Your Name
email@fas.harvard.edu
CS281-F17

Assignment #1 v 1.2
Due: 5:00pm September 22, 2017

Collaborators: John Doe, Fred Doe

Homework 1: Fundamentals - Solutions

Grading Instructions

In the solutions, you will see several highlighted checkpoints. These each have a label that corresponds to an entry in the Canvas quiz for this problem set. The highlighted statement should clearly indicate the criteria for being correct on that problem. If you satisfy the criteria for a problem being correct, mark "Yes" on the corresponding position on the Canvas quiz. Otherwise, mark "No". Your homework scores will be verified by course staff at a later date.

That being said, many of the problems in this course will be proofs. If you find a proof that isn't referred to by the course solution, don't worry. If you're uncertain about the proof, make a Piazza post. If you're certain, mark it correct (and we'll look at it during verification).

Introduction

There is a mathematical component and a programming component to this homework. Please submit your PDF and Python files to Canvas, and push all of your work to your GitHub repository. If a question requires you to make any plots, please include those in the writeup.

This assignment is intended to ensure that you have the background required for CS281. You should be able to answer the problems below without complicated calculations.

Problem 1 (A Classic on the Gaussian Algebra, 10pts)

Let X and Y be independent univariate Gaussian random variables. In the previous problem set, you likely used the closure property that $Z = X + Y$ is also a Gaussian random variable. Here you'll prove this fact.

(a) Suppose X and Y have mean 0 and variances σ_X^2 and σ_Y^2 respectively. Write the pdf of $X + Y$ as an integral.

(b) Evaluate the integral from the previous part to find a closed-form expression for the pdf of $X+Y$, then argue that this expression implies that $X + Y$ is also Gaussian with mean 0 and variance $\sigma_X^2 + \sigma_Y^2$. Hint: what is the integral, over the entire real line, of

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

i.e., the pdf of a univariate Gaussian random variable?

(c) Extend the above result to the case in which X and Y may have arbitrary means.

(d) Univariate Gaussians are supported on the entire real line. Sometimes this is undesirable because we are modeling a quantity with positive support. A common way to transform a Gaussian to solve this problem is to exponentiate it. Suppose X is a univariate Gaussian with mean μ and variance σ^2. What is the pdf of e^X?

Solution

(a) Let $f(z)$ be the density of $Z = X + Y$. Since $Z|X \sim \mathcal{N}(X, \sigma_Y^2)$, the law of total probability gives

$$\begin{aligned} f(z) &= \int_{-\infty}^{\infty} \mathcal{N}(\alpha; 0, \sigma_X^2)\mathcal{N}(z; \alpha, \sigma_Y^2)d\alpha \\ &= \frac{1}{2\pi\sigma_X\sigma_Y} \int_{-\infty}^{\infty} \exp\left[-\frac{1}{2}\left(\frac{\alpha^2}{\sigma_X^2} + \frac{(z-\alpha)^2}{\sigma_Y^2}\right)\right] d\alpha \end{aligned}$$

which is also recognizable as the convolution of the pdfs of X and Y.

Check 1a: You wrote down the PDF, using the procedure above or possibly another like Moment Generating Functions. If done like above, note that it is okay for the function to still have the integral and not yet have mean and variance in an obvious form.

(b) To evaluate this integral, let us first pull as much as we can outside the integral by separating the integrand into terms that depend on α and terms that do not. We write

$$\frac{\alpha^2}{\sigma_X^2} + \frac{(z-\alpha)^2}{\sigma_Y^2} = \left(\frac{\alpha^2}{\sigma_X^2} + \frac{\alpha^2}{\sigma_Y^2} - 2\frac{\alpha z}{\sigma_Y^2}\right) + \frac{z^2}{\sigma_Y^2}$$

and so

$$f(z) = \frac{1}{2\pi\sigma_X\sigma_Y} \exp\left[-\frac{z^2}{2\sigma_Y^2}\right] \int_{-\infty}^{\infty} \exp\left[-\frac{1}{2}\left(\frac{\alpha^2}{\sigma_X^2} + \frac{\alpha^2}{\sigma_Y^2} - 2\frac{\alpha z}{\sigma_Y^2}\right)\right] d\alpha$$

That integral still looks challenging. However, it has the feature that it is an exponential of something quadratic in α, just like a Gaussian density. So if we can massage it into something that is exactly a Gaussian density then we'll already know what it integrates to. We do this by completing the square

as follows

$$\begin{aligned}\frac{\alpha^2}{\sigma_X^2}+\frac{\alpha^2}{\sigma_Y^2}-2\frac{\alpha z}{\sigma_Y^2} &= \alpha^2\left(\frac{1}{\sigma_X^2}+\frac{1}{\sigma_Y^2}\right)-2\alpha\frac{z}{\sigma_Y^2}\\ &= \frac{1}{s^2}\left(\alpha^2-2\alpha\frac{zs^2}{\sigma_Y^2}\right)\\ &= \frac{1}{s^2}\left(\alpha^2-2\alpha\frac{zs^2}{\sigma_Y^2}+\frac{z^2s^4}{\sigma_Y^4}-\frac{z^2s^4}{\sigma_Y^4}\right)\\ &= \frac{1}{s^2}\left(\alpha-z\frac{s^2}{\sigma_Y^2}\right)^2-\frac{z^2s^2}{\sigma_Y^4}\end{aligned}$$

where

$$s^2=\left(\frac{1}{\sigma_X^2}+\frac{1}{\sigma_Y^2}\right)^{-1}=\frac{\sigma_X^2\sigma_Y^2}{\sigma_X^2+\sigma_Y^2}.$$

Letting $\sigma^2=\sigma_X^2+\sigma_Y^2$, we can simplify this as

$$\frac{1}{s^2}\left(\alpha-z\frac{s^2}{\sigma_Y^2}\right)^2-\frac{z^2s^2}{\sigma_Y^4}=\frac{1}{s^2}\left(\alpha-z\frac{s^2}{\sigma_Y^2}\right)^2-\frac{z^2}{\sigma^2}\cdot\frac{\sigma_X^2}{\sigma_Y^2}$$

We can now re-write our expression for $f(z)$ as

$$f(z)=\frac{1}{2\pi\sigma_X\sigma_Y}\exp\left[-\frac{z^2}{2}\left(\frac{1}{\sigma_Y^2}-\frac{1}{\sigma^2}\cdot\frac{\sigma_X^2}{\sigma_Y^2}\right)\right]\int_{-\infty}^{\infty}\exp\left[-\frac{1}{2s^2}\left(\alpha-z\frac{s^2}{\sigma_Y^2}\right)^2\right]d\alpha$$

The exponential inside the integral is exactly the pdf of a univariate normal with mean zs^2/σ_Y^2 and variance s^2, except it's missing the $1/(\sqrt{2\pi}s)$ term. Since the density of this univariate normal must integrate to 1, this tells us that the integral above must equal $\sqrt{2\pi}s$. This yields:

$$\begin{aligned}f(z) &= \frac{\sqrt{2\pi}s}{2\pi\sigma_X\sigma_Y}\exp\left[-\frac{z^2}{2}\left(\frac{1}{\sigma_Y^2}-\frac{1}{\sigma^2}\cdot\frac{\sigma_X^2}{\sigma_Y^2}\right)\right]\\ &= \frac{1}{\sqrt{2\pi(\sigma_X^2+\sigma_Y^2)}}\exp\left[-\frac{z^2}{2}\left(\frac{1}{\sigma_Y^2}-\frac{1}{\sigma^2}\cdot\frac{\sigma_X^2}{\sigma_Y^2}\right)\right]\end{aligned}$$

To clean up the exponential term, we note that

$$\begin{aligned}\frac{1}{\sigma_Y^2}-\frac{1}{\sigma^2}\cdot\frac{\sigma_X^2}{\sigma_Y^2} &= \frac{\sigma^2-\sigma_X^2}{\sigma^2\sigma_Y^2}\\ &= \frac{\sigma_Y^2}{\sigma^2\sigma_Y^2}\\ &= \frac{1}{\sigma_X^2+\sigma_Y^2}\end{aligned}$$

which gives that

$$f(z)=\frac{1}{\sqrt{2\pi(\sigma_X^2+\sigma_Y^2)}}\exp\left(-\frac{z^2}{2(\sigma_X^2+\sigma_Y^2)}\right).$$

This shows that $f(z)$ is a Gaussian density with mean 0 and variance $\sigma_X^2+\sigma_Y^2$ as desired.

Check 1b: You wrote the PDF, this time in the form of a normal PDF where the mean and variance are clear

(c) Let $X' = X - \mathbb{E}(X)$ and $Y' = Y - \mathbb{E}(Y)$. Clearly $\mathbb{E}(X) = 0$, and since X' and X differ by a constant, $\text{var}(X') = \text{var}(X)$. The same applies to Y', and so by the previous part, $X' + Y' \sim \mathcal{N}(0, \sigma_X^2 + \sigma_Y^2)$. But $X + Y = (X' + Y') + (\mathbb{E}(X) + \mathbb{E}(Y))$ differs from $X' + Y'$ by a constant, so it is normally distributed as well and has the same variance. Linearity of expectation then gives that $X + Y \sim \mathcal{N}\left(\mathbb{E}(X) + \mathbb{E}(Y), \sigma_X^2 + \sigma_Y^2\right)$.

Check 1c: Using the linearity of expectation and the fact that absolute position does not affect variance, you showed the PDF. Note that if you re-derived the PDF from scratch, that this is not a strictly wrong way to do the problem.

(d) We define $g(x) = e^x$, and note that $Y = e^X = g(X)$. Since $g(x)$ is monotonic, the change of variables formula for pdf's tells us that if f_X is the density of X and f_Y is the density of Y, then

$$f_Y(y) = \left|\frac{d}{dy}(g^{-1}(y))\right| f_X(g^{-1}(y)).$$

Plugging in $g^{-1}(y) = \ln y$ and noting that the derivative of g^{-1} is $1/y$, we obtain

$$f_Y(y) = f_X(\ln y)/y$$

which gives

$$f_Y(y) = \frac{1}{\sqrt{2\pi} y \sigma} \exp\left(-\frac{(\ln y - \mu)^2}{2\sigma^2}\right)$$

Check 1d: You showed the PDF of e^x. No justification was needed, but be sure to understand change of random variables.

Problem 2 (Regression, 13pts)

Suppose that $X \in \mathbb{R}^{n \times m}$ with $n \geq m$ and $Y \in \mathbb{R}^n$, and that $Y \sim \mathcal{N}(Xw, \sigma^2 I)$. You learned in class that the maximum likelihood estimate $\hat{w}$ of w is given by

$$\hat{w} = (X^T X)^{-1} X^T Y$$

(a) Why do we need to assume that $n \geq m$?

(b) Define $H = X(X^T X)^{-1} X^T$, so that the "fitted" values $\hat{Y} = X\hat{w}$ satisfy $\hat{Y} = HY$. Show that H is an orthogonal projection matrix that projects onto the column space of X, so that the fitted y-values are a projection of Y onto the column space of X.

(c) What are the expectation and covariance matrix of $\hat{w}$?

(d) Compute the gradient with respect to w of the log likelihood implied by the model above, assuming we have observed Y and X.

(e) Suppose we place a normal prior on w. That is, we assume that $w \sim \mathcal{N}(0, \tau^2 I)$. Show that the MAP estimate of w given Y in this context is

$$\hat{w}_{MAP} = (X^T X + \lambda I)^{-1} X^T Y$$

where $\lambda = \sigma^2/\tau^2$. (You may employ standard conjugacy results about Gaussians without proof in your solution.)

[Estimating w in this way is called ridge regression because the matrix λI looks like a "ridge". Ridge regression is a common form of regularization that is used to avoid the overfitting (resp. underdetermination) that happens when the sample size is close to (resp. higher than) the output dimension in linear regression.]

(f) Do we need $n \geq m$ to do ridge regression? Why or why not?

(g) Show that ridge regression is equivalent to adding m additional rows to X where the j-th additional row has its j-th entry equal to $\sqrt{\lambda}$ and all other entries equal to zero, adding m corresponding additional entries to Y that are all 0, and then computing the maximum likelihood estimate of w using the modified X and Y.

Solution

(a) The reason for the assumption that $n \geq m$ is that we need the matrix $X^T X$ to be invertible. The matrix is $m \times m$, so this means that its rank must be at least m. But as we will show, its rank is at most n, and so if $n < m$, there is no hope of inverting $X^T X$.

To upper-bound the rank of $X^T X$, we let the column-vector x_i denote the i-th observation (i.e., x_i is m-by-1), so that

$$X^T X = \begin{pmatrix} x_1 & \cdots & x_m \end{pmatrix} \cdot \begin{pmatrix} x_1^T \\ \vdots \\ x_m^T \end{pmatrix} = \sum_{i=1}^{n} x_i \cdot x_i^T.$$

The result then follows easily, for each matrix $X_i \cdot X_i^T$ is a rank-1 n-by-n matrix and $X^T X$, beings a sum of n of these, cannot have its rank exceed n.

Check 2a: You described the necessity for $\mathbf{X}^\top \mathbf{X}$ to be invertible, and stated that the rank cannot exceed n.

(b) To show that H is an orthogonal projection onto the image of X, we need to establish two things: first, that if $v \in \mathrm{Im}(X)$ then $Hv = v$, and, second, that if v is orthogonal to $\mathrm{Im}(X)$ then $Hv = 0$. To establish the first claim, we can write v generically as Xa for some m-dimensional vector a. We then have

$$Hv = X(X^TX)^{-1}X^T(Xa) = X(X^TX)^{-1}(X^TX)a = Xa = v$$

as desired. The second claim is even easier, for the i-th component of X^Tv equals the inner product of v with the i-th column of X, so if v is orthogonal to $\mathrm{Im}(X)$, then $X^Tv = 0$.

Check 2b: You showed that H is a projection onto the column space of X.

(c) Representing Y as $Y = X\mathbf{w} + \varepsilon$ for $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$, we have

$$\hat{\mathbf{w}} = (X^TX)^{-1}X^TX\mathbf{w} + (X^TX)^{-1}X^T\varepsilon = \mathbf{w} + (X^TX)^{-1}X^T\varepsilon.$$

Since $\mathbb{E}(\varepsilon) = 0$, this easily gives that $\mathbb{E}(\hat{\mathbf{w}}) = \mathbf{w}$. To compute the variance, we write

$$\begin{aligned}
\mathrm{var}(\hat{\mathbf{w}}) &= \mathbb{E}\left((X^TX)^{-1}X^T\varepsilon((X^TX)^{-1}X^T\varepsilon)^T\right) \\
&= \mathbb{E}\left((X^TX)^{-1}X^T\varepsilon\varepsilon^TX(X^TX)^{-1}\right) \\
&= (X^TX)^{-1}X^T\mathbb{E}\left(\varepsilon\varepsilon^T\right)X(X^TX)^{-1} \\
&= (X^TX)^{-1}X^T(\sigma^2 I)X(X^TX)^{-1} \\
&= \sigma^2(X^TX)^{-1}
\end{aligned}$$

(This is a special case of the general fact that if $Z \sim \mathcal{N}(\mu, \Sigma)$ and A is a matrix, then $AZ \sim \mathcal{N}(A\mu, A\Sigma A^T)$.)

Check 2c.1: Using any number of techniques, you showed the correct expectation of $\hat{\mathbf{w}}$

Check 2c.2: Using any number of techniques, you showed the correct covariance of $\hat{\mathbf{w}}$

(d) The density of Y given X is

$$\begin{aligned}
f(y|X, \mathbf{w}) &= (2\pi)^{-n/2}\sigma^{-n}\exp\left(-\frac{1}{2}\left((y - X\mathbf{w})^T(\sigma^2 I)^{-1}(y - X\mathbf{w})\right)\right) \\
&= (2\pi)^{-n/2}\sigma^{-n}\exp\left(-\frac{(y - X\mathbf{w})^T(y - X\mathbf{w})}{2\sigma^2}\right).
\end{aligned}$$

The log-likelihood is therefore

$$\ell(\mathbf{w}) = -\frac{n}{2}\left(\ln\left(\frac{2}{\pi}\right) + 2\ln\sigma\right) - \frac{1}{2\sigma^2}(y - X\mathbf{w})^T(y - X\mathbf{w})$$

Since the first term is free of $\mathbf{w}$, it suffices to take the gradient of the second term only. To do so, we write

$$(y - X\mathbf{w})^T(y - X\mathbf{w}) = y^Ty + \mathbf{w}^T(X^TX)\mathbf{w} - 2y^TX\mathbf{w}$$

from which application of the identities proven in the previous problem set quickly gives

$$\nabla\left((y - X\mathbf{w})^T(y - X\mathbf{w})\right) = 2(\mathbf{w}^TX^TX - y^TX)$$

Putting everything together, we get that the gradient of the log-likelihood is as follows.

$$\nabla\ell(\mathbf{w}) = -\frac{1}{\sigma^2}(\mathbf{w}^TX^TX - y^TX)$$

Notice that the solution to $\nabla\ell(\mathbf{w}) = 0$ is indeed the MLE of $\mathbf{w}$. It's also free of σ, which means we do not need to know the magnitude of our noise in order to estimate $\mathbf{w}$. The latter fact is fortunate, but is a special feature of this model and not a foregone conclusion.

Check 2d: You wrote down the correct gradient.

(e) In this solution we utilize the following lemma, a special case of which you effectively proved in the previous problem set and which gives a version of Bayes rule for Gaussians.

Lemma 1

Let $\mathbf{w} \sim \mathcal{N}(\mu, \Sigma)$ and let $Y|\mathbf{w} \sim \mathcal{N}(A\mathbf{w}, \Sigma')$. Then $\mathbf{w}|Y \sim \mathcal{N}(\mu_{\mathbf{w}|Y}, \Sigma_{\mathbf{w}|Y})$, where

$$\Sigma_{\mathbf{w}|Y}^{-1} = \Sigma^{-1} + A^T \Sigma'^{-1} A$$

and

$$\mu_{\mathbf{w}|Y} = \Sigma_{\mathbf{w}|Y} \left(\Sigma^{-1}\mu + A^T \Sigma'^{-1} Y \right)$$

Plugging in $\mu = 0$, $\Sigma = \tau^2 I$, $A = X$, and $\Sigma' = \sigma^2 I$ above gives

$$\Sigma_{\mathbf{w}|Y}^{-1} = \frac{1}{\tau^2} I + \frac{1}{\sigma^2} X^T X = \frac{1}{\sigma^2} \left(X^T X + \frac{\sigma^2}{\tau^2} \right)$$

and, therefore,

$$\mu_{\mathbf{w}|Y} = \sigma^2 (X^T X + \lambda I)^{-1} \left(\frac{1}{\sigma^2} X^T Y \right) = (X^T X + \lambda I)^{-1} X^T Y.$$

Since the mean of a normal distribution is its mode, the MAP estimate of $\mathbf{w}$ given Y is $\mu_{\mathbf{w}|Y}$, and so we are done.

An alternate way to do this is to take the derivative of the log likelihood that we already have:

$$\frac{\partial}{\partial w} \ln p(\mathbf{w}|\mathbf{X}, \mathbf{Y}) = -2\sigma^{-2} \mathbf{X}^\top (\mathbf{Y} - \mathbf{X}\mathbf{w}) + 2\tau^{-2}\mathbf{w}$$

Set the partial derivative equal to 0:

$$\sigma^{-2} \mathbf{X}^\top (\mathbf{Y} - \mathbf{X}w) = \tau^{-2}\mathbf{w}$$

Distribute out $\mathbf{X}^\top$:

$$\mathbf{w}_{map} = \left(\mathbf{X}^\top \mathbf{X} + \frac{\sigma^2}{\tau^2} \mathbf{I} \right)^{-1} \mathbf{X}^\top \mathbf{Y}$$

We see that $\lambda = \frac{\sigma^2}{\tau^2}$

Check 2e: You showed the above relationship between λ, σ, and τ.

(f) We do not require $n \geq m$ for ridge regression. This is because the matrix $X^T X + \lambda I$ is always invertible for positive λ, for the following reason: $X^T X$, being a sample covariance matrix, is positive definite. (You can prove this by considering the SVD of X.) Therefore, its eigenvalues are non-negative. Adding λI to $X^T X$ adds λ to each eigenvalue, and therefore renders all the eigenvalues strictly positive. Therefore, $X^T X + \lambda I$ is invertible regardless of the original rank of $X^T X$.

Check 2f: You stated that $\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}$ is always invertible for positive λ, regardless of the original, potentially insufficient rank of $\mathbf{X}^\top \mathbf{X}$.

(g) Let X' and Y' denote the modified X and Y respectively. It is easy to see that $X'^T Y' = X^T Y$: the reason is that the last m entries of Y' are 0. It remains then only to show that $X'^T X' = X^T X + \lambda I$. But this is also simple, because the i, j-th entry of $X'^T X'$ is the inner product of the i-th and j-th columns of X'. If $i \neq j$, this will equal the inner product of the corresponding columns of X', i.e., the i, j-th entry of $X^T X$. But when $i = j$, it will equal the squared norm of the i-th column of X plus $(\sqrt{\lambda})^2 = \lambda$.

Check 2g: You showed that $X'^\top Y' = X^\top Y$ and that $X'^\top X' = X^\top X + \lambda I$, probably by acknowledging that the i^{th}, j^{th} entry of a matrix $A^\top A$ is equal to the dot product of A's i^{th} and j^{th} columns.

Problem 3 (The Dirichlet and Multinomial Distributions, 12pts)

The Dirichlet distribution over K categories is a generalization of the beta distribution. It has a shape parameter $\alpha \in \mathbb{R}^K$ with non-negative entries and is supported over the set of K-dimensional positive vectors whose components sum to 1. Its density is given by

$$f(\theta_{1:K}|\alpha_{1:K}) = \frac{\Gamma\left(\sum_k \alpha_k\right)}{\prod_k \Gamma(\alpha_k)} \prod_{k=1}^{K} \theta_k^{\alpha_k - 1}$$

(Notice that when $K = 2$, this reduces to the density of a beta distribution.) For the rest of this problem, assume a fixed $K \geq 2$.

(a) Suppose θ is Dirichlet-distributed with shape parameter α. Without proof, state the value of $E(\theta)$. Your answer should be a vector defined in terms of either α or K or potentially both.

(b) Suppose that $\theta \sim \text{Dir}(\alpha)$ and that $X \sim \text{Cat}(\theta)$, where *Cat* is a Categorical distribution. That is, suppose we first sample a K-dimensional vector θ with entries in $(0, 1)$ from a Dirichlet distribution and then roll a K-sided die such that the probability of rolling the number k is θ_k. Prove that the posterior $p(\theta|X)$ also follows a Dirichlet distribution. What is its shape parameter?

(c) Now suppose that $\theta \sim \text{Dir}(\alpha)$ and that $X^{(1)}, X^{(2)}, \ldots \overset{iid}{\sim} \text{Cat}(\theta)$. Show that the posterior predictive after $n-1$ observations is given by,

$$P(X^{(n)} = k|X^{(1)}, \ldots, X^{(n-1)}) = \frac{\alpha_k^{(n)}}{\sum_k \alpha_k^{(n)}}$$

where for all k, $\alpha_k^{(n)} = \alpha_k + \sum_{i=1}^{n-1} \mathbf{1}\{X^{(i)} = k\}$. (Bonus points if your solution does not involve any integrals.)

(d) Consider the random vector $Z_k = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\{X^{(i)} = k\}$ for all k. What is the mean of this vector? What is the distribution of the vector? (If you're not sure how to rigorously talk about convergence of random variables, give an informal argument. Hint: what would you say if θ were fixed?) What is the marginal distribution of a single class $p(Z_k)$?

(e) Suppose we have K distinct colors and an urn with α_k balls of color k. At each time step, we choose a ball uniformly at random from the urn and then add into the urn an additional new ball of the same color as the chosen ball. (So if at the first time step we choose a ball of color 1, we'll end up with $\alpha_1 + 1$ balls of color 1 and α_k balls of color k for all $k > 1$ at the start of the second time step.) Let $\rho_k^{(n)}$ be the fraction of all the balls that are of color k at time n. What is the distribution of $\lim_{n\to\infty} \rho_k^{(n)}$? Prove your answer.

Solution

(a) The mean of the Dirichlet distribution with shape parameter $a = (a_1, \ldots, a_K)$ is the vector

$$\frac{a}{\sum_{k=1}^{K} a_k}$$

Check 3a: You stated the correct mean.

(b) We have a prior of $\theta \sim \text{Dirichlet}(a)$ and a likelihood of $X|\theta \sim \text{Categorical}(\theta)$. Using Bayes rule and

letting X_i be 1 if $X = i$ and 0 otherwise, we obtain

$$\begin{aligned} P(\theta|X) &\propto_\theta P(X|\theta)P(\theta) \\ &\propto_\theta \Pi_{k=1}^K \theta_k^{\mathbb{1}\{X=k\}} \Pi_{k=1}^K \theta_k^{a_k-1} \\ &= \Pi_{k=1}^K \theta_k^{a_k+\mathbb{1}\{X=k\}-1} \end{aligned}$$

and so the posterior must also be a Dirichlet distribution, and the k-th entry of its shape parameter is $a_k + \mathbb{1}\{X = k\}$.

Check 3b: You demonstrated that the posterior is also Dirichlet and stated its shape parameter.

(c) You can do this problem by conditioning on θ and using the law of total expectation, and then using simple identities involving the beta integral together with properties of the gamma function. However, a simpler solution that just uses the law of total expectation ("Adams' law") is as follows: we know by repeated application of the previous problem that

$$\theta|X_1, \ldots, X_{n-1} \sim \text{Dirichlet}((a_{1,n}, \ldots, a_{K,n}))$$

where the $a_{k,n}$ are as in the problem statement. We also know that $P(X_n = k|X_1, \ldots, X_{n-1}) = \mathbb{E}(\mathbb{1}(X_n = k)|X_1, \ldots, X_{n-1})$. But

$$\begin{aligned} \mathbb{E}(\mathbb{1}(X_n = k)|X_1, \ldots, X_{n-1}) &= \mathbb{E}\left(\mathbb{E}(\mathbb{1}(X_n = k)|\theta, X_1, \ldots, X_{n-1})|X_1, \ldots, X_{n-1}\right) \\ &= \mathbb{E}\left(\mathbb{E}(\mathbb{1}(X_n = k)|\theta)|X_1, \ldots, X_{n-1}\right) \\ &= \mathbb{E}\left(\theta_k|X_1, \ldots, X_{n-1}\right) \\ &= \frac{a_{k,n}}{\sum_k a_{k,n}} \end{aligned}$$

as desired. Alternatively:

$$\begin{aligned} p(X^n = k|X^1, ..., X^{n-1}) &= \int_\theta p(X^n = k|\theta)p(\theta|x^1, ..., X^{n-1}) \\ &\propto \int_\theta \theta_k \prod_{i=1}^{n-1} \prod_{k=1}^{C} \theta_k^{\mathbb{1}(X^i=k)} \\ &= \int_\theta \prod_k \theta_k^{\alpha_k - 1} = \mathbb{E}[\theta] \end{aligned}$$

Check 3c.1: You derived the posterior predictive.

Check 3c.2 (bonus): Your solution did not involve an integral.

(d) Given θ_k, the k-th component of θ, $\mathbb{1}\{X_i = k\}|\theta_k \overset{i.i.d.}{\sim} \text{Bernouli}(\theta_k)$. The strong law of large numbers tells us that the sequence $\frac{1}{n}\sum_{i=1}^n \mathbb{1}\{X_i = k\}|\theta_k$ converges almost surely to θ_k. Therefore, the limiting distribution of $\frac{1}{n}\sum_{i=1}^n \mathbb{1}\{X_i = k\}$ is exactly that of θ_k (by dominated convergence theorem). We know that $\theta \sim \text{Dirichlet}(\alpha)$, and the marginal of a Dirichlet distribution is the beta distribution: $\theta_k \sim \text{Beta}(\alpha_k, \sum_{j\neq k} \alpha_j)$. Thus, $Z_k = \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^n \mathbb{1}\{X_i = k\} \sim \text{Beta}(\alpha_k, \sum_{j\neq k} \alpha_j)$ and $E(Z_k) = \frac{\alpha_k}{\sum_k \alpha_k}$.

Check 3d: You noted that the indicator RV of $X_i = k$ given a certain θ is distributed Bernouli(θ_k). Using the law of large numbers, or informally the wisdom behind using large sample sizes, you noted that the average of the indicator RV over many samples converges to the value of the parameter θ_k. Considering θ_k is distributed Dirichlet, you stated the marginal and expectation of the averaged indicator RV

(e) In part(c), we have showed that there are two equivalent ways to generate samples from Multinomial-Dirichlet distribution. (1) first generate the parameter θ from the Dirichlet distribution and then draw n i.i.d samples from the categorical distribution with parameter θ (2) sequentially sample from posterior predictive distribution given previous samples: first sample X_1 from its marginal distribution, then sample X_2 from $X_2|X_1$, then sample X_3 from $X_3|X_2, X_1$, and so on. For any finite n, these two generating processes have the same joint distribution:

$$P(X^{(1)}, X^{(2)}, ..., X^{(n)}) = \int P(X^{(1)}, X^{(2)}, ..., X^{(n)}|\theta)P(\theta|\alpha)d\theta \tag{1}$$

$$= \prod_{i=1}^{n} P(X^{(i)} = k|X^{(1)}, X^{(2)}, ..., X^{(i-1)}) = \prod_{i=1}^{n} \frac{\alpha_k^{(i)}}{\sum_k \alpha_k^{(i)}} = \prod_{i=1}^{n} \frac{\alpha_k + \sum_{j=1}^{i-1} \mathbf{1}\{X^{(j)} = k\}}{n - 1 + \sum_{k=1}^{K} \alpha_k} \tag{2}$$

In the view of the first generating process, part (d) then implies that if we do this procedure for infinitely many time-steps, the limiting distribution of $\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{X_i = k\}$ will be Beta$(\alpha_k, \sum_{j\neq k} \alpha_j)$.

For this question, which describes a statistical model called Polya urn model, we will utilize the second generating process. We will first show that if we run the urn experiment for n steps, the color of the ball added to the urn at each step has the same joint distribution as $(X_1, X_2, ...X_n)$ using the results in part (c). Then, we will use the result from (d) to show that $\lim_{n\to\infty} \rho_k^{(n)} = \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{X_i = k\}$ will also be Beta$(\alpha_k, \sum_{j\neq k} \alpha_j)$.

- The probability of adding a ball of color k at time n given the colors of the balls in the urn is exactly equal to $P(X^{(i)} = k|X^{(1)}, X^{(2)}, ..., X^{(i-1)})$ in (c). To be more specific, let Y_i be the color of ball chosen at step i; at time 1, there are α_k balls of color k in the urn, so $P(Y_1 = k) = \frac{\alpha_k}{\sum_k \alpha_k}$; at time n, there will be $\sum_{i=1}^{n-1} \mathbb{1}\{Y_i = k\}+\alpha_k$ balls of color k in the urn, so the probability of choosing color k will be $P(Y_n = k|Y_1, \ldots, Y_{n-1}) = \frac{\sum_{i=1}^{n-1} \mathbb{1}\{Y_i=k\}+\alpha_k}{n-1+\sum_k \alpha_k}$. Thus, for any finite n, $(Y_1, ...Y_n)$ have the same joint distribution as $(X_1, ...X_n)$ defined in (c): $P(Y_1, ...Y_n) = \prod_{i=1}^{n} P(Y_i|Y_1, Y_2, ..., Y_{i-1}) = \prod_{i=1}^{n} P(X^{(i)}|X^{(1)}, X^{(2)}, ..., X^{(i-1)}) = P(X^{(1)}, X^{(2)}, ..., X^{(n)})$. Therefore, running the urn for n time steps and then tallying the colors of the balls added to the urn will exactly give us a sample from the joint distribution of $(X_1, \ldots, X_n)$.
- By definition, $\rho_k^{(n)} = \frac{\sum_{i=1}^{n} \mathbb{1}\{Y_i=k\}+\alpha_k}{n+\sum_k \alpha_k}$, for any fixed α, $\rho_k^{(n)}$ will eventually be dominated by the contribution of the new balls as n grows: $\frac{\sum_{i=1}^{n} \mathbb{1}\{Y_i=k\}+\alpha_k}{n+\sum_k \alpha_k} - \frac{\sum_{i=1}^{n} \mathbb{1}\{Y_i=k\}}{n} = \frac{n\alpha_k - \sum_k \alpha_k \cdot \sum_{i=1}^{n} \mathbb{1}\{Y_i=k\}}{n(n+\sum_k \alpha_k)} = \frac{\alpha_k - \sum_k \alpha_k \cdot \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{Y_i=k\}}{n+\sum_k \alpha_k}$ since $\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{Y_i = k\}$ is bounded between 0 and 1, the difference between $\rho_k^{(n)}$ and $\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{Y_i = k\}$ will goes to 0 as $n \to \infty$. This means that $\lim_{n\to\infty} \rho_k^{(n)} = \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{Y_i = k\}$ has the same distribution as $\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{X_i = k\}$, and so it will also be Beta$(\alpha_k, \sum_{j\neq k} \alpha_j)$.

Check 3e: You correctly showed that the fraction of balls of color k is distributed Beta$(\alpha_k, \sum_{j\neq k} \alpha_j)$.

Physicochemical Properties of Protein Tertiary Structure

In the following problems we will code two different approaches for solving linear regression problems and compare how they scale as a function of the dimensionality of the data. We will also investigate the effects of linear and non-linear features in the predictions made by linear models.

We will be working with the regression data set Protein Tertiary Structure: `https://archive.ics.uci.edu/ml/machine-learning-databases/00265/` (download CASP.csv). This data set contains information about predicted conformations for 45730 proteins. In the data, the target variable y is the root-mean-square deviation (RMSD) of the predicted conformations with respect to the true properly folded form of the protein. The RMSD is the measure of the average distance between the atoms (usually the backbone atoms) of superimposed proteins. The features $\mathbf{x}$ are physico-chemical properties of the proteins in their true folded form. After downloading the file CASP.csv we can load the data into python using

```
>>> import numpy as np
>>> data = np.loadtxt("CASP.csv", delimiter = ",", skiprows = 1)
```

We can then obtain the vector of target variables and the feature matrix using

```
>>> y = data[:, 0]
>>> X = data[:, 1:]
```

We can then split the original data into a training set with 90% of the data entries in the file CASP.csv and a test set with the remaining 10% of the entries. Normally, the splitting of the data is done at random, but here **we ask you to put into the training set the first 90% of the elements from the file CASP.csv** so that we can verify that the values that you will be reporting are correct. (This should not cause problems, because the rows of the file are in a random order.)

We then ask that you **normalize** the features so that they have zero mean and unit standard deviation in the training set. This is a standard step before the application of many machine learning methods. After these steps are done, we can concatenate a **bias feature** (one feature which always takes value 1) to the observations in the normalized training and test sets.

We are now ready to apply our machine learning methods to the normalized training set and evaluate their performance on the normalized test set. In the following problems, you will be asked to report some numbers and produce some figures. Include these numbers and figures in your assignment report. **The numbers should be reported with up to 8 decimals**.

Problem 4 (7pts)

Assume that the targets y are obtained as a function of the normalized features $\mathbf{x}$ according to a Bayesian linear model with additive Gaussian noise with variance $\sigma^2 = 1.0$ and a Gaussian prior on the regression coefficients $\mathbf{w}$ with *precision* matrix $\Sigma^{-1} = \tau^{-2}\mathbf{I}$ where $\tau^{-2} = 10$. Code a routine using the **QR decomposition** (see Section 7.5.2 in Murphy's book) that finds the Maximum a Posteriori (MAP) value $\hat{\mathbf{w}}$ for $\mathbf{w}$ given the normalized training data

- Report the value of $\hat{\mathbf{w}}$ obtained.
- Report the root mean squared error (RMSE) of $\hat{\mathbf{w}}$ in the normalized test set.

Solution

```
[[ 5.55782079]
 [ 2.25190765]
 [ 1.07880135]
 [-5.91177796]
```

```
 [-1.73480336]
 [-1.63875478]
 [-0.26610556]
 [ 0.81781409]
 [-0.65913397]
 [ 7.74153395]]

Error: 5.20880460745
```

Check 4a: You reported your weights
Check 4b: You reported your RMSE, and it was lower than 8.0

Problem 5 (14pts)

L-BFGS is an iterative method for solving general nonlinear optimization problems. For this problem you will use this method as a black box that returns the MAP solution by sequentially evaluating the objective function and its gradient for different input values. The goal of this problem is to use a built-in implementation of the L-BFGS algorithm to find a point estimate that maximizes our posterior of interest. Generally L-BFGS requires your black box to provide two values: the current objective and the gradient of the objective with respect to any parameters of interest. To use the optimizer, you need to first write two functions: (1) to compute the loss, or the *negative* log-posterior and (2) to compute the gradient of the loss with respect to the weights w.

As a preliminary to coming work in the class, we will use the L-BFGS implemented in PyTorch. [Warning: For this assignment we are using a small corner of the PyTorch world. Do not feel like you need to learn everything about this library.]

There are three parts to using this optimizer:

1. Create a vector of weights in NumPy, wrap in a pytorch `Tensor` and `Variable`, and pass to the optimizer.

```
from torch import Tensor
from torch.autograd import Variable

# Construct a PyTorch variable array (called tensors).
weights = Variable(Tensor(size))

# Initialize an optimizer of the weights
optimizer = torch.optim.LBFGS([weights])

...
```

2. Write a python function that uses the current weights to compute the log-posterior **and** sets weights.grad to be the gradient of the log-posterior with respect to the current weights.

```
def black_box():
    # Access the value of the variable as a numpy array.
    weights_data = weights.data.numpy()

    ...

    # Set the gradient of the variable.
    weights.grad = Tensor({numpy})

    return {objective}
```

3. Repeatedly call `optimizer.step(black_box)` to optimize.

[If you are feeling adventurous, you might find it useful to venture into the land of autograd and check your computation with PyTorch's `torch.autograd.gradcheck.get_numerical_jacobian`.]

- After running for 100 iterations, report the value of $\hat{\mathbf{w}}$ obtained.
- Report the RMSE of the predictions made with $\hat{\mathbf{w}}$ in the normalized test set.

Solution

```
[[ 5.55780742]
 [ 2.2516761 ]
 [ 1.078904  ]
 [-5.9118024 ]
 [-1.73438477]
 [-1.63887755]
 [-0.26611567]
 [ 0.81785313]
 [-0.65901996]
 [ 7.7415115 ]]

5.20880242943
```

Check 5a: You reported your weights
Check 5b: You reported your RMSE, and it was lower than 8.0

Problem 6 (14pts)

Linear regression can be extended to model non-linear relationships by replacing the original features $\mathbf{x}$ with some non-linear functions of the original features $\boldsymbol{\phi}(\mathbf{x})$. We can automatically generate one such non-linear function by sampling a random weight vector $\mathbf{a} \sim \mathcal{N}(0, \mathbf{I})$ and a corresponding random bias $b \sim \mathrm{U}[0, 2\pi]$ and then making $\phi(\mathbf{x}) = \cos(\mathbf{a}^{\mathrm{T}}\mathbf{x} + b)$. By repeating this process d times we can generate d non-linear functions that, when applied to the original features, produce a non-linear mapping of the data into a new d dimensional space. We can encode these d functions into a matrix $\mathbf{A}$ with d rows, each one with the weights for each function, and a d-dimensional vector $\mathbf{b}$ with the biases for each function. The new mapped features are then obtained as $\boldsymbol{\phi}(\mathbf{x}) = \cos(\mathbf{A}\mathbf{x} + \mathbf{b})$, where cos applied to a vector returns another vector whose elements are the result of applying cos to the individual elements of the original vector.

Generate 4 sets of non-linear functions, each one with $d = 100, 200, 400, 600$ functions, respectively, and use them to map the features in the original normalized training and test sets into 4 new feature spaces, each one of dimensionality given by the value of d. After this, for each value of d, find the MAP solution $\hat{\mathbf{w}}$ for $\mathbf{w}$ using the corresponding new training set and the method from problem 4. Use the same values for σ^2 and τ^{-2} as before. You are also asked to record the time taken by the method QR to obtain a value for $\hat{\mathbf{w}}$. In python you can compute the time taken by a routine using the time package:

```
>>> import time
>>> time_start = time.time()
>>> routine_to_call()
>>> running_time = time.time() - time_start
```

Next, compute the RMSE of the resulting predictor in the normalized test set. Repeat this process with the method from problem 5 (L-BFGS).

- Report the test RMSE obtained by each method for each value of d.

You are asked to generate a plot with the results obtained by each method (QR and L-BFGS) for each value of d. In this plot the x axis should represent the time taken by each method to run and the y axis should be the RMSE of the resulting predictor in the normalized test set. The plot should contain 4 points in red, representing the results obtained by the method QR for each value of d, and 4 points in blue, representing the results obtained by the method L-BFGS for each value of d. Answer the following questions:

- Do the non-linear transformations help to reduce the prediction error? Why?
- What method (QR or L-BFGS) is faster? Why?
- (Extra Problem, Not Graded) Instead of using random $\mathbf{A}$, what if we treat $\mathbf{A}$ as another parameter for L-BFGS to optimize? You can do this by wrapping it as a variable and passing to the constructor. Compute its gradient as well in *black_box* either analytically or by using PyTorch *autograd*.

Solution

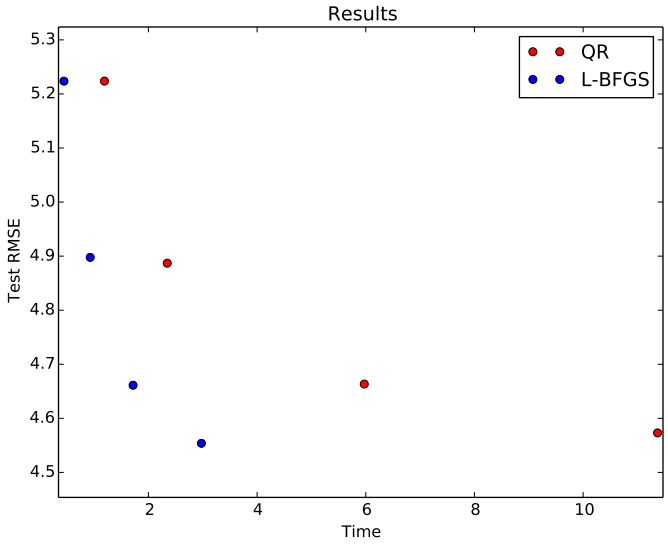
The errors obtained by L-BFGS

```
[ 5.22364062  4.89773226  4.66121231  4.5537481 ]
```

The errors obtained by QR are

```
[ 5.22378849  4.8869642   4.66336145  4.57306643]
```

The plot obtained is shown in Figure 1. The non-linear transformation reduces the prediction error because the true function is probably non-linear. L-BFGS is faster because its cost scales as d^2 and not as d^3.



Plot with the results

Check 6a: You reported RMSE for all four d, for both QR and L-BFGS.

Check 6b: You suggested a reason why the non-linear transformations might help reduce the prediction error.

Check 6c: You suggested a reason for why QR or L-BFGS was faster. This varies on your computing machinery. The inverses in QR decomposition are of triangular matrices and are therefore quick. However, as the dimensionality goes up, solving the actual decomposition becomes slow. This may take less or more time than gradient-based optimization.

Check 6d: You generated a plot to visually present time vs. RMSE