STANFORD UNIVERSITY
CS 229, Spring 2019
Midterm Examination Solutions

Question	Points
1 Short Answers	/25
2 Newton's Method	/15
3 Exponential - Supervised and Unsupervised	/30
4 Kernelized GLM	/20
5 Kernel Fun	/10
Total	/100

Name of Student: _____

SUNetID: _____@stanford.edu

The Stanford University Honor Code:

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 Honor Code.

Signed: _____

1. **[25 points] Short answers**

   (a) [10 points] True or False

   i. Maximum Likelihood objective of any Exponential Family distribution is concave in its natural parameters.
   Answer: TRUE

   ii. When using Kernel methods, to make the prediction on a new example x at test time, we still need to access the training data.
   Answer: TRUE

   iii. Naive Bayes for text classification requires us to remember the training data at test time.
   Answer: FALSE

   iv. We know that the squared error is a convex loss. So when we train all the parameters of any Neural Network with squared error loss, we are effectively minimizing a convex function.
   Answer: FALSE

   v. Unsupervised Learning models, specifically those trained with the EM algorithm cannot overfit.
   Answer: FALSE

   (b) [5 points] **Not Overfitting?**

   Suppose we have n training examples $\{x^{(i)}, y^{(i)}\}_{i=1}^{n}$, $x^{(i)} \in \mathbb{R}^d, y^{(i)} \in \mathbb{R}$ where $d > n$. On this data set we train a linear regression model with a correctly implemented software package, and to our surprise find that we CANNOT achieve 0 training loss. What could be the reason?

   Hint: It is something about the data. You only need to give one possible reason to explain the scenario.

   Answer: One possible reason: there are two examples $\{x^{(i)}, y^{(i)}\}$ and $\{x^{(j)}, y^{(j)}\}$, with $i \neq j$ such that $x^{(i)} = x^{(j)}$ but $y^{(i)} \neq y^{(j)}$.

(c) [5 points] **Not Underfitting?**

Suppose we have n training examples $\{x^{(i)}, y^{(i)}\}_{i=1}^{n}$, $x^{(i)} \in \mathbb{R}^d, y \in \mathbb{R}$ where $n >> d$. On this data set we train a linear regression model with a correctly implemented software package, and to our surprise find that we CAN achieve 0 training loss. What could be the reason?

Hint: It is something about the data. You only need to give one possible reason to explain the scenario.

Answer:

One possible reason: there are lots of repeated examples, or linearly dependent examples such that the number of linearly independent examples (considering the full (x, y) pair) is less than d.

Another possible reason: y is a linear combination of the columns of X without any noise.

(d) [5 points] **Not fitting?**

Suppose we are trying to fit a model with gradient descent with correctly implemented gradient calculation, and observe that with each iteration the training loss is *increasing*. What could be the reason?

Hint: You only need to give one possible reason to explain the scenario.

Answer:

Most likely reason is that the step size / learning rate is too large for the problem.

2. **[15 points] Newton's Method**

   We introduced Newton's method in class by providing its update rule. In this question you will be asked to derive the update rule of Newton's methods from a given mathematical motivation, using the optimization tools learned from the class. As a corollary you will show that Newton's method can solve linear regression with a single iteration.

   Suppose $f(\theta)$ is a twice-differentiable convex function that we aim to optimize. Recall the Taylor series expansion of a function around some input point θ_0 is

   $$f(\theta) = f(\theta_0) + \nabla f(\theta_0)^T(\theta - \theta_0) + \frac{1}{2}(\theta - \theta_0)^T \nabla^2 f(\theta_0)(\theta - \theta_0) + o\left(||\theta - \theta_0||^3\right)$$

   The approach taken by Newton's method is to approximate the function in each iteration up to the quadratic term in the Taylor expansion, and minimize this approximation instead of the actual function. Concretely, the algorithm computes a sequence of variables $\theta^{(0)}, \ldots, \theta^{(t)}, \ldots$, where the variable obtained at iteration t is denoted by $\theta^{(t)}$. Let us denote the quadratic approximation near a point $\theta^{(t)}$ by $\tilde{f}_{\theta^{(t)}}$, as follows:

   $$\tilde{f}_{\theta^{(t)}}(\theta) := f(\theta^{(t)}) + \nabla f(\theta^{(t)})^T(\theta - \theta^{(t)}) + \frac{1}{2}(\theta - \theta^{(t)})^T \nabla^2 f(\theta^{(t)})(\theta - \theta^{(t)})$$

   The Newton's method can written as

   1. $\theta^{(0)} =$ random initialization
   2. For $t = 0, 1, 2, \ldots \{$

      $$\theta^{(t+1)} = \arg\min_{\theta} \tilde{f}_{\theta^{(t)}}(\theta) \tag{1}$$

   $\}$

   The following subquestions will use the above algorithm.

(a) [5 points] Show that the above algorithm is equivalent to the update rule we saw in class (which was $\theta^{(t+1)} = \theta^{(t)} - \left[\nabla^2 f(\theta^{(t)})\right]^{-1} \nabla f(\theta^{(t)})$). Here we assume that $\nabla^2 f(\theta^{(t)})$ is full rank for every t, and because the function f is assumed to be convex, you can use the fact that $\nabla^2 f(\theta^{(t)})$ is positive semidefinite, and that the approximation $\tilde{f}_{\theta^{(t)}}(\theta)$ is also convex. You are also allowed to assume that $\tilde{f}_{\theta^{(t)}}(\theta)$ has a unique minimizer.

Answer:

In each iteration t, we minimize $\tilde{f}_{\theta^{(t)}}(\theta)$ by setting its derivative to 0 and solving for θ.

$$
\begin{aligned}
0 &= \nabla_\theta \tilde{f}_{\theta^{(t)}}(\theta) \\
&= \nabla_\theta \left(f(\theta^{(t)}) + \nabla_\theta f(\theta^{(t)})^T (\theta - \theta^{(t)}) + \frac{1}{2}(\theta - \theta^{(t)})^T \nabla_\theta^2 f(\theta^{(t)})(\theta - \theta^{(t)}) \right) \\
&= \nabla f(\theta^{(t)}) + \nabla^2 f(\theta^{(t)})(\theta - \theta^{(t)}) \\
\Rightarrow \theta - \theta^{(t)} &= -\left(\nabla_\theta^2 f(\theta^{(t)})\right)^{-1} \nabla_\theta f(\theta^{(t)}) \\
\Rightarrow \theta^{(t+1)} &:= \theta^{(t)} - H^{-1} \nabla_\theta f(\theta^{(t)}) \quad \blacksquare
\end{aligned}
$$

(b) [5 points] Now recall the loss function of linear regression is

$$J(\theta) = ||X\theta - \vec{y}||_2^2.$$

Show that by starting with a random initialization, Newton's method gets us to the final solution in a single iteration.

Answer:

Solution 1 (Indirect):

The third derivative of $J(\theta)$ w.r.t θ is zero, and hence the approximation of Newton's method is exact. Hence Newton's method must solve linear regression in a single step.

Solution 2 (Direct):

The cost function, gradient and Hessian of linear regression at $\theta^{(0)}$ is:

$$\begin{aligned}
J(\theta) &= ||X\theta - \vec{y}||_2^2 \\
&= \theta^T(X^TX)\theta + \vec{y}^T\vec{y} - 2\theta^TX^T\vec{y} \\
\nabla J(\theta) &= 2(X^TX)\theta - 2X^T\vec{y} \\
\Rightarrow \nabla_\theta J(\theta^{(0)}) &= 2X^TX\theta^{(0)} - 2X^T\vec{y} \\
H(\theta) &= 2(X^TX) \\
\Rightarrow H(\theta^{(0)}) &= 2(X^TX)
\end{aligned}$$

So, starting at $\theta^{(0)}$ and applying one Newton's method update step takes us to $\theta^{(1)}$ as

$$\begin{aligned}
\theta^{(1)} &:= \theta^{(0)} - H^{-1}(\theta^{(0)})\nabla_\theta J(\theta^{(0)}) \\
&= \theta^{(0)} - \left(2(X^TX)\right)^{-1}\left(2(X^TX)\theta^{(0)} - 2X^T\vec{y}\right) \\
&= \theta^{(0)} - (X^TX)^{-1}(X^TX)\theta^{(0)} + (X^TX)^{-1}X^T\vec{y} \\
&= \theta^{(0)} - \theta^{(0)} + (X^TX)^{-1}X^T\vec{y} \\
&= (X^TX)^{-1}X^T\vec{y} \\
&= \hat{\theta}.
\end{aligned}$$

So we see that taking one Newton's step from any randomly initialized $\theta^{(0)}$ will always take us directly to the optimal solution.

(c) [5 points] In this previous two sub-questions we used Newton's method to *minimize* a *convex* cost function. In this sub-question, we will see that convexity is essential in the interpretation.

Show that applying Newton's method to $\tilde{J}(\theta) = -J(\theta)$ takes us to the same solution as applying it to $J(\theta)$. In other words, applying Newton's method to $\tilde{J}(\theta)$ will give us a maximizer of $\tilde{J}$, instead of a minimizer.

Remark 1: This suggest that if we know the function is either convex or concave, then Newton's method is "plug and play" in the sense that, the method automatically maximizes or minimizes it appropriately. By contrast, for gradient descent or stochastic gradient descent, we explicitly need to maximize or minimize by either adding or subtracting the gradient.

Remark 2: This also suggests that the performance of Newton's method on functions that are neither convex nor concave is unpredictable. It may converge to a local minimum, local maximum, or a saddle point. By contrast, the gradient descent or stochastic gradient descent algorithm always tend to minimize a function (and under certain assumptions, it can be proven to converge to a local minimum).

Answer:

Newton's method update rule starting at $\theta^{(0)}$ on $\tilde{J}(\theta)$ is:

$$\begin{aligned}\theta^{(1)} &= \theta^{(0)} - \left(\nabla^2 \tilde{J}(\theta)\right)^{-1} \nabla \tilde{J}(\theta^{(0)}) \\ &= \theta^{(0)} - \left(\nabla^2 - J(\theta)\right)^{-1} \nabla \left(-J(\theta^{(0)})\right) \\ &= \theta^{(0)} - \left(\nabla^2 J(\theta)\right)^{-1} \nabla J(\theta^{(0)})\end{aligned}$$

which is exactly the update rule for $J(\theta)$.

3. **[30 points] Exponential - Supervised and Unsupervised**

   In this question we will derive the methods to learn from data distributed according to the Exponential distribution. We will consider two cases - one with labeled data in the supervised learning setting (Exponential Discriminant Analysis) and the other with unlabeled data in the unsupervised learning setting (EM algorithm on a Mixture of Exponentials).

Supervised Setting (Exponential Discriminant Analysis)

We have seen the Gaussian Discriminant Analysis algorithm in the class, and we also heard in the lecture that the same algorithm can be easily extended to the Poisson distribution (i.e. modeling input x as Poisson in place of Gaussian). While the Gaussian distribution is suited to model real valued inputs, Poisson is appropriate for modeling count (positive integers) inputs. Similarly, the Exponential distribution is commonly used for modeling positive real values (for e.g. when the output of a model is the time until some future event, and hence always positive).

In the first part of this problem we will derive the Exponential Discriminant Analysis, a generative model for supervised learning.

Recall that the Exponential distribution parameterized by $\lambda > 0$ has density

$$p(x; \lambda) = \lambda \exp(-\lambda x), \qquad x \in \mathbb{R}_+.$$

Now suppose that our model is described as follows:

$$\begin{aligned} y &\sim \text{Bernoulli}(\phi) \\ x|y = 0 &\sim \text{Exponential}(\lambda_0) \\ x|y = 1 &\sim \text{Exponential}(\lambda_1) \end{aligned} \tag{2}$$

where ϕ is the parameter of the class marginal distribution, and λ_0 and λ_1 are the class specific parameters for the distribution over input x given $y \in \{0, 1\}$.

We will refer to the above model in the following subquestions.

(a) [5 points] Derive an exact formula for $p(y = 1|x)$ from the terms defined above, and also show that the resulting classifier has a linear decision boundary in x. Specifically, show that

$$p(y = 1|x) = \frac{1}{1 + \exp\{-(\theta_0 + \theta_1 x)\}}$$

for some θ_0 and θ_1. Clearly state what θ_0 and θ_1 are.

Answer:

$$\begin{aligned} p(y = 1|x) &= \frac{p(x|y = 1)p(y = 1)}{p(x|y = 0)p(y = 0) + p(x|y = 1)p(y = 1)} \\ &= \frac{\lambda_1 \exp\{-\lambda_1 x\}\,\phi}{\lambda_1 \exp\{-\lambda_1 x\}\,\phi + \lambda_0 \exp\{-\lambda_0 x\}\,(1-\phi)} \\ &= \frac{1}{1 + \frac{\lambda_0}{\lambda_1} \exp\{(\lambda_1 - \lambda_0)x\}\,\frac{1-\phi}{\phi}} \\ &= \frac{1}{1 + \exp\left\{-(\lambda_0 - \lambda_1)\,x - \log\frac{\lambda_1\phi}{\lambda_0(1-\phi)}\right\}} \end{aligned}$$

We see that $p(y = 1|x)$ has the form $1/(1+\exp(-\theta^T x - \theta_0))$ where $\theta_0 = \log\frac{\lambda_1\phi}{\lambda_0(1-\phi)}$ and $\theta = (\lambda_0 - \lambda_1)$, and hence has a decision boundary that is linear in x.

(b) [10 points] Derive the Maximum Likelihood Estimates of ϕ, λ_0 and λ_1 for the given training data using the joint probability (i.e $\ell(\phi, \lambda_0, \lambda_1) = \log \prod_{i=1}^{n} p(x^{(i)}, y^{(i)}; \phi, \lambda_0, \lambda_1)$).

Answer: First we write out the full log-likelihood

$$\begin{aligned}
\ell(\phi, \lambda_0, \lambda_1) &= \log \prod_{i=1}^{n} p(x^{(i)}, y^{(i)}; \phi, \lambda_0, \lambda_1) \\
&= \log \prod_{i=1}^{n} p(y^{(i)}; \phi) p(x^{(i)} | y^{(i)}; \lambda_{y^{(i)}}) \\
&= \sum_{i=1}^{n} \log p(y^{(i)}; \phi) + \log p(x^{(i)} | y^{(i)}; \lambda_{y^{(i)}}) \\
&= \sum_{i=1}^{n} y^{(i)} \log \phi + (1 - y^{(i)}) \log(1 - \phi) + \log \lambda_{y^{(i)}} - \lambda_{y^{(i)}} x^{(i)}
\end{aligned}$$

First we estimate ϕ by taking the gradient of ℓ w.r.t ϕ and setting it to 0.

$$\begin{aligned}
0 &= \nabla_\phi \ell(\phi, \lambda_0, \lambda_1) \\
&= \nabla_\phi \left[\sum_{i=1}^{n} y^{(i)} \log \phi + (1 - y^{(i)}) \log(1 - \phi) + \log \lambda_{y^{(i)}} - \lambda_{y^{(i)}} x^{(i)} \right] \\
&= \nabla_\phi \left[\sum_{i=1}^{n} y^{(i)} \log \phi + (1 - y^{(i)}) \log(1 - \phi) \right] \\
&= \sum_{i=1}^{n} \frac{y^{(i)}}{\phi} - \frac{1 - y^{(i)}}{1 - \phi} \\
&= \frac{1}{\phi(1 - \phi)} \sum_{i=1}^{n} \left(y^{(i)} - \phi \right) \\
\Rightarrow \hat{\phi}_{\text{MLE}} &= \boxed{\frac{1}{n} \sum_{i=1}^{n} y^{(i)}}
\end{aligned}$$

Next we estimate λ_1

$$\begin{aligned}
0 &= \nabla_{\lambda_1} \ell(\phi, \lambda_0, \lambda_1) \\
&= \nabla_{\lambda_1} \left[\sum_{i=1}^{n} y^{(i)} \log \phi + (1 - y^{(i)}) \log(1 - \phi) + \log \lambda_{y^{(i)}} - \lambda_{y^{(i)}} x^{(i)} \right] \\
&= \nabla_{\lambda_1} \left[\sum_{i=1}^{n} \log \lambda_{y^{(i)}} - \lambda_{y^{(i)}} x^{(i)} \right] \\
&= \nabla_{\lambda_1} \left[\sum_{i=1}^{n} \mathbf{1}\{y^{(i)} == 1\} \left(\log \lambda_1 - \lambda_1 x^{(i)} \right) \right] \\
&= \sum_{i=1}^{n} \mathbf{1}\{y^{(i)} == 1\} \left(\frac{1}{\lambda_1} - x^{(i)} \right) \\
\Rightarrow \hat{\lambda}_{1,\text{MLE}} &= \boxed{\frac{\sum_{i=1}^{n} \mathbf{1}\{y^{(i)} == 1\}}{\sum_{i=1}^{n} \mathbf{1}\{y^{(i)} == 1\} x^{(i)}}}
\end{aligned}$$

And by symmetry

$$\hat{\lambda}_{0,\text{MLE}} = \boxed{\frac{\sum_{i=1}^{n} \mathbf{1}\{y^{(i)} == 0\}}{\sum_{i=1}^{n} \mathbf{1}\{y^{(i)} == 0\} x^{(i)}}}$$

Unsupervised Setting (EM on Mixture of Exponential distributions)

Now consider the scenario where we only observe x from all the examples, and *do not observe* y. In this scenario we can apply the EM algorithm to infer the missing labels y and learn the parameters iteratively.

Concereteley, assume n examples $\{x^{(i)}, y^{(i)}\}_{i=1}^n$ are i.i.d generated from the process (2), but we only observe $\{x^{(i)}\}_{i=1}^n$. We will use EM algorithm to learn the parameters.

(c) [5 points] At the t-th iteration of the EM algorithm, suppose the current parameters $\phi^{(t)}$, $\lambda_0^{(t)}$ and $\lambda_1^{(t)}$. Derive the E-Step to infer the posterior distribution of the missing labels $y^{(i)}$'s given given the $x^{(i)}$'s and the current parameters. Specifically, write out the expression for

$$Q_i^{(t)} = p(y^{(i)} \mid x^{(i)}; \phi^{(t)}, \lambda_0^{(t)}, \lambda_1^{(t)}) = \dots$$

Hint: the steps would be similar in spirit to part (a). You are free to re-use parts of your solution in (a).

Answer: We calculate the posterior distribution at iteration t, $Q_i^{(t)}$ for each $i = 1, \dots, n$

$$\begin{aligned} Q_i^{(t)}(y^{(i)} = 1) &= p(y^{(i)} = 1 \mid x^{(i)}; \phi^{(t)}, \lambda_0^{(t)}, \lambda_1^{(t)}) \\ &= \frac{1}{1 + \exp\left\{ -\left(\lambda_0^{(t)} - \lambda_1^{(t)}\right) x^{(i)} - \log \frac{\lambda_1^{(t)} \phi^{(t)}}{\lambda_0^{(t)}(1-\phi^{(t)})} \right\}} \text{ (same steps as part-a)} \end{aligned}$$

(d) [10 points] Derive the M-Step to (re-)estimate the parameters $\phi^{(t+1)}$, $\lambda_0^{(t+1)}$, and $\lambda_1^{(t+1)}$. Specifically, write out and **highlight** the expressions for

$$\phi^{(t+1)} = \dots$$
$$\lambda_0^{(t+1)} = \dots$$
$$\lambda_1^{(t+1)} = \dots$$

Hint: the steps would be similar in spirit to part (b).

Answer:

$$\begin{aligned}
\phi^{(t+1)}, \lambda_0^{(t+1)}, \lambda_1^{(t+1)} &= \arg\max_{\phi,\lambda_0,\lambda_1} \sum_{i=1}^{n} \sum_{j\in\{0,1\}} Q_i^{(t)}(y^{(i)} = j) \log \frac{p(x^{(i)}, y^{(i)}; \phi, \lambda_0, \lambda_1)}{Q_i^{(t)}(y^{(i)})} \\
&= \arg\max_{\phi,\lambda_0,\lambda_1} \sum_{i=1}^{n} \sum_{j\in\{0,1\}} Q_i^{(t)}(y^{(i)} = j) \log p(x^{(i)}, y^{(i)}; \phi, \lambda_0, \lambda_1) \\
&= \arg\max_{\phi,\lambda_0,\lambda_1} \sum_{i=1}^{n} \sum_{j\in\{0,1\}} Q_i^{(t)}(y^{(i)} = j) \left(\log p(y^{(i)} = j; \phi) + \log p(x^{(i)} | y^{(i)}; \lambda_0, \lambda_1)\right) \\
&= \arg\max_{\phi,\lambda_0,\lambda_1} \sum_{i=1}^{n} \sum_{j\in\{0,1\}} Q_i^{(i)}(y^{(i)} = j) \left(j \log \phi + (1-j) \log(1-\phi) + \log \lambda_j - \lambda_j\right.
\end{aligned}$$

Let $w^{(i)} := Q_i^{(t)}(y^{(i)} = 1)$, and so $Q_i^{(t)}(y^{(i)} = 0) = 1 - w^{(i)}$.

Since all the parameters are separable, we can solve for each of them separately. First

we solve for ϕ by taking the gradient of the RHS w.r.t ϕ and setting it to 0.

$$\begin{aligned}
0 &= \nabla_\phi \left(\sum_{i=1}^{n} \sum_{j \in \{0,1\}} Q_i^{(t)}(y^{(i)} = j) \left(j \log \phi + (1-j) \log(1-\phi) + \log \lambda_j - \lambda_j x^{(i)} \right) \right) \\
&= \nabla_\phi \left(\sum_{i=1}^{n} \sum_{j \in \{0,1\}} Q_i^{(t)}(y^{(i)} = j) \left(j \log \phi + (1-j) \log(1-\phi) \right) \right) \\
&= \nabla_\phi \left(\sum_{i=1}^{n} w^{(i)} \log \phi + (1 - w^{(i)}) \log(1-\phi) \right) \\
&= \sum_{i=1}^{n} \frac{w^{(i)}}{\phi} - \frac{1 - w^{(i)}}{1-\phi} \\
&= \sum_{i=1}^{n} \frac{w^{(i)}(1-\phi) - \phi(1-w^{(i)})}{\phi(1-\phi)} \\
&= \sum_{i=1}^{n} \frac{w^{(i)} - \phi}{\phi(1-\phi)} \\
&= \frac{1}{\phi(1-\phi)} \left(-n\phi + \sum_{i=1}^{n} w^{(i)} \right) \\
\Rightarrow \phi^{(t+1)} &= \boxed{\frac{1}{n} \sum_{i=1}^{n} w^{(i)}}
\end{aligned}$$

Next we solve for λ_1:

$$\begin{aligned}
0 &= \nabla_{\lambda_1} \left(\sum_{i=1}^{n} \sum_{j \in \{0,1\}} Q_i^{(t)}(y^{(i)} = j) \left(j \log \phi + (1-j) \log(1-\phi) + \log \lambda_j - \lambda_j x^{(i)} \right) \right) \\
&= \nabla_{\lambda_1} \left(\sum_{i=1}^{n} Q_i^{(t)}(y^{(i)} = 1) \left(\log \lambda_1 - \lambda_1 x^{(i)} \right) \right) \\
&= \nabla_{\lambda_1} \left(\sum_{i=1}^{n} w^{(i)} \left(\log \lambda_1 - \lambda_1 x^{(i)} \right) \right) \\
&= \sum_{i=1}^{n} w^{(i)} \left(\frac{1}{\lambda_1} - x^{(i)} \right) \\
&= \frac{1}{\lambda_1} \left(\sum_{i=1}^{n} w^{(i)} \right) - \left(\sum_{i=1}^{n} w^{(i)} x^{(i)} \right) \\
\Rightarrow \frac{1}{\lambda_1} \left(\sum_{i=1}^{n} w^{(i)} \right) &= \left(\sum_{i=1}^{n} w^{(i)} x^{(i)} \right) \\
\Rightarrow \lambda_1^{(t+1)} &= \boxed{\frac{\sum_{i=1}^{n} w^{(i)}}{\sum_{i=1}^{n} w^{(i)} x^{(i)}}}
\end{aligned}$$

And by symmetry, for λ_0:

$$\lambda_0^{(t+1)} = \boxed{\frac{\sum_{i=1}^n 1 - w^{(i)}}{\sum_{i=1}^n (1 - w^{(i)}) x^{(i)}}}$$

To summarize, the M-Step is:

$$\begin{aligned}
\phi^{(t+1)} &= \frac{1}{n} \sum_{i=1}^n w^{(i)} \\
\lambda_0^{(t+1)} &= \frac{\sum_{i=1}^n 1 - w^{(i)}}{\sum_{i=1}^n (1 - w^{(i)}) x^{(i)}} \\
\lambda_1^{(t+1)} &= \frac{\sum_{i=1}^n w^{(i)}}{\sum_{i=1}^n w^{(i)} x^{(i)}}
\end{aligned}$$

where $w^{(i)} = p(y^{(i)} | x^{(i)}; \phi^{(t)}, \lambda_0^{(t)}, \lambda_1^{(t)})$.

4. **[20 points] Kernelized GLM**

We have seen how to kernelize Linear Regression via Gradient Descent in class. We have also studied Exponential Families and Generalized Linear Models, and seen how Linear Regression is a special case of GLM where the exponential family is chosen to be the Gaussian distribution (with fixed variance).

Now we will derive a gradient descent based algorithm to kernelize GLMs in general. Recall that an Exponential Family distribution is one which has the following form for its density/pmf:

$$p(y;\eta) = b(y)\exp\left\{\eta y - a(\eta)\right\},$$

where η is the natural parameter (assumed to be scalar in this problem), the sufficient statistic is y itself (which in general would be $T(y)$), $b(y)$ is the base measure, and $a(\eta)$ is the log-partition function.

In GLMs, we assume $\eta = \theta^T x$, where $\theta, x \in \mathbb{R}^d$. That is, we assume

$$p(y|x;\theta) = b(y)\exp\left\{\theta^T x \cdot y - a(\theta^T x)\right\}.$$

We are also given a training set $S = \{x^{(i)}, y^{(i)}\}_{i=1}^n$.

(a) [5 points] **Warm-up: non-kernel case**

First let us consider the case where we ignore feature maps and kernels. State explicitly what the hypothesis function $h_\theta(x)$ will be, using only the terms defined above (recall that $h_\theta(x)$ should be the expectation of $y|x$). Then, derive the gradient ascent update rule (not stochastic) for $\theta \in \mathbb{R}^d$ by maximizing the conditional log-likelihood (i.e $\ell(\theta) = \sum_{i=1}^n \log p(y^{(i)}|x^{(i)};\theta)$) of the raw training data S (i.e., no feature maps or kernels). Assume a learning rate $\alpha > 0$ and zero initialization $\theta^{(0)} = 0$.

Answer: Hypothesis function is:

$$\begin{aligned} h_\theta(x) &= \mathbb{E}[y|x;\theta] \\ &= a'(\theta^T x) \end{aligned}$$

Update rule:

$$\theta^{(t+1)} := \theta^{(t)} + \alpha\left(\sum_{i=1}^n (y^{(i)} - a'(\theta^T x^{(i)}))x^{(i)}\right)$$

(b) [10 points] **Kernelize GLM**

We now want to use some feature map $\phi(x) \in \mathbb{R}^p$, and wish to do so using the corresponding kernel representation $K(x^{(i)}, x^{(j)})$ since $\phi(x)$ could potentially be infinite dimensional (i.e. $p = \infty$).

Let $\beta \in \mathbb{R}^n$ be the set coefficients corresponding to the n examples with which we wish to represent $\theta \in \mathbb{R}^p$.

Derive the update rule for β based on the gradient descent algorithm that implicitly uses $\phi(x)$ as the features for training the GLM. Remember that in your final form of the update rule, there should be NO $\phi(x)$ terms (but should have $K(\cdot, \cdot)$ terms instead). Use learning rate $\alpha > 0$ and the state the hypothesis only using the terms defined so far.

Hint: You may find it easy to imitate the process we used for Kernelizing linear regression. First show that the parameter θ should always stay as a linear combination of $\phi(x^{(i)})$, and denote the coefficient of the combination as β. Then eliminate the occurrence of θ and derive the update for β.

Answer:

- Set $\beta^{(0)} = 0$.
- Iterate for $t = 0, 1, \ldots$ until convergence:

$$\forall i \in \{1, \ldots, n\} \quad \beta_i^{(t+1)} := \beta_i^{(t)} + \alpha \left(y^{(i)} - a' \left(\sum_{j=1}^{n} \beta_j^{(t)} K(x^{(i)}, x^{(j)}) \right) \right)$$

(c) [5 points] **Prediction with Kernelized GLM**

Suppose we train the kernelized GLM model successfully using the algorithm from part b until it converges. Next we deploy the model and encounter a new input x, and need to make a prediction using $\phi(x)$ (implicitly). You may assume that all your n training examples are available during test time. What is the prediction function over the new input x? Remember, as before your answer should NOT have any $\phi(x)$ terms, but have $K(\cdot, \cdot)$ terms instead.

Hint: The prediction function must output $\mathbb{E}[y|x; \hat{\beta}]$.

Answer: We predict using the learnt hypothesis $h(x)$

$$\begin{aligned} h_\theta(x) &= \mathbb{E}[y|x;\theta] \\ &= a'(\theta^T \phi(x)) \\ &= a'\left(\sum_{i=1}^{n} \beta_i K(x^{(i)}, x)\right) \blacksquare \end{aligned}$$

5. **[10 points] Kernel Fun**

In the following sub-questions, we will explore various properties of Kernels. Throughout the question, we assume $x, z \in \mathbb{R}^d$, $\phi : \mathbb{R}^d \to \mathbb{R}^p$, $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$.

(a) [5 points]

Suppose we have a Positive Semidefinite Matrix $G \in \mathbb{R}^{d \times d}$, and define a function K as follows:

$$K(x, z) := x^T G z.$$

Show that K is a valid kernel.

Remark: Note that G is *not* to be confused to be the kernel matrix.

Hint: You could consider using eigendecomposition of G, though it is possible to show the result without constructing an explicit feature map.

Answer:

Solution 1. Consider a set of n examples $\{x^{(1)}, \ldots, x^{(n)}\}$ where $x^{(i)} \in \mathbb{R}^d$. The corresponding kernel matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ is defined by $\mathbf{K}_{ij} = K(x^{(i)}, x^{(j)}) = {x^{(i)}}^T G x^{(j)}$. Showing that $\mathbf{K}$ is PSD and symmetric suffices to prove that K is a valid kernel. Consider any arbitrary $z \in \mathbb{R}^n$. We have

$$\begin{aligned}
z^T \mathbf{K} z &= \sum_{ij}^{n} z_i z_j \mathbf{K}_{ij} \\
&= \sum_{i,j}^{n} z_i z_j {x^{(i)}}^T G x^{(j)} \\
&= \sum_{i}^{n} \sum_{j}^{n} (z_i x^{(i)})^T G (z_j x^{(j)}) \\
&= \sum_{i}^{n} (z_i x^{(i)})^T G (\sum_{j}^{n} z_j x^{(j)}) \\
&= (\sum_{i}^{n} z_i x^{(i)})^T G (\sum_{j}^{n} z_j x^{(j)}) \\
&= y^T G y \\
&\geq 0
\end{aligned}$$

where $y = \sum_i^n z_i x^{(i)}$. Since z was arbitrarily chosen, $\mathbf{K}$ is PSD.

We also see that K is symmetric if G is symmetric:

$$\begin{aligned}
K_{ij} &= {x^{(i)}}^T G x^{(j)} \\
&= ({x^{(i)}}^T G x^{(j)})^T \\
&= {x^{(j)}}^T G^T x^{(i)} \\
&= K_{ji}
\end{aligned}$$

Therefore by Mercer's Theorem K is a valid kernel.

Solution 2 (explicit construction of feature map ϕ):

Let $G = Q\Lambda Q^T$ be the eigendecomposition of G where Q is unitary and Λ is diagonal matrix with non-negative entries (since it is PSD).

$$\begin{aligned}
K(x,z) &= x^T G z \\
&= x^T Q \Lambda Q^T z \\
&= x^T Q \Lambda^{1/2} \Lambda^{1/2} Q^T z \\
&= x^T Q \Lambda^{1/2} (\Lambda^{1/2})^T Q^T z \\
&= x^T Q \Lambda^{1/2} (Q\Lambda^{1/2})^T z \\
&= \left[(Q\Lambda^{1/2})^T x\right]^T \left[(Q\Lambda^{1/2})^T z\right] \\
&= \phi(x)^T \phi(z) \\
\text{where } \phi(x) &= \boxed{(Q\Lambda^{1/2})^T x}
\end{aligned}$$

Kernels or Not?

Is the following two sub-questions we explore whether a function K defined as $K(x, z) := x^T G z$ for different matrix values of G a valid kernel. If it is a valid kernel function, prove it. If it is not, then find a pair of suitable examples $\{x^{(1)}, x^{(2)}\}$ to construct a matrix $\mathbf{K}$ where $\mathbf{K}_{ij} = K(x^{(i)}, x^{(j)})$, and then find a vector $z \in \mathbb{R}^2$ such that $z^T \mathbf{K} z < 0$.

(b) [1 points]

Is the function K defined as $K(x, z) := x^T G z$ where $G = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$ a valid kernel?

Hint: You might find the result of part (a) useful.

Answer:

Yes. The eigen values λ_1, λ_2 are solutions to $\lambda_1 + \lambda_2 = 4$ and $\lambda_1 \lambda_2 = 3$ i.e., $\lambda_1 = 3, \lambda_2 = 1$. So G is PSD, and by part-a, K is a valid kernel.

(c) [1 points]

Is the function K defined as $K(x, z) := x^T G z$ where $G = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$ a valid kernel?

Answer:

No. Consider examples $x^{(1)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $x^{(2)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. This gives us the kernel matrix $\mathbf{K} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$. Now consider the vector $z = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$. This gives $z^T \mathbf{K} z = -2 < 0$. Hence K is not a valid kernel function.

Show me the features!

(d) [3 points] Consider the setting of part (a), with $x \in \mathbb{R}^2$ and $G = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}$. Write out the explicit feature map $\phi(x)$ for the kernel $K(x, z) = x^T G z$ where $\phi(x)^T \phi(z) = K(x, z)$.

Answer:

We have the decomposition $G = Q \Lambda Q^T$ where $Q = I$ and $\Lambda^{1/2} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$. Thus we have

$$\begin{aligned} \phi(x) &= (Q\Lambda^{1/2})^T x \\ &= 2x. \end{aligned}$$

That's all! Congratulations on completing the midterm exam!