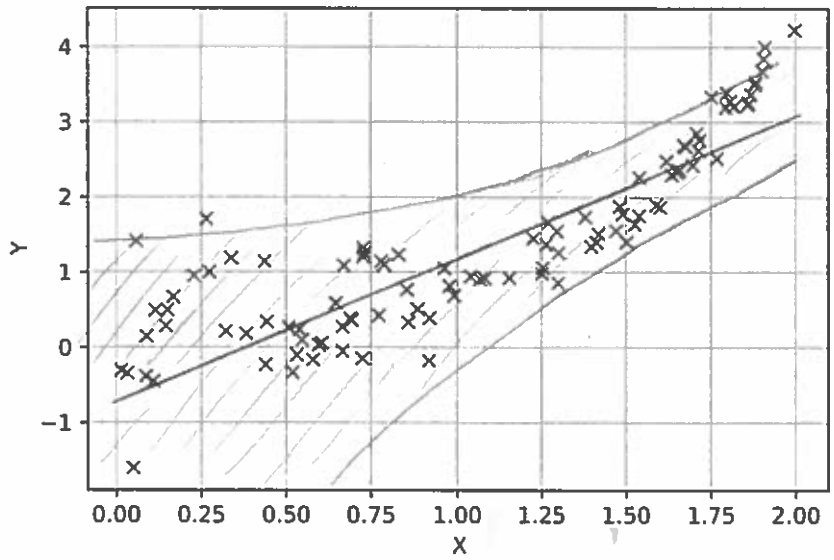
In this model $x \in \mathbb{R}$, $y \in \mathbb{R}$, $w_0 \in \mathbb{R}$, $w_1 \in \mathbb{R}$, $v_0 \in \mathbb{R}$, $v_1 \in \mathbb{R}$. The model parameters are $\theta = [w_0, w_1, v_0, v_1]$. Recall that the notation $\mathcal{N}(z, \mu, \sigma^2)$ indicates the normal probability density function with mean μ and standard deviation σ evaluated at $z \in \mathbb{R}$.

(a) (5 points) What is $\mathbb{E}_{p(Y|X=x,\theta)}[Y]$ for this model? Explain your answer.

Since $P(Y=y|X=x,\Theta)$ is a normal distribution conditioned on X, the expected value $\mathbb{E}_{P(Y|X=x,\Theta)}[Y]$ is just the mean of this distribution, which is

$w_0 + w_1 X$.

(b) (5 points) Suppose we train the model from Part (a) on the data set shown below by minimizing the negative log conditional likelihood. Sketch a ribbon plot of the fit of the trained model to the data set. Show the mean of y given x as well as the two-standard deviation region around the mean. Explain your answer.



Since the mean of $p(Y=y|X=x,\Theta)$ is a linear function of x, $w_0 + w_1 x$, the conditional mean line must be a straight line.

Since the standard deviation of $p(Y=y|X=x,\Theta)$ is an exponential of a linear function of x, the standard deviation will not be the same for all x. It can be larger for small x where the data have more spread, and smaller for large x where the data have less spread.

2. (*10 points*) **Probabilistic Logistic Regression** Consider the probabilistic logistic regression model shown below. For this model $y \in \{0, 1\}$, $\mathbf{x} \in \mathbb{R}^D$, and the parameters are $\mathbf{w} \in \mathbb{R}^D$. Assume that $\mathbf{x}$ is a row vector, $\mathbf{w}$ is a column vector, and bias absorption has been applied.

$$P(Y = y|\mathbf{X} = \mathbf{x}, \mathbf{w}) = \left(\frac{1}{1 + \exp(-\mathbf{xw})}\right)^y \left(1 - \frac{1}{1 + \exp(-\mathbf{xw})}\right)^{(1-y)}$$

(a) (5 points) Prove that $P(Y = y|\mathbf{X} = \mathbf{x}, \mathbf{w})$ is a valid probability mass function for any value of $\mathbf{x} \in \mathbb{R}^D$ and any value of $\mathbf{w} \in \mathbb{R}^D$. Assume that all data and parameter values are finite.

We need to show normalization and non-negativity.

Normalization:

$$P(Y=0|X=x,w) + P(Y=1|X=x,w)$$

$$= \frac{1}{1+\exp(-xw)} + \left(1 - \frac{1}{1+\exp(-xw)}\right)$$

$$= 1 + \frac{1}{1+\exp(-xw)} - \frac{1}{1+\exp(-xw)} = 1$$

Non-Negativity:

$P(Y=1|x,w) = \frac{1}{1+\exp(-xw)}$, which must be ≥ 0 since $\exp(z) \geq 0$ for all $z \in \mathbb{R}$.

$P(Y=0|x,w) = 1 - \frac{1}{1+\exp(-xw)} = \frac{\exp(-xw)}{1+\exp(-xw)}$, which is also ≥ 0 due to $\exp(z) \geq 0$ for all $z \in \mathbb{R}$

(b) (5 points) The decision boundary for the probabilistic logistic regression model is a surface in $\mathbb{R}^D$ such that all of the points on one side of the surface belong to class 0 and all the points on the other side of the surface belong to class 1. Starting from the definition of $P(Y = y|\mathbf{X} = \mathbf{x}, \mathbf{w})$, derive an expression that is as simple as possible for the set of points that fall on this decision surface. Explain your answer.

In a probabilistic classifier, the predicted class is the one with the highest probability. The decision boundary will occur at points where $P(Y=0|x,w) = P(Y=1|x,w)$.

This happens when $P(Y=0|x,w) = P(Y=1|x,w) = 0.5$.

We thus need to find the set of points $\{x \mid P(Y=1|x,w)=0.5\}$.

We have $P(Y=1|x,w) = \frac{1}{1+\exp(-xw)} = 0.5$

$\Rightarrow 1 = 0.5\,(1+\exp(-xw))$

$\Rightarrow 0.5 = 0.5\exp(-xw)$

$\Rightarrow 1 = \exp(-xw)$

$\Rightarrow \log(1) = \log(\exp(-xw))$

$\Rightarrow 0 = -xw$

$\Rightarrow xw = 0 \Rightarrow$ The decision boundary is $\{x \mid xw=0\}$.

3. (*10 points*) **Count Regression** Suppose we have a data set $\mathcal{D} = \{(\mathbf{x}_n, y_n) | 1 \leq n \leq N\}$ where $\mathbf{x}_n \in \mathbb{R}^D$ and $y_n \in \{1, 2, 3, ...\}$. Use this information to answer the following questions.

(a) (5 points) Give an expression for a probabilistic supervised model $P(Y = y|\mathbf{X} = \mathbf{x}, \theta)$ for these data based on the geometric distribution and explain your answer. Recall that for $z \in \{1, 2, 3, ...\}$, the probability mass function for the geometric distribution is $P(Z = z|\phi) = (1 - \phi)^{(z-1)}\phi$. The parameter constraints for this distribution are $0 < \phi \leq 1$.

We need to relate x to the parameters of the geometric distribution.

The easiest way to do this is to use a transformation of a linear prediction function.

Let $w \in \mathbb{R}^D$ and $b \in \mathbb{R}$. We let $f(x) = xw + b$.

The parameters are positive, so we select $\sigma()$ as the transformation. We have $\varphi(x) = \sigma(f(x)) = \sigma(xw + b)$.

The model is $P(Y = y | X = x, \theta) = (1 - \varphi(x))^{(y-1)} \cdot \varphi(x)$

$$= \underline{\left(1 - \sigma(xw + b)\right)^{(y-1)} \left(\sigma(xw + b)\right)}$$

with $\theta = [w, b]$.

(b) (5 points) Explain how you would learn the model you described in Part (a).

To learn the model we would use negative log likelihood minimization. The optimization problem is:

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmin}} - \sum_{n=1}^{N} \log P(Y = y_n \mid X = x_n, \Theta)$$

$$= \underset{\Theta}{\operatorname{argmin}} - \sum_{n=1}^{N} \left((y-1) \log \left(1 - \exp(xw)\right) + (xw+b) \right)$$

$$\log\left(\frac{1}{1+\exp^{-z}}\right)$$

$$\log\left(-\log\left(1+\exp^{-z}\right)\right)$$

0

4. (*10 points*) **Exponential Mixtures** Suppose we have a data set $\mathcal{D} = \{\mathbf{x}_n | 1 \leq n \leq N\}$ where $\mathbf{x}_n = [x_{n1}, ..., x_{nD}]$ is a vector of length D. Suppose that the data are non-negative so that $x_{nd} \in \mathbb{R}^{\geq 0}$ for all n and d. Suppose we decide to model these data using mixture components based on the exponential distribution. Recall that the exponential probability density function is given by $p(Y = y|\lambda) = \lambda \exp(-\lambda y)$ for $y \in \mathbb{R}^{\geq 0}$ and $\lambda \in \mathbb{R}^{>0}$.

Provide an expression for the joint distribution $P(Z = z, \mathbf{X} = \mathbf{x}|\theta)$ for such a model. Explain your answer.

A mixture has the form $P(Z=z)P(X=x \mid Z=z)$.

$P(Z=z)$ is discrete such that $P(Z=z) = \pi_z$ with $\sum_{z=1}^{K} \pi_z = 1$ and $\pi_z \geq 0 \ \forall z$. We assume $z \in \{1, ..., K\}$

We will use a mixture of marginals and write

$$P(X=x \mid Z=z) = \prod_{d=1}^{D} P(X_d = x_d \mid Z=z)$$

We let $P(X_d = x_d \mid Z=z) = \lambda_{dz} \exp(-\lambda_{dz} x_d)$ with $\lambda_{dz} \geq 0$.

The full model's joint distribution is:

$$P(Z=z)P(X=x \mid Z=z) = \pi_z \cdot \prod_{d=1}^{D} \lambda_{dz} \exp(-\lambda_{dz} x_d)$$

5. (*10 points*) **Prediction with Mixtures** Suppose we have a data set $\mathcal{D} = \{\mathbf{x}_n | 1 \leq n \leq N\}$. Assume each data vector is two-dimensional such that $\mathbf{x}_n = [x_{n1}, x_{n2}]$. We decide to model these data using a mixture with component distributions based on the univariate normal density. Recall that the univariate normal probability density function with mean $\mu \in \mathbb{R}$ and standard deviation $\sigma \in \mathbb{R}^{>0}$ evaluated at $y \in \mathbb{R}$ is given by:

$$\mathcal{N}(y; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{1}{2\sigma^2}(y-\mu)^2)$$

Derive an expression for the mean of $p(X_1 = x_1 | X_2 = x_2)$ under this mixture model. Explain your answer.

We have $P(x_1|x_2) = \frac{P(x_1, x_2)}{P(x_2)} = \frac{\sum_z P(x_1, x_2, z)}{\sum_{z'} \int P(x_1', x_2, z') dx_1'}$

For the mixture model, we have $P(x_1, x_2, z) = P(x_1|z)P(x_2|z)P(z)$.

We have $P(x_1|z) = \mathcal{N}(x_1; \mu_{1z}, \sigma_{1z}^2)$ and $P(x_2|z) = \mathcal{N}(x_2; \mu_{2z}, \sigma_{2z}^2)$.

We have $P(z) = \pi_z$.

$$\int P(x_1', x_2, z') dx_1' = \int P(x_1'|z')P(x_2|z')P(z') dx_1'$$
$$= P(x_2|z')P(z') \int P(x_1'|z') dx_1'$$
$$= P(x_2|z')P(z') \quad \text{(by normalization of } P(x_1'|z'))$$

We thus have: $P(x_1|x_2) = \frac{\sum_z \mathcal{N}(x_1; \mu_{1z}, \sigma_{1z}^2)\mathcal{N}(x_2; \mu_{2z}, \sigma_{2z}^2)\pi_z}{\sum_{z'} \mathcal{N}(x_2; \mu_{2z'}, \sigma_{2z'}^2)\pi_{z'}}$

$$= \sum_z N(x_1; \mu_{1z}, \sigma_{1z}^2) \cdot \frac{\pi_z N(x_2; \mu_{2z}, \sigma_{2z}^2)}{\sum_{z'} \pi_{z'} N(x_2; \mu_{2z'}, \sigma_{2z'}^2)}$$

By linearity of expectation, we have

$$\int x_1 \cdot \sum_z N(x_1; \mu_{1z}, \sigma_{1z}^2) \cdot \frac{\pi_z N(x_2; \mu_{2z}, \sigma_{2z}^2)}{\sum_{z'} \pi_{z'} N(x_2; \mu_{2z'}, \sigma_{2z'}^2)} \, dx_1$$

$$= \sum_z \left(\int x_1 N(x_1; \mu_{1z}, \sigma_{1z}^2) \, dx_1 \right) \cdot \frac{\pi_z N(x_2; \mu_{2z}, \sigma_{2z}^2)}{\sum_{z'} \pi_{z'} N(x_2; \mu_{2z'}, \sigma_{2z'}^2)}$$

$$= \sum_z \mu_{1z} \cdot \frac{\pi_z N(x_2; \mu_{2z}, \sigma_{2z}^2)}{\sum_{z'} \pi_{z'} N(x_2; \mu_{2z'}, \sigma_{2z'}^2)}$$

6. (*10 points*) **Semi-Supervised Factor Analysis** Consider the classical factor analysis model where $p(\mathbf{Z} = \mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I)$ and $p(\mathbf{X} = \mathbf{x}|\mathbf{Z} = \mathbf{z}, \theta) = \mathcal{N}(\mathbf{x}; \mathbf{W}\mathbf{z}, \Psi)$. Assume $\mathbf{z} \in \mathbb{R}^K$, $\mathbf{x} \in \mathbb{R}^D$, $\mathbf{W}$ is a $D \times K$ matrix and Ψ is a $D \times D$ diagonal covariance matrix. The parameters are $\theta = [\mathbf{W}, \Psi]$.

Suppose we have access to two data sets: $\mathcal{D}_u = \{\mathbf{x}_n | 1 \leq n \leq N_u\}$ and $\mathcal{D}_s = \{(\mathbf{x}_n, \mathbf{z}_n) | 1 \leq n \leq N_s\}$. $\mathcal{D}_u$ consists of observations of the data vectors $\mathbf{x}$ only, while $\mathcal{D}_s$ consists of pairs of data vectors $\mathbf{x}$ and their corresponding codes $\mathbf{z}$. Give an equation for an objective function that we could use to learn this model while leveraging all of the data in both of the data sets. Explain your answer.

For the unsupervised data set, we can use the nlml objective: $nlml(D_u, \theta) = -\sum_{x_n \in D_u} \log N(x_n; 0, WW^T + \Psi)$

For the supervised data set, we can use the nll of the joint distribution. However $p(Z=z)$ has no parameters, so we only need the $p(X=x|Z=z, \Theta)$ term: $nll(D_s, \Theta) = -\sum_{x_n, z_n \in D_s} \log N(x_n; Wz_n, \Psi)$

For the complete objective function, we can combine these two terms:

$$obj(D_u, D_s, \Theta) = nlml(D_u, \theta) + nll(D_s, \Theta)$$

7. (*10 points*) **Non-Linear Factor Analysis** In the non-linear factor analysis model we have $p(\mathbf{Z} = \mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I)$ and $p(\mathbf{X} = \mathbf{x}|\mathbf{Z} = \mathbf{z}, \theta) = \mathcal{N}(\mathbf{x}; f_{\mathbf{w}}(\mathbf{z}), \Psi)$ where $\mathbf{w}$ are the parameters of a neural network model $f_{\mathbf{w}}(\mathbf{z})$ and Ψ is a diagonal covariance matrix. The variational autoencoder attempts to deal with the presence of intractable integrals in the negative log marginal likelihood of the non-linear factor analysis model using several approximations. Describe an alternative, computationally tractable optimization-based approach to learning the parameters for this model. Include formulas for the optimization objective function and the learning problem along with your explanation.

We can use the idea of optimizing both the model parameters and the latent variables.

our objective function would be:

$$J(D, \Theta, z_{1:N}) = -\sum_{n=1}^{N} \log P(X = x_n, Z = z_n | \Theta)$$

$$= -\sum_{n=1}^{N} \log P(X = x_n | Z = z_n, \Theta) P(Z = z_n)$$

$$= -\sum_{n=1}^{N} \left(\log N(x_n; f_w(z_n), \Psi) + \log N(z_n; 0, I) \right)$$

The learning problem is:

$$\min_{\Theta} \min_{z_{1:N}} J(D, \Theta, z_{1:N})$$

8. (*10 points*) **De-Noising Autoencoders** In a basic non-linear de-noising autoencoder we start with a data set $\mathcal{D} = \{\mathbf{x}_n | 1 \leq n \leq N\}$. Each data case $\mathbf{x}_n = [\mathbf{x}_{n1}, ..., \mathbf{x}_{nD}]$ is a D-dimensional real vector. On each learning iteration for each data case n, we form an input-output pair $(\mathbf{x}'_n, \mathbf{x}_n)$ by sampling a normally distributed noise variable $\eta_{nd} \sim N(0, \sigma^2)$ for each dimension and adding the noise variables to the original data cases to create the noisy copies: $\mathbf{x}'_n = [\mathbf{x}_{n1} + \eta_{n1}, ..., \mathbf{x}_{nD} + \eta_{nD}]$.

An important question when learning using this approach is how to set the noise standard deviation σ. Describe a learning experiment to solve this problem. You may make any additional assumptions you think are necessary for the problem to be well defined. You should explain your answer and justify any additional assumptions you make.

We can't set the σ parameter using a validation set experiment using the denoising reconstruction loss itself since the optimal value will be $\sigma = 0$.

To create a sensible experiment, we need a way to evaluate the model trained using σ that does not depend on σ.

One option when using the codes as input to a downstream supervised task is to select the value of σ that minimizes the loss in the supervised task. That selection can be made using a train-validation-test experiment for the supervised task that fits the autoencoder as part of training and evaluates prediction performance on the validation set for multiple values of σ.

9. (*10 points*) **Implementation** A student is tasked with implementing the probabilistic linear regression model $p(Y = y|\mathbf{X} = \mathbf{x}, \theta) = \mathcal{N}(y; \mathbf{x}\mathbf{w} + b, \sigma^2)$ in PyTorch. For this model we have $\mathbf{x} \in \mathbb{R}^D$, $y \in \mathbb{R}$, $\mathbf{w} \in \mathbb{R}^D$, $b \in \mathbb{R}$ and $\sigma \in \mathbb{R}^{>0}$. The inputs to the learning algorithm are a tensor of features values X of shape $[N, D]$ and a tensor of target values Y of shape $[N, 1]$. To learn the model, the exp() function is used as the standard deviation parameter transformation.

The student produces the implementation below that runs for one iteration and then crashes. The code contains several implementation errors. As your answer to this question, identify four errors and briefly explain how to fix them.

```
import torch
class lr:
    def __init__(self,D):
        self.w = torch.zeros(D,1, requires_grad=True)
        self.b = torch.tensor([0.0] , requires_grad=True)
        self.tau = torch.tensor([0.0] , requires_grad=True)
        self.sigma = torch.exp(self.tau)

    def nll(self,X,Y):
        v    = torch.pow(self.sigma,2)
        nll  = torch.sum(-0.5*torch.log(2*torch.pi*v) \
               -0.5/v*torch.pow(Y-(torch.matmul(X, self.w)+self.b ),2))
        return nll

    def fit(self,X,Y,epochs):
        opt = torch.optim.Adam([self.w, self.b],0.1)
        for i in range(epochs):
            loss = self.nll(X,Y)
            loss.backward()
            opt.step()
```

(1) (2) (3) (4) ⇒ zero grad!

1) The nll function needs to recompute sigma from the unconstrained parameter tau. It currently uses the value of sigma set during initialization.

2) The sign of the nll term is wrong. The final output needs to be negated.

3) The optimizer needs to take self.tau as an input in the call to Adam. Currently, only w and b are being optimized.

4) The optimization loop is missing a step to clear the gradient information on each step. We need a call to opt.zero_grad().

10. (*10 points*) **Modeling** To be robust, deployed machine learning classification systems need a mechanism to deal with out-of-distribution examples (examples that do not match the training distribution). One possibility is to allow a classification system to reject (not issue a classification) for examples that are deemed to be out-of-distribution. Using models that we have explored this semester, describe a classification system implementing classification with rejection. Describe what model or models your system would use, how the models would be learned, and how they would be used to decide what inputs to reject and what class values to assign when not rejecting.

(1) We could use a mixture model to model $p(X=x)$. Assuming that all of our training data are in-distribution, once the model is learned (using NLML) we could evaluate it on all of the training data and compute the minimum probability of any data case $\tau = \operatorname{argmin}_{x \in D} p(X=x)$.

(2) For the classifier we can select any model. We propose a probabilistic neural network classifier $p(Y=y \mid X=x)$ trained to minimize NLL.

(3) At deployment time, given a new input x, we compute $p(X=x)$ using the mixture model. If $p(X=x) < \tau$, we reject the instance. If $p(X=x) \geq \tau$, we output $\hat{y} = \operatorname{argmax}_y p(Y=y \mid X=x)$.