Christopher Hase
christopher_hase@g.harvard.edu
CS281-F17

Assignment #2 v 1.0

Due: 5:00pm October 6, 2017

Collaborators: None

NOTE: you must show derivations for your answers unless a question explicitly mentions that no justification is required.

Problem 1 (Spherical Gaussian, 10pts)

One intuitive way to summarize a probability density is via the mode, as this is the "most likely" value in some sense. A common example of this is using the maximum *a posteriori* (MAP) estimate of a model's parameters. In high dimensions, however, the mode becomes less and less representative of typical samples. Consider variates from a D-dimensional zero mean spherical Gaussian with unit variance:

$$\mathbf{x} \sim \mathcal{N}(\mathbf{0}_D, \mathbb{I}_D),$$

where $\mathbf{0}_D$ indicates a column vector of D zeros and $\mathbb{I}_D$ is a $D \times D$ identity matrix.

1. Compute the distribution that this implies over the distance of these points from the origin. That is, compute the distribution over $\sqrt{\mathbf{x}^{\mathsf{T}}\mathbf{x}}$, if $\mathbf{x}$ is a realization from $\mathcal{N}(\mathbf{0}_D, \mathbb{I}_D)$. (Note: Consider transformations of a Gamma distribution described in Murphy 2.4.5.)
2. Make a plot that shows this probability density function for several different values of D, up to $D = 100$.
3. Make a plot of the cumulative distribution function (CDF) over this distance distribution for $D = 100$. A closed-form solution may be difficult to compute, so you can do this numerically.)
4. From examining the CDF we can think about where most of the mass lives as a function of radius. For example, most of the mass for $D = 100$ is within a thin spherical shell. From eyeballing the plot, what are the inner and outer radii for the shell that contains 90% of the mass in this case?

(a) $\mathbf{x} = \big[X_1, ..., X_D\big]^T$. Using properties of the multivariate normal, we know that $X_i \overset{i.i.d.}{\sim} \mathcal{N}(0,1)$. Then, note that $\sqrt{\mathbf{x}^T\mathbf{x}} = \sqrt{\sum_{i=1}^{D} X_i^2}$. First we'll find the distribution of $Y_i = X_i^2$ with support $[0, \infty)$.

$P(Y_i < y_i) = P(X_i^2 < y_i) = P(-\sqrt{y_i} < X_i < \sqrt{y_i})$

$= 2P(X_i < \sqrt{y_i}) - 1 = 2\Phi(\sqrt{y_i}) - 1$ (by symmetry of the normal distribution; Φ is the standard normal CDF)

$\dfrac{d\big(2\Phi(\sqrt{y_i}) - 1\big)}{dy_i} = \phi(\sqrt{y_i})\dfrac{1}{\sqrt{y_i}}$ (ϕ is the standard normal PDF)

$= \dfrac{1}{\sqrt{2\pi}} y_i^{-\frac{1}{2}} \exp\left(-\dfrac{y_i}{2}\right)$

This is the PDF for a χ^2 random variable with 1 degree of freedom, so $Y_i \sim \chi_1^2 \Rightarrow X_i^2 \sim \chi_1^2$. Now we need to find the distribution of $Z = \sum_{i=1}^{D} X_i^2$ with support $[0, \infty)$. We can do this using the MGF of Z.

$$\mathrm{MGF}_Z(t) = \prod_{i=1}^{D} \mathrm{MGF}_{X_i^2}(t)$$

$$= \prod_{i=1}^{D} (1-2t)^{-\frac{1}{2}}$$

$$= (1-2t)^{-\frac{D}{2}}$$

This the the MGF of a χ^2 random variable with D degrees of freedom, so $Z \sim \chi^2_D \Rightarrow \sum_{i=1}^{D} X_i^2 \sim \chi^2_D$.

Lastly, we need to find the distribution of $W = \sqrt{Z} = \sqrt{\sum_{i=1}^{D} X_i^2}$ with support $[0, \infty)$.

$$P(W < w) = P\big(\sqrt{Z} < w\big)$$

$$= P\big(Z < w^2\big)$$

$$\frac{d\Big(P\big(Z < w^2\big)\Big)}{dw} = 2w f_Z(w^2)$$

$$= \frac{2^{1-\frac{D}{2}} w^{D-1}}{\Gamma\big(\frac{D}{2}\big)} \exp\left(-\frac{w^2}{2}\right)$$

This is the PDF for a χ random variable with D degrees of freedom, so $W \sim \chi_D$

$$\Rightarrow \sqrt{\sum_{i=1}^{D} X_i^2} = \sqrt{\mathbf{x}^T\mathbf{x}} \sim \chi_D \text{ and } f_{\sqrt{\mathbf{x}^T\mathbf{x}}}(x) = \frac{2^{1-\frac{D}{2}} x^{D-1}}{\Gamma\big(\frac{D}{2}\big)} \exp\left(-\frac{x^2}{2}\right)$$

Problem 2 (Hurdle Models for Count Data, 10pts)

In this problem we consider predictive models of count data. For instance given information about the student x, can we predict how often they went to the gym that week y? A natural choice is to use a Poisson GLM i.e. y conditioned on x is modeled as a Poisson distribution.

However, in practice, it is common for count data of this form to follow a bi-modal distribution over count data. For instance, our data may come from a survey asking students how often they went to the gym in the past week. Some would do so frequently, some would do it occasionally but not in the past week (a random zero), and a substantial percentage would never do so.

When modeling this count data with generalized linear models, we may observe more zero examples than expected from our model. In the case of a Poisson, the mode of the distribution is the integer part of the mean. A Poisson GLM may therefore be inadequate when means can be relatively large but the mode of the output is 0. Such data is common when many data entries have 0 outputs and many also have much larger outputs, so the mode of output is 0 but the overall mean is not near 0. This problem is known as *zero-inflation*.

This problem considers handling zero-inflation with a two-part model called a *hurdle model*. One part is a binary model such as a logistic model for whether the output is zero or positive. Conditional on a positive output, the "hurdle is crossed" and the second part uses a truncated model that modifies an ordinary distribution by conditioning on a positive output. This model can handle both zero inflation and zero deflation.

Suppose that the first part of the process is governed by probabilities $p(y > 0 \mid x) = \pi$ and $p(y = 0 \mid x) = 1 - \pi$; and the second part depends on $\{y \in \mathbb{Z} \mid y > 0\}$ and follows a probability mass function $f(y \mid \mathbf{x})$ that is truncated-at-zero. The complete distribution is therefore:

$$
\begin{aligned}
P(y = 0 \mid x) &= 1 - \pi \\
P(y = j \mid x) &= \pi \frac{f(j \mid \mathbf{x})}{1 - f(0 \mid \mathbf{x})}, \ j = 1, 2, ...
\end{aligned}
$$

One choice of parameterization is to use a logistic regression model for π:

$$\pi = \sigma(\mathbf{x}^\top \mathbf{w}_1)$$

and use a Poisson GLM for f with mean parameters λ (see Murphy 9.3):

$$\lambda = \exp(\mathbf{x}^\top \mathbf{w}_2)$$

(a) Suppose we observe N data samples $\{(x_n, y_n)\}_{n=1}^N$. Write down the log-likelihood for the hurdle model assuming an unspecified mass function f. Give an maximum likelihood estimation approach for the specified parts of the model.

(b) Assume now that we select Poisson distribution for f. Show that the truncated-at-zero Poisson distribution (as used in the hurdle model) is a member of the exponential family. Give its the sufficient statistics, natural parameters and log-partition function.

(c) What is the mean and variance of a truncated Poisson model with mean parameter λ? If we observe n i.i.d. samples from a truncated Poisson distribution, what is the maximum likelihood estimate of λ? (Note: Give an equation which could be solved numerically to obtain the MLE.)

(d) Now assume that we using a hurdle model as a GLM with f as a Poisson distribution. Show that this is a valid GLM (exponential family for y), derive its log-likelihood, and give its sufficient statistics.

(a) $\mathbf{X} \in \mathbb{R}^{N\times m}$ with $\mathbf{x}_i \in \mathbb{R}^m$ as the ith row of $\mathbf{X}$ and $x_{ij} \in \mathbb{R}$ as the jth element of $\mathbf{x}_i$.

$\mathbf{Y} \in \mathbb{R}^N$ with Y_i as the ith element in $\mathbf{Y}$. Then $\mathbf{y}$ is an instance of $\mathbf{Y}$ and y_i is an instance of Y_i.

$\mathbf{w}_1 \in \mathbb{R}^m$ with w_{1j} as the jth element in $\mathbf{w}_1$.

$\mathbf{w}_2 \in \mathbb{R}^m$ with w_{2j} as the jth element in $\mathbf{w}_2$.

$\pi_i = \sigma(\mathbf{x}_i^T\mathbf{w}_1)\ \forall\ i \in \{1, ..., N\}$.

$\lambda_i = \exp(\mathbf{x}_i^T\mathbf{w}_2)\ \forall\ i \in \{1, ..., N\}$.

$\boldsymbol{\pi} \in \mathbb{R}^N$ is a vector with π_i as the ith element and $\boldsymbol{\lambda} \in \mathbb{R}^N$ is a vector with λ_i as the ith element.

f is an unspecified PDF.

$\mathbb{1}$ is an indicator function that takes the value 1 when its input is true.

$$P(\mathbf{Y}=\mathbf{y};\boldsymbol{\pi},\boldsymbol{\lambda}) = \prod_{i=1}^{N}(1-\pi_i)^{\mathbb{1}[y_i=0]}\left(\pi_i\frac{f(y_i;\lambda_i)}{1-f(0;\lambda_i)}\right)^{1-\mathbb{1}[y_i=0]}$$

$$\Rightarrow \log\big(P(\mathbf{Y}=\mathbf{y};\boldsymbol{\pi},\boldsymbol{\lambda})\big) = \sum_{i=1}^{N}\mathbb{1}[y_i=0]\log(1-\pi_i)+\big(1-\mathbb{1}[y_i=0]\big)\log\left(\frac{f(y_i;\lambda_i)}{1-f(0;\lambda_i)}\right)+\big(1-\mathbb{1}[y_i=0]\big)\log(\pi_i)$$

$$= \sum_{i=1}^{N}\log(\pi_i)+\mathbb{1}[y_i=0]\log\left(\frac{1-\pi_i}{\pi_i}\right)+\big(1-\mathbb{1}[y_i=0]\big)\log\left(\frac{f(y_i;\lambda_i)}{1-f(0;\lambda_i)}\right)$$

$$= \sum_{i=1}^{N}\log(\sigma(\mathbf{x}_i^T\mathbf{w}_1))+\mathbb{1}[y_i=0]\log\left(\frac{1-\sigma(\mathbf{x}_i^T\mathbf{w}_1)}{\sigma(\mathbf{x}_i^T\mathbf{w}_1)}\right)+\big(1-\mathbb{1}[y_i=0]\big)\log\left(\frac{f(y_i;\lambda_i)}{1-f(0;\lambda_i)}\right)$$

$$= \sum_{i=1}^{N}-\log\big(1+\exp(\mathbf{x}_i^T\mathbf{w}_1)\big)+\big(1-\mathbb{1}[y_i=0]\big)\mathbf{x}_i^T\mathbf{w}_1+\big(1-\mathbb{1}[y_i=0]\big)\log\left(\frac{f(y_i;\lambda_i)}{1-f(0;\lambda_i)}\right)$$

$$\frac{\partial\log\big(P(\mathbf{Y}=\mathbf{y};\boldsymbol{\pi},\boldsymbol{\lambda})\big)}{\partial w_{1j}} = \sum_{i=1}^{n}-\frac{x_{ij}\exp(\mathbf{x}_i^T\mathbf{w}_1)}{1+\exp(\mathbf{x}_i^T\mathbf{w}_1)}+\big(1-\mathbb{1}[y_i=0]\big)x_{ij}\ \forall\ j\in\{1,...,m\}$$

$$\Rightarrow \frac{\partial\log\big(P(\mathbf{Y}=\mathbf{y};\boldsymbol{\pi},\boldsymbol{\lambda})\big)}{\partial\mathbf{w}_1} = \left[\frac{\partial\log\big(P(\mathbf{Y}=\mathbf{y};\boldsymbol{\pi},\boldsymbol{\lambda})\big)}{\partial w_{11}},...,\frac{\partial\log\big(P(\mathbf{Y}=\mathbf{y};\boldsymbol{\pi},\boldsymbol{\lambda})\big)}{\partial w_{1m}}\right]^T$$

We want to find $\hat{\mathbf{w}}_1^{MLE}$. This will be the $\mathbf{w}_1$ that makes the gradient of the log-likelihood with respect to $\mathbf{w}_1$ equal to $\mathbf{0}$. To find this $\mathbf{w}_1$, we can use a gradient ascent method. For example, we could initialize $\mathbf{w}_1^{(0)}$ and then use:

$\mathbf{w}_1^{(i+1)} \leftarrow \mathbf{w}_1^{(i)} + \alpha\frac{\partial\log\big(P(\mathbf{Y}=\mathbf{y};\boldsymbol{\pi},\boldsymbol{\lambda})\big)}{\partial\mathbf{w}_1}\big(\mathbf{w}_1^{(i)}\big)$ until convergence ($\alpha \in \mathbb{R}$ is the learning rate).

(b) Let Y be truncated-at-zero Poisson distributed with parameter λ.

$$P(Y = y; \lambda) = \frac{\frac{\exp(-\lambda)\lambda^y}{y!}}{1 - \exp(-\lambda)}$$

$$= \exp\big(-\lambda + y\log(\lambda) - \log(y!) - \log(1 - \exp(-\lambda))\big)$$

$$= (y!)^{-1}\exp\Big(y\log(\lambda) - \big(\lambda + \log(1 - \exp(-\lambda))\big)\Big)$$

$$\theta = \log(\lambda)$$

$$A(\theta) = \exp(\theta) + \log\big(1 - \exp(-\exp(\theta))\big)$$

$$\phi(y) = y$$

$$h(y) = (y!)^{-1}$$

θ is the natural parameter, $A(\theta)$ is the log-partition function, $\phi(y)$ is the sufficient statistic, and $h(y)$ is the scaling constant. Because we can write the truncated-at-zero Poisson PMF can be written in the form $P(Y = y; \lambda) = h(y)\exp\big(\theta\phi(y) - A(\theta)\big)$, we know that the truncated-at-zero Poisson distribution is a member of the exponential family.

(c) Let $Y \sim \text{Pois}(\lambda)$ and Y_{trunc} be truncated-at-zero Poisson distributed with parameter λ.

$$E(Y_{trunc}) = \sum_{y=0}^{\infty} yP(Y_{trunc} = y)$$

$$\Rightarrow \big(1 - \exp(-\lambda)\big)E(Y_{trunc}) = \sum_{y=0}^{\infty} yP(Y = y) = \lambda$$

$$\Rightarrow E(Y_{trunc}) = \frac{\lambda}{1 - \exp(-\lambda)}$$

$$\text{Var}(Y_{trunc}) = E(Y_{trunc}^2) - \big[E(Y_{trunc})\big]^2$$

We'll need to solve for $E(Y_{trunc}^2)$.

$$E(Y_{trunc}^2) = \sum_{y=0}^{\infty} y^2P(Y_{trunc} = y)$$

$$\Rightarrow \big(1 - \exp(-\lambda)\big)E(Y_{trunc}^2) = \sum_{y=0}^{\infty} y^2P(Y = y) = \text{Var}(Y) + \big[E(Y)\big]^2 = \lambda + \lambda^2$$

$$\Rightarrow E(Y_{trunc}^2) = \frac{\lambda + \lambda^2}{1 - \exp(-\lambda)}$$

Plugging this and the expectation for Y_{trunc} in, we get:

$$\mathrm{Var}(Y_{trunc}) = \frac{\lambda + \lambda^2}{1 - \exp(-\lambda)} - \frac{\lambda^2}{\big(1 - \exp(-\lambda)\big)^2}$$

Now let Y_i follow a truncated-at-zero Poisson distribution with parameter $\lambda\ \forall\ i \in \{1, ..., n\}$ and let $\mathbf{Y} \in \mathbb{R}^n$ have Y_i as its ith element. Then $\mathbf{y}$ is an instance of $\mathbf{Y}$ and y_i is an instance of Y_i.

$$P(\mathbf{Y} = \mathbf{y}; \lambda) = \prod_{i=1}^{n} \frac{\frac{\exp(-\lambda)\lambda_i^y}{y_i!}}{1 - \exp(-\lambda)}$$

$$\Rightarrow \log\big(P(\mathbf{Y} = \mathbf{y}; \lambda)\big) = \sum_{i=1}^{n} -\lambda + y_i \log(\lambda) - \log(y_i!) - \log(1 - \exp(-\lambda)\big)$$

$$= -n\lambda - n\log\big(1 - \exp(-\lambda)\big) + n\bar{y}\log(\lambda) - \sum_{i=1}^{n} \log(y_i!)$$

$$\frac{d\log\big(P(\mathbf{Y} = \mathbf{y}; \lambda)\big)}{d\lambda} = -n + \frac{n}{1 - \exp(\lambda)} + \frac{n\bar{y}}{\lambda}$$

We want to find $\hat{\lambda}_{MLE}$. This will be the λ that makes the derivative of the log-likelihood with respect to λ equal to 0. To find this λ, we can use a gradient ascent method. For example, we could initialize $\lambda^{(0)}$ and then use:

$$\lambda^{(i+1)} \leftarrow \lambda^{(i)} + \alpha \frac{d\log\big(P(\mathbf{Y} = \mathbf{y}; \lambda)\big)}{d\lambda}\big(\lambda^{(i)}\big) \text{ until convergence } (\alpha \in \mathbb{R} \text{ is the learning rate}).$$

(d) $f(y_i; \lambda_i) = \dfrac{\exp(-\lambda_i)\lambda_i^{y_i}}{y_i!}$ with $\lambda_i = \exp(\mathbf{x}_i^T \mathbf{w}_2)\ \forall\ i \in \{1, ..., n\}$

The rest of the notation is the same as in part **(a)**.

$$P(Y_i = y_i; \pi_i, \lambda_i) = (1 - \pi_i)^{\mathbb{1}[y_i = 0]} \left(\pi_i \frac{\frac{\exp(-\lambda_i)\lambda_i^{y_i}}{y_i!}}{1 - \exp(-\lambda_i)}\right)^{1 - \mathbb{1}[y_i = 0]}$$

$$= \exp\Big(\mathbb{1}[y_i = 0]\log(1 - \pi_i) + \big(1 - \mathbb{1}[y_i = 0]\big)\log(\pi_i)\Big)\exp\Big(-\lambda_i\big(1 - \mathbb{1}[y_i = 0]\big) + y_i\big(1 - \mathbb{1}[y_i = 0]\big)\log(\lambda_i) - \big(1 - \mathbb{1}[y_i = 0]\big)\log(y_i!) - \big(1 - \mathbb{1}[y_i = 0]\big)\log\big(1 - \exp(-\lambda_i)\big)\Big)$$

$$= \exp\left(\mathbb{1}[y_i = 0]\log\left(\frac{1-\pi_i}{\pi_i}\right) + \log(\pi_i)\right)\exp\big(y_i\log(\lambda_i) - \lambda_i - \log(y_i!)\big)\exp\big(\lambda_i\mathbb{1}[y_i = 0] - y_i\mathbb{1}[y_i = 0]\log(\lambda_i) +$$
$$\mathbb{1}[y_i = 0]\log(y_i!)\big)\exp\left(\big(\mathbb{1}[y_i = 0] - 1\big)\Big(\log\big(\exp(\lambda_i) - 1\big) - \lambda_i\Big)\right)$$

$$= \exp\left(\mathbb{1}[y_i = 0]\log\left(\frac{1-\pi_i}{\pi_i}\right) + \log(\pi_i)\right)\exp\big(y_i\log(\lambda_i) - \lambda_i - \log(y_i!)\big)\exp\big(\lambda_i\mathbb{1}[y_i = 0] - y_i\mathbb{1}[y_i = 0]\log(\lambda_i) +$$
$$\mathbb{1}[y_i = 0]\log(y_i!)\big)\exp\left(\mathbb{1}[y_i = 0]\log\big(\exp(\lambda_i) - 1\big) - \lambda_i\mathbb{1}[y_i = 0] - \log\big(\exp(\lambda_i) - 1\big) + \lambda_i\right)$$

The terms $y_i\mathbb{1}[y_i = 0]\log(\lambda_i)$ and $\mathbb{1}[y_i = 0]\log(y_i!)$ will always be equal to 0, regardless of the value of y_i. Continuing, we have:

$$= (y_i!)^{-1}\exp\left(\mathbb{1}[y_i = 0]\left(\log\left(\frac{1-\pi_i}{\pi_i}\right) + \log\big(\exp(\lambda_i) - 1\big)\right) + y_i\log(\lambda_i) - \Big(\log\big(\exp(\lambda_i) - 1\big) - \log(\pi_i)\Big)\right)$$

$$= h(y_i)\exp\big(\boldsymbol{\theta}^T\boldsymbol{\phi}(y_i) - A(\boldsymbol{\theta})\big)$$

$$\boldsymbol{\theta} = [\theta_1,\, \theta_2]^T = \left[\log\left(\frac{1-\pi_i}{\pi_i}\right) + \log\big(\exp(\lambda_i) - 1\big),\, \log(\lambda_i)\right]^T$$

$$= \left[\log\left(\frac{1-\sigma(\mathbf{x}_i^T\mathbf{w}_1)}{\sigma(\mathbf{x}_i^T\mathbf{w}_1)}\right) + \log\Big(\exp\big(\exp(\mathbf{x}_i^T\mathbf{w}_2)\big) - 1\Big),\, \mathbf{x}_i^T\mathbf{w}_2\right]^T$$

$$= \left[\log\Big(\exp\big(\exp(\mathbf{x}_i^T\mathbf{w}_2)\big) - 1\Big) - \mathbf{x}_i^T\mathbf{w}_1,\, \mathbf{x}_i^T\mathbf{w}_2\right]^T$$

$$A(\boldsymbol{\theta}) = \log\Big(\exp(\theta_1) + \exp\big(\exp(\theta_2)\big) - 1\Big)$$

$$\boldsymbol{\phi}(y_i) = [\mathbb{1}[y_i = 0],\, y_i]$$

$$h(y_i) = (y_i!)^{-1}$$

$\boldsymbol{\theta}$ represents the the natural parameters, $A(\boldsymbol{\theta})$ is the log-partition function, $\boldsymbol{\phi}(y_i)$ represents the sufficient statistics, and $h(y_i)$ is the scaling constant. Because we can write the hurdle model PMF for Y_i in the exponential family form $P(Y_i = y_i; \pi_i, \lambda_i) = h(y_i)\exp\big(\boldsymbol{\theta}^T\boldsymbol{\phi}(y_i) - A(\boldsymbol{\theta})\big)$, we know that the hurdle model is a valid GLM.

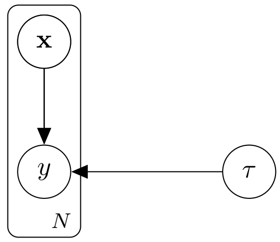
Now we need to derive the log-likelihood for the hurdle model GLM. Starting from the final log-likelihood expression in part **(a)**, we have:

$$\log\big(P(Y_i = y_i; \pi_i, \lambda_i)\big) = -\log\big(1 + \exp(\mathbf{x}_i^T\mathbf{w}_1)\big) + \big(1 - \mathbb{1}[y_i = 0]\big)\mathbf{x}_i^T\mathbf{w}_1 + \big(1 - \mathbb{1}[y_i = 0]\big)\log\left(\frac{\frac{\exp(-\lambda_i)\lambda_i^{y_i}}{y_i!}}{1 - \exp(-\lambda_i)}\right)$$
$$= -\log\big(1 + \exp(\mathbf{x}_i^T\mathbf{w}_1)\big) + \big(1 - \mathbb{1}[y_i = 0]\big)\mathbf{x}_i^T\mathbf{w}_1 + \big(1 - \mathbb{1}[y_i = 0]\big)\Big(y_i\log(\lambda_i) - \big(\lambda_i + \log\big(1 - \exp(-\lambda_i)\big)\big) - \log(y_i!)\Big)$$

$= -\log\big(1 + \exp(\mathbf{x}_i^T\mathbf{w}_1)\big) + \big(1 - \mathbb{1}[y_i = 0]\big)\mathbf{x}_i^T\mathbf{w}_1 + \big(1 - \mathbb{1}[y_i = 0]\big)\Big(y_i\mathbf{x}_i^T\mathbf{w}_2 - \big(\exp(\mathbf{x}_i^T\mathbf{w}_2) + \log\big(1 - \exp(-\exp(\mathbf{x}_i^T\mathbf{w}_2))\big)\big) - \log(y_i!)\Big)$

Problem 3 (Directed Graphical and Naive Bayes, 10pts)

To draw the DGMs for this problem, we recommend using the `tikzbayesnet` *library. For example the following is drawn in LaTeX:*



This problem focuses on modeling a joint distribution of random variables, $p(y, x_1, \dots, x_V)$, consisting of discrete variables. These variables represent a class label $y \in \{1, \dots, C\}$ and features $x_1 \dots, x_V$ each of each can take on a values $x_v \in \{0, 1\}$.

(a) Let $V = 4$. Use the chain rule to select any valid factorization of this joint distribution into univariate distributions. Draw the directed graphical model corresponding to this factorization.

(b) What is the sum of the sizes of the *conditional probability tables* associated with this graphical model. Can you reduce the order of magnitude of this value with a different DGM?

(c) Now consider a naive Bayes factorization of this model, given by,

$$p(y, x_1, \dots, x_V) \approx p(y) \prod_v p(x_v | y).$$

Draw a directed graphical model for this factorization. What is the size of the conditional probability tables required to fully express any factored distribution of this form?

(d) In class, we parameterized naive Bayes such that the class distribution is Categorical with a Dirichlet prior, and the class-conditional distributions are Bernoulli with a Beta prior. Extend the graphical model above to show the generative model of N data points and include the parameters and hyper-parameters as random variables.

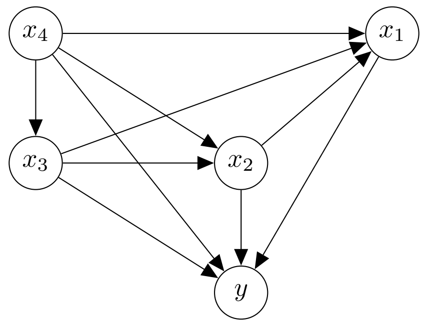
(e) Assuming the data obeys the naive Bayes assumptions, answer the following questions as true/false using your directed graphical model. Justify your answer.

- For a given example, features x_1 and x_2 are independent.
- The class labels y are always conditionally independent of the class-conditional parameters.
- Upon observing the class distribution parameters, the class labels are conditionally independent.
- Upon observing the class distribution parameters, the features are conditionally independent.
- Upon observing the class distribution hyper-parameters, the class labels are conditionally independent.

(f) For the next problem, we will utilize naive Bayes for a problem where each example has a *bag* or multiset of items. A bag is a set that may contain multiple instances of the same value. One approach is to ignore this property and use x_v as an indicator function for each item type. An alternative is to model x_v with sample space $\{0, \dots, D\}$, where D is the maximum times an item appears and to use a Dirichlet-Categorical for the class-conditional. Give one benefit and one drawback of this approach. Propose a third option for modeling this distribution.

(a) N is the number of data points.

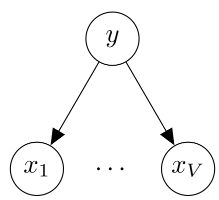
$$P(y, x_1, x_2, x_3, x_4) = P(y|x_1, x_2, x_3, x_4) \cdot P(x_1|x_2, x_3, x_4) \cdot P(x_2|x_3, x_4) \cdot P(x_3|x_4) \cdot P(x_4)$$



(b) The size of the conditional probability tables will be $C \cdot 2^4$ for $y|x_1, x_2, x_3, x_4$, 2^4 for $x_1|x_2, x_3, x_4$, 2^3 for $x_2|x_3, x_4$, 2^2 for $x_3|x_4$. Then the sum of the sizes of the conditional probability tables will be $C \cdot 2^4 + 2^4 + 2^3 + 2^2 = 16C + 28$. The size of the marginal probability table for x_4 is 2, so the sum of the sizes of all the probability tables will be $16C + 30$.

Without assuming independence or conditional independence, we cannot reduce the magnitude of this value with a different DGM. Any valid factorization of the joint PDF into univariate distributions where $y|x_1, x_2, x_3, x_4$ is included in the factorization will necessarily yield the same sized conditional probability tables if we do not make independence or conditional independence assumptions.

(c) $X_1|Y, ..., X_V|Y$ are independent. Then the DGM can be represented by the following graph:



Each conditional probability table will be size $2C$, so the sum of the sizes of the conditional probability tables will be $2CV$. The size of the marginal probability table for y is C, so the sum of the sizes of all the probability tables will be $2CV + C$.

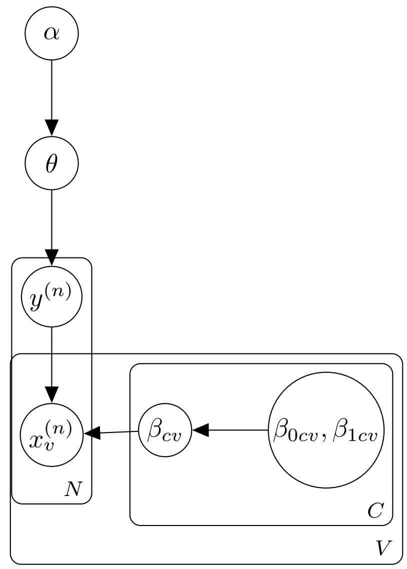
(d) $\theta \sim \text{Dir}(\alpha)$ with $\theta, \alpha \in \mathbb{R}^C$. This the prior for the distribution of class labels.

$Y^{(n)}|\theta \sim \text{Cat}(\theta)$ with $Y^{(n)} \in \mathbb{R}^C$ as a one hot encoded vector such that $Y_c^{(n)} = 1$ if the nth data point has class label c and 0 otherwise.

$\beta_{cv} \sim \text{Beta}(\beta_{0cv}, \beta_{1cv})$ with $\beta_{cv}, \beta_{0cv}, \beta_{1cv} \in \mathbb{R}$. This is the prior for the vth feature for data points that have class label c.

$X_v^{(n)}|\beta_{cv} \sim \text{Bern}\big(\beta_{cv}\big)$

The DGM can then be represented by the following graph:



Note that β_{cv} is used to generate $x_v^{(n)} \Leftrightarrow y_c^{(n)} = 1$. Otherwise it is ignored.

(e)

(1) False. The path from feature $x_1^{(n)}$ to $x_2^{(n)}$ through $y^{(n)}$ is not blocked.

(2) False. The path from the class labels to the class-conditional parameters through the features is not blocked when the features are in the evidence.

(3) True. The path from $y^{(i)}$ to $y^{(j)}$ through θ is blocked when θ is in the evidence.

(4) False. The path from $x_i^{(n)}$ to $x_j^{(n)}$ through $y^{(n)}$ is not blocked when the class distribution parameters are in the evidence.

(5) False. The path from $y^{(i)}$ to $y^{(j)}$ through θ is not blocked when α is in the evidence.

(f) A benefit would be that the model captures the fact that there can be multiple instances of each feature in a given example via the class-conditional distributions for each feature. We lose information about a given example by only allowing for binary features. A drawback is that the number of parameters required to fit the model is multiplied by something on the order of $\frac{D}{2}$. As a result, if we don't have a ton of data, we may not be able to achieve a generalizable model fit. A third option would be to model the entire distribution over features for each class using a Dirichlet-Multinomial. This would allow us to capture the fact that there can be multiple instances of each feature in a given example without requiring as many parameters to fit the model.

Problem 4 (Naive Bayes Implementation, 10pts)

You will now implement a naive Bayes classifier for for sentiment classification. For this problem you will use the IMDB Movie Reviews dataset which consists of positive and negative movie reviews . Here are two example reviews:

`there is no story! the plot is hopeless! a filmed based on a car with a`
`stuck accelerator, no brakes, and a stuck automatic transmission gear`
`lever cannot be good! ... i feel sorry for the actors ... poor script ...`
`heavily over-dramatized ... this film was nothing but annoying,`
`stay away from it!` [negative review]

`i had forgotten both how imaginative the images were, and how witty`
`the movie ... anyone interested in politics or history will love the movie's`
`offhand references - anyone interested in romance will be moved - this`
`one is superb.` [positive review]

As noted in the last problem, it is common to think of the input data as a bag/multiset. In text applications, sentences are often represented as a *bag-of-words*, containing how many times each word appears in the sentence. For example, consider two sentences:

- `We like programming. We like food.`
- `We like CS281.`

A vocabulary is constructed based on these two sentences:

`["We", "like", "programming", "food", "CS281"]`

Then the two sentences are represented as the number of occurrences of each word in the vocabulary (starting from position 1):

- `[0, 2, 2, 1, 1, 0]`
- `[0, 1, 1, 0, 0, 1]`

We have included a utility file `utils.py` that does this mapping. For these problems you can therefore treat text in this matrix representation.

- Implement a Naive Bayes classifier using a Bernoulli class-conditional with a Beta prior where each feature is an indicator that a word appears at least once in the bag.
- Implement a Naive Bayes classifier using a Categorical class-conditional with a Dirichlet prior. Here the features represent that count of each word in the bag.
- For both models, experiment with various settings for the priors. For the Dirichlet prior on the class, begin with $\boldsymbol{\alpha} = \mathbf{1}$ (Laplace Smoothing). Do the same for the class-conditional prior (be it Dirichlet or Beta). Keeping uniformity, vary the magnitude to .5 and smaller. If the classes are unbalanced in the dataset, does it help to use a larger α for the less-often occuring class? Optionally, choose class-conditional priors based on an outside text source. Validate your choices on the validation set, and report accuracy on the test set.
- (Optional) With the bag-of-words representation, would the model be able to capture phrases like "don't like"? An alternative to the bag-of-words model is known as the bag-of-bigrams model, where a bigram is two consecutive words in a sentence. Modify `utils.py` to include bigram features with either model and see if they increase accuracy.
- (Optional Reading) *Baselines and Bigrams: Simple, Good Sentiment and Topic Classification* `http://www.aclweb.org/anthology/P/P12/P12-2.pdf#page=118`

Test accuracy = 0.864
Class prior parameter magnitude = 1
Beta class-conditional prior parameter magnitude = 1

Test accuracy = 0.863
Class prior parameter magnitude = 1
Dirichlet class-conditional prior parameter magnitude = 0.4
Note that the maximum number of times that a feature could be drawn to was set to $D = 10$ (see **3(f)**)

The classes are not unbalanced in the dataset, so increasing the relative value of the prior parameter for the less often occurring class is not a relevant question here. If the classes were imbalanced, it still seems that we would want our prior to reflect the truth. That is, if our goal is to maximize accuracy, we would want our posterior mean for the distribution over classes to reflect the distribution over classes for the data we want to make predictions on, and choosing a prior that reflects the distribution over classes for data we want to make predictions on would help to give us such a posterior mean. If we have a goal for our model for which we care about false positives, false negatives, true positives, and true negatives, then we may want to choose a prior with non-uniformity for the distribution over classes to help us achieve that goal.

Problem 5 (Logistic Regression with Autograd, 15pts)

In the previous problem, you implemented a Naive Bayes classifier for sentiment classification on the IMDB Movie Reviews dataset. In this problem, you will apply logistic regression to the same task.

(a) ℓ_1-regularized logistic regression. Consider a model parameterized by $\mathbf{w}$:

$$\begin{aligned} p(\mathbf{w}) &= \frac{1}{2b}\exp(-\frac{\|\mathbf{w}\|_1}{b}) \\ p(y=1|\mathbf{x},\mathbf{w}) &= \sigma(\mathbf{w}^\top \mathbf{x}) \\ p(y=0|\mathbf{x},\mathbf{w}) &= 1-\sigma(\mathbf{w}^\top \mathbf{x}) \end{aligned}$$

where $\sigma(\cdot)$ is the sigmoid function. Note that we are imposing a Laplacian prior on $\mathbf{w}$, see Murphy, 2.4.4.

(i) Given a dataset $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^N$, derive the necessary gradient updates for MAP of $\mathbf{w}$.[a]

(ii) Show that for some constant λ, MAP inference of $\mathbf{w}$ is equivalent to minimizing

$$-\frac{1}{N}\sum_{i=1}^{N}\log p(y^{(i)}|\mathbf{x}^{(i)},\mathbf{w}) + \lambda\|\mathbf{w}\|_1$$

(b) Implementation using PyTorch automatic differentiation.[b]

(i) Using the bag-of-words feature representation from the previous question, train a logistic regression model using PyTorch autograd and `torch.nn`. Report test accuracy. Select regularization strength λ based on validation accuracy.

(ii) Which 5 words correspond to the largest weight indices, per class, in the learnt weight vectors? Which 5 words correspond to the least weight indices?

(iii) Study how sparsity (i.e percentage of zero elements in a vector) of the parameter vector changes with different values of λ. Again, tune λ on the validation set and report the test accuracies on the test set. Suggested values to try are $\{0, 0.001, 0.01, 0.1, 1\}$. You can treat parameters with $< 1e-4$ absolute values as zeros.

[a] You only need to consider the case where $\forall i, w_i \neq 0$. If $\exists i, w_i = 0$, we can use its subgradients instead.

[b] `https://github.com/harvard-ml-courses/cs281/blob/master/cs281-f17/sections/04/walkthrough.ipynb`.

(a)(i) $\mathbf{X} \in \mathbb{R}^{N\times m}$ with $\mathbf{x}_i \in \mathbb{R}^m$ as the ith row of $\mathbf{X}$ and $x_{ij} \in \mathbb{R}$ as the jth element of $\mathbf{x}_i$.

$\mathbf{Y} \in \mathbb{R}^N$ with Y_i as the ith element in $\mathbf{Y}$. Then $\mathbf{y}$ is an instance of $\mathbf{Y}$ and y_i is an instance of Y_i.

$\mathbf{W} \in \mathbb{R}^m$ with W_i as the ith element in $\mathbf{W}$. Then $\mathbf{w}$ is an instance of $\mathbf{W}$ and w_i is an instance of W_i.

$p(\mathbf{w}|\mathbf{y},\mathbf{X}) \propto p(\mathbf{y}|\mathbf{X},\mathbf{w})p(\mathbf{w}|\mathbf{X})$

$$= \prod_{i=1}^{N} p(y_i|\mathbf{x}_i,\mathbf{w})p(\mathbf{w}|\mathbf{x}_i) = p(\mathbf{w})^N \prod_{i=1}^{N} p(y_i|\mathbf{x}_i,\mathbf{w})$$

$$\Rightarrow \log\big(p(\mathbf{y}|\mathbf{X},\mathbf{w})p(\mathbf{w})\big) = \sum_{i=1}^{N}\log\big(p(y_i|\mathbf{x}_i,\mathbf{w})\big) + N\log\big(p(\mathbf{w})\big)$$

$$= \sum_{i=1}^{N} \log\big(p(y_i|\mathbf{x}_i, \mathbf{w})\big) - \frac{N||\mathbf{w}||_1}{b} - N\log(2b)$$

From problem **2(a)**, we know that this simplifies to:

$$= \sum_{i=1}^{N} -\log\big(1+\exp(\mathbf{x}_i^T\mathbf{w})\big) + y_i\mathbf{x}_i^T\mathbf{w} - \frac{N||\mathbf{w}||_1}{b} - N\log(2b)$$

$$\frac{\partial\big(p(\mathbf{w}|\mathbf{y},\mathbf{X})\big)}{\partial w_j} = \sum_{i=1}^{n} -\frac{x_{ij}\exp(\mathbf{x}_i^T\mathbf{w})}{1+\exp(\mathbf{x}_i^T\mathbf{w})} + y_i x_{ij} - \frac{Nw_j}{b|w_j|} \ \forall\ j \in \{1,...,m\} \text{ (case where no } w_i = 0)$$

$$\Rightarrow \frac{\partial\big(p(\mathbf{w}|\mathbf{y},\mathbf{X})\big)}{\partial \mathbf{w}} = \left[\frac{\partial\big(p(\mathbf{w}|\mathbf{y},\mathbf{X})\big)}{\partial w_1}, ..., \frac{\partial\big(p(\mathbf{w}|\mathbf{y},\mathbf{X})\big)}{\partial w_m}\right]^T$$

We want to find $\hat{\mathbf{w}}^{MAP}$. This will be the $\mathbf{w}$ that makes the gradient of the log-likelihood with respect to $\mathbf{w}$ equal to $\mathbf{0}$. To find this $\mathbf{w}$, we can use a coordinate ascent method.

(a)(ii) Note that $\hat{\mathbf{w}}^{MAP}$, which is the $\mathbf{w}$ maximizes $\sum_{i=1}^{N} \log\big(p(y_i|\mathbf{x}_i, \mathbf{w})\big) - \frac{N||\mathbf{w}||_1}{b} - N\log(2b)$ (we showed this in part **(a)(i)**) will be the same $\mathbf{w}$ that minimizes $-\frac{1}{N}\sum_{i=1}^{N} \log\big(p(y_i|\mathbf{x}_i, \mathbf{w})\big) + \frac{||\mathbf{w}||_1}{b}$ since we modified the first expression by negating it, multiplying by a constant, and deleting a term that did not depend on $\mathbf{w}$. Thus, we can find $\hat{\mathbf{w}}^{MAP}$ by minimizing the following with respect to $\mathbf{w}$:

$$-\frac{1}{N}\sum_{i=1}^{N} \log\big(p(y_i|\mathbf{x}_i, \mathbf{w})\big) + \lambda||\mathbf{w}||_1 \text{ where } \lambda = \frac{1}{b}$$

(b)(i) Validation accuracy when $\lambda = 0 : 0.858$
Validation accuracy when $\lambda = 0.001 : 0.819$
Validation accuracy when $\lambda = 0.01 : 0.738$
Validation accuracy when $\lambda = 0.1 : 0.557$
Validation accuracy when $\lambda = 1 : 0.534$
Test accuracy when $\lambda = 0 : 0.860$

(b)(ii)

Words with highest valued coefficients for predicting class 0: ['worst' 'bad' 'waste' 'poor' 'nothing']

Words with lowest valued coefficients for predicting class 0: ['great' 'best' 'excellent' 'well' 'loved']

Words with highest valued coefficients for predicting class 1: ['great' 'best' 'excellent' 'well' 'loved']

Words with lowest valued coefficients for predicting class 1: ['worst' 'bad' 'waste' 'poor' 'nothing']

(b)(iii) Validation accuracy when $\lambda = 0$: 0.858
Validation accuracy when $\lambda = 0.001$: 0.819
Validation accuracy when $\lambda = 0.01$: 0.738
Validation accuracy when $\lambda = 0.1$: 0.557
Validation accuracy when $\lambda = 1$: 0.534

Test accuracy when $\lambda = 0$: 0.860
Test accuracy when $\lambda = 0.001$: 0.815
Test accuracy when $\lambda = 0.01$: 0.739
Test accuracy when $\lambda = 0.1$: 0.554
Test accuracy when $\lambda = 1$: 0.522

Number of 0 valued parameters for class 0 ($\lambda = 0$): 0.0453
Number of 0 valued parameters for class 1 ($\lambda = 0$): 0.0462

Number of 0 valued parameters for class 0 ($\lambda = 0.001$): 0.9898
Number of 0 valued parameters for class 1 ($\lambda = 0.001$): 0.9898

Number of 0 valued parameters for class 0 ($\lambda = 0.01$): 0.995
Number of 0 valued parameters for class 1 ($\lambda = 0.01$): 0.996

Number of 0 valued parameters for class 0 ($\lambda = 0.1$): 0.0995
Number of 0 valued parameters for class 1 ($\lambda = 0.1$): 0.0982

Number of 0 valued parameters for class 0 ($\lambda = 1$): 0.0424
Number of 0 valued parameters for class 1 ($\lambda = 1$): 0.0415

When we increase λ from 0 to 0.001, the sparsity of the parameters increases drastically. It increases still for the increase in λ from 0.001 to 0.01. However, for the increase in λ from 0.01 to 0.1 we see a huge drop in sparsity. We then see another drop in sparsity when we increase λ from 0.1 to 1. The increase in sparsity along with increases in λ is what we would expect to see. I suspect that the reason that we see the drops in sparsity for the last two increases in λ is that stochastic gradient descent never achieves any sort of convergence to a minimum.

Problem 6 (Neural Networks, 5pts)

In the previous problem, we have implemented a Logistic Regression classifier using PyTorch. Logistic Regression can be seen as a 1-layer neural network. With PyTorch automatic differentiation, implementing a multi-layer neural network only requires incremental change to our logistic regression implementation.

(a) Implement a multi-layer neural network for IMDB classification and report accuracy on the test set. You are free to design the network structure (number of hidden units, activation function) and choose the optimization methods (SGD or ADAM, regularization or not, etc.).

(b) (Optional) Implement sentiment classification based on Convolutional Neural Networks. We recommend reading Yoon Kim (2014) *Convolution Neural Networks for Sentence Classification*. Note that in this part, you need to treat the text as a sequence of word vectors instead of bag-of-words. You can do this by forwarding `batch.text[0]` to `torch.nn.Embedding(vocab_size, embedding_dim)` after setting the weights using the pretrained vectors from `text_field.vocab.vectors`.

(a) I implemented a neural network for IMDB classification with one hidden layer of size $\frac{V}{1000}$ (V is the size of the vocabulary) that used a Tanh activation function. I used an SGD optimizer with a learning rate of 0.01. I used negative log likelihood loss as my loss function.

Test accuracy = 0.862