Runqiu Ye
Stanford CS299
Problem Set #3
07/29/2024

Problem Set #3: Deep Learning & Unsupervised Learning

Problem 1 A simple neural network
Let $X = \{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$ be dataset of m examples with 2 features. That is, $x^{(i)} \in \mathbb{R}^2$. Samples are classifed into 2 categorie with labels $y \in \{0, 1\}$, as shown in Figure 1. Want to perform binary classification using a simple neural networks with the architecture shown in Figure 2.
Two features x_1 and x_2, the three neurons in the hidden layer h_1, h_2, h_3, and the output neuron as o. Weight from x_i to h_j be $w_{i,j}^{[1]}$ for $i = 1, 2$ and $j = 1, 2, 3$, and weight from h_j to o be $w_j^{[2]}$. Finally, denote intercept weight for h_j as $w_{0,j}^{[1]}$ and the intercept weight for o as $w_0^{[2]}$. Use average squared loss instead of the usual negative log-likelihood:

$$l = \frac{1}{m} \sum_{i=1}^{m} (o^{(i)} - y^{(i)})^2.$$

(a) Suppose we use sigmoid function as activation function for h_1, h_2, h_3, and o. We have

$$h_1 = g(w_1^{[1]} x), \quad h_2 = g(w_2^{[1]} x), \quad h_3 = g(w_3^{[1]} x), \quad o = g(w^{[2]} h).$$

Hence,

$$\frac{\partial l}{\partial w_{1,2}^{[1]}} = \frac{1}{m} \sum_{i=1}^{m} 2(o^{(i)} - y^{(i)}) o^{(i)} (1 - o^{(i)}) w_2^{[2]} h_2^{(i)} (1 - h_2^{(i)}) x_1^{(i)},$$

where $h_2^{(i)} = g(w_{0,2}^{[1]} + w_{1,2}^{[1]} x_1^{(i)} + w_{2,2}^{[1]} x_2^{(i)})$ and g is the sigmoid function. Therefore, the gradient descent update to $w_{1,2}^{[1]}$, assuming learning rate α is

$$w_{1,2}^{[1]} := w_{1,2}^{[1]} - \frac{2\alpha}{m} \sum_{i=1}^{m} (o^{(i)} - y^{(i)}) o^{(i)} (1 - o^{(i)}) w_2^{[2]} h_2^{(i)} (1 - h_2^{(i)}) x_1^{(i)}$$

where $h_2^{(i)} = g(w_{0,2}^{[1]} + w_{1,2}^{[1]} x_1^{(i)} + w_{2,2}^{[1]} x_2^{(i)})$.

(b) Now, suppose the activation function for h_1, h_2, h_3, and o is the step function $f(x)$, defined as

$$f(x) = \begin{cases} 1, & (x \geq 0), \\ 0, & (x < 0). \end{cases}$$

Is it possible to have a set of weights that allow the neural network to classify this dataset with 100% accuracy? If so, provide a set of weights by completing `optimal_step_weights` wihin `src/p01_nn.py` and explain your reasoning for those weights. If not, please explain the reasoning.

There is a set of weights that allow the neural network to classify this dataset with 100% accuracy. For the step function activation, we have

$$\begin{aligned}
h_1 &= f(w_1^{[1]}x) = f(w_{0,1}^{[1]} + w_{1,1}^{[1]}x_1 + w_{2,1}^{[1]}x_2)\\
h_2 &= f(w_2^{[1]}x) = f(w_{0,2}^{[1]} + w_{1,2}^{[1]}x_1 + w_{2,3}^{[1]}x_2)\\
h_3 &= f(w_3^{[1]}x) = f(w_{0,3}^{[1]} + w_{1,3}^{[1]}x_1 + w_{2,3}^{[1]}x_2)\\
o &= f(w^{[2]}h) = f(w_0^{[2]} + w_1^{[2]}h_1 + w_2^{[2]}h_2 + w_3^{[2]}h_3).
\end{aligned}$$

Notice from Figure 1 that the label $y^{(i)} = 0$ if and only if $x^{(i)}$ satisfies

$$\begin{cases} x_2^{(i)} > 0.5,\\ x_1^{(i)} > 0.5,\\ x_1^{(i)} + x_2^{(i)} < 4.\end{cases}$$

Now, let

$$w_1^{[1]} = \begin{bmatrix}0.5\\0\\-1\end{bmatrix},\quad w_2^{[1]} = \begin{bmatrix}0.5\\-1\\0\end{bmatrix},\quad w_3^{[1]} = \begin{bmatrix}-4\\1\\1\end{bmatrix},\quad w_1^{[1]} = \begin{bmatrix}-0.5\\1\\1\\1\end{bmatrix}.$$

Under this set of weights, if all inequalities are satisifed, then $h_1 = h_2 = h_3 = 0$ and $w^{[2]}h = -0.5$. Otherwise, $h_1 + h_2 + h_3 \geq 1$ and $w^{[2]}h \geq -0.5$. Hence, this set of weights will capture all the conditions and allow the nerual network to classify this dataset with 100% accuracy.

(c) Let the activation function for h_1, h_2, h_3, be the linear function $f(x) = x$, and the activation function for o be the same step function as before. Is it possible to have a set of weights that allow the neural network to classify this dataset with 100% accuracy? If so, provide a set of weights by completing `optimal_linear_weights` wihin `src/p01_nn.py` and explain your reasoning for those weights. If not, please explain the reasoning.

There does not exist a set of weights that allow the neural network to classify this dataset with 100% accuracy. As the activation function for h_1, h_2, h_3 is the linear function, we have

$$\begin{aligned}
h_1 &= f(w_1^{[1]}x) = w_{0,1}^{[1]} + w_{1,1}^{[1]}x_1 + w_{2,1}^{[1]}x_2\\
h_2 &= f(w_2^{[1]}x) = w_{0,2}^{[1]} + w_{1,2}^{[1]}x_1 + w_{2,3}^{[1]}x_2\\
h_3 &= f(w_3^{[1]}x) = w_{0,3}^{[1]} + w_{1,3}^{[1]}x_1 + w_{2,3}^{[1]}x_2
\end{aligned}$$

Now,

$$\begin{aligned} w^{[2]}h &= w_0^{[2]} + w_1^{[2]}h_1 + w_2^{[2]}h_2 + w_3^{[2]}h_3 \\ &= (w_{0,1}^{[1]} + w_{0,2}^{[1]} + w_{0,3}^{[1]}) + (w_{1,1}^{[1]} + w_{1,2}^{[1]} + w_{1,3}^{[1]})x_1 + (w_{2,1}^{[1]} + w_{2,2}^{[1]} + w_{2,3}^{[1]})x_2. \end{aligned}$$

That is, the neural network degenerates into a linear model. However, the dataset is clearly not linearly separable. Thus, there does not exist a set of weights that allow the neural network to classify this dataset with 100% accuracy if the activation function for h_1, h_2, h_3 are the linear function $f(x) = x$.

$\blacksquare$

Problem 2 KL divergence and maximum likelihood

Kullback-Leibler (KL) divergence is a measure of how much one probability distribution is different from a second one. The *KL divergence* between two discrete-valued distribution $P(X)$, $Q(X)$ over the outcome space $\mathcal{X}$ is defined as follows:

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \frac{P(x)}{Q(x)}.$$

Assuem $P(x) > 0$ for all x. (One other standard thing to do is adopt the convention that $0 \log 0 = 0$.) Sometimes, we also write the KL divergence more explicityly as $D_{\mathrm{KL}}(P \parallel Q) = D_{\mathrm{KL}}(P(X) \parallel Q(X))$.

Background on Information Theory

The *entropy* of a probability distribution $P(X)$, defined as

$$H(P) = -\sum_{x \in \mathcal{X}} P(x) \log P(x).$$

measures how dispersed a probability distribution is. Notably, $\mathcal{N}(\mu, \sigma^2)$ has the highest entropy among all possible continuous distribution that has mean μ and variance σ^2. The entropy $H(P)$ is the best possible long term average bits per message (optimal) that can be achieved under probability distribution $P(X)$.

The *cross entropy* is defined as

$$H(P, Q) = -\sum_{x \in \mathcal{X}} P(x) \log Q(x).$$

The cross entropy $H(P, Q)$ is the long term average bits per message (suboptimal) that results under a distribution $P(X)$, by reusing an encoding scheme designed to be optimal for a scenario with probability distribution $Q(X)$.

Notice that

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log P(x) - \sum_{x \in \mathcal{X}} P(x) \log Q(x) = H(P, Q) - H(P).$$

If $H(P, Q) = 0$, then it necessarily means $P = Q$. In ML, it is common task to find distribution Q that is close to another distribution P. To achieve this, we optimize $D_{\mathrm{KL}}(P \parallel Q)$. Later we will see that Maximum Likelihood Estimation turns out to be equivalent minimizing KL divergence between the training data and the model.

(a) **Nonnegativity.** Prove that

$$D_{\mathrm{KL}}(P \parallel Q) \geq 0$$

and $D_{\mathrm{KL}}(P \parallel Q) = 0$ if an only if $P = Q$.

Hint: Use Jensen's inequality.

Proof. By definition,

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x\in\mathcal{X}} P(x)\log\frac{P(x)}{Q(x)} = -\sum_{x\in\mathcal{X}} P(x)\log\frac{Q(x)}{P(x)}.$$

Since $-\log x$ is strictly convex, by Jensen's inequality, we have

$$D_{\mathrm{KL}}(P \parallel Q) = -\sum_{x\in\mathcal{X}} P(x)\log\frac{Q(x)}{P(x)} \geq -\log\sum_{x\in\mathcal{X}} P(x)\frac{Q(x)}{P(x)} = 0.$$

When the equality holds,

$$\log\frac{Q(x)}{P(x)} = 0$$

with probability 1. That is, $Q = P$ with probability 1. This completes the proof. □

(b) **Chain rule for KL divergence.** The KL divergence between 2 conditional distributions $P(X \mid Y)$, $Q(X \mid Y)$ is defined as follows:

$$D_{\mathrm{KL}}(P(X \mid Y) \parallel Q(X \mid Y)) = \sum_y P(y)\left(\sum_x P(x \mid y)\log\frac{P(x \mid y)}{Q(x \mid y)}\right).$$

This can be thought of as the expected KL divergence between the corresponding conditional distributions on x. That is, between $P(X \mid Y = y)$ and $Q(X \mid Y = y)$, where the expectation is taken over the random y.

Prove the following chain rule for KL divergence:

$$D_{\mathrm{KL}}(P(X,Y) \parallel Q(X,Y)) = D_{\mathrm{KL}}(P(X) \parallel Q(X)) + D_{\mathrm{KL}}(P(Y \mid X) \parallel Q(Y \mid X)).$$

Proof.

$$\begin{aligned}
\text{LHS} &= \sum_x\sum_y P(x,y)\log\frac{P(x,y)}{Q(x,y)} \\
&= \sum_x\sum_y P(y \mid x)P(x)\left[\log\frac{P(y \mid x)}{Q(y \mid x)} + \log\frac{P(x)}{Q(x)}\right] \\
&= \sum_x\sum_y P(y \mid x)P(x)\log\frac{P(y \mid x)}{Q(y \mid x)} + \sum_x P(x)\log\frac{P(x)}{Q(x)}\sum_y P(y \mid x) \\
&= \sum_x\sum_y P(y \mid x)P(x)\log\frac{P(y \mid x)}{Q(y \mid x)} + \sum_x P(x)\log\frac{P(x)}{Q(x)} \\
&= D_{\mathrm{KL}}(P(X) \parallel Q(X)) + D_{\mathrm{KL}}(P(Y \mid X) \parallel Q(Y \mid X)) \\
&= \text{RHS}.
\end{aligned}$$

□

(c) **KL and maximum likelihood.** Consider density estimation problem and suppose we are given training set $\{x^{(i)}\}_{i=1}^m$. Let the empirical distribution be $\hat{P}(x) = \frac{1}{m}\sum_{i=1}^m \mathbb{I}\{x^{(i)} = x\}$. ($\hat{P}$ is just the uniform distribution over the training set; i.e., sampling from the empirical distribution is the same as picking a random example from the training set.)

Suppose we have a family of distributions P_θ parametrized by θ. Prove that finding the maximum likelihood estimates for the parameter θ is equivalent to finding P_θ with minimal KL divergence from $\hat{P}$. That is, prove that

$$\operatorname*{argmin}_\theta D_{\mathrm{KL}}(\hat{P} \parallel P_\theta) = \operatorname*{argmax}_\theta \sum_{i=1}^m \log P_\theta(x^{(i)}).$$

Proof. Notice that $\hat{P}$ is the uniform distribution over the training set, thus $\hat{P}(x^{(i)}) = \frac{1}{m}$ for $i = 1, \ldots, m$. It follows that

$$D_{\mathrm{KL}}(\hat{P} \parallel P_\theta) = \sum_x \hat{P}(x) \log \frac{\hat{P}(x)}{P_\theta(x)} = -\log m - \frac{1}{m}\sum_{i=1}^m \log P_\theta(x^{(i)}).$$

Hence,

$$\operatorname*{argmin}_\theta D_{\mathrm{KL}}(\hat{P} \parallel P_\theta) = \operatorname*{argmax}_\theta \sum_{i=1}^m \log P_\theta(x^{(i)}),$$

as desired. □

■

Remark: Consider the relationship between parts (b-c) and multi-variate Bernoulli Naive bayes parameter estimation. In Naive Bayes model we assumed P_θ is the following form: $P_\theta(x, y) = p(y)\prod_{i=1}^n p(x_i \mid y)$. By the chain rule for KL divergence, we therefore have

$$D_{\mathrm{KL}}(\hat{P} \parallel P_\theta) = D_{\mathrm{KL}}(\hat{P}(y) \parallel p(y)) + \sum_{i=1}^n D_{\mathrm{KL}}(\hat{P}(x_i \mid y) \parallel p(x_i \mid y)).$$

This shows that finding the maximum likelihood/minimum KL divergence estimates of the parameters decomposes into $2n + 1$ independent optimization problems: One for the class priors $p(y)$, and one for each conditional distributions $p(x_i \mid y)$ for each feature x_i given each of the two possible labels for y. Specifically, finding the maximum likelihood estimates for each of these problems individually results in also maximizing the likelihood of the joint distribution. This similarly applies to bayesian networks. △

Problem 3 KL divergence, Fisher Information, and the Natural Gradient
KL divergence between the two distributions is an asymetric measure of how different two distributions are. Consider two distributions over the same space given by densities $p(x)$, $q(x)$. The KL divergence between two continuous distributions is defined as

$$\begin{aligned} D_{\mathrm{KL}}(p \parallel q) &= \int p(x) \log \frac{p(x)}{q(x)} dx \\ &= \mathop{\mathbb{E}}_{x \sim p(x)}[\log p(x)] - \mathop{\mathbb{E}}_{x \sim p(x)}[\log q(x)]. \end{aligned}$$

A nice property of KL divergence is that it is invariant to parametrization. This means, KL divergence evaluates to the same value no matter how we parametrize the distribution P and Q. For example, if P and Q are in exponential family, the KL divergence between them is the same whether we are using natural parameters, natural parameters, or canonical parameters, or any arbitrary parametrization.
Now consider the problem of fitting model parameters using gradient descent. While KL divergence is invariant to parametrization, the gradient w.r.t the model parameters gradient is *invariant to parametrization.* We need to use *natural gradient.* This will make the optimization process invariant to the parametrization.
We will construct and derive the natural gradient update rule. Along the way, we will introduce *score function* and *Fisher Information.* Finally, we will see how this new natural gradient based optimization is actually equivalent to Newton's method for Generalized Linear Models.
Let the distribution of a random variable Y parametrized by $\theta \in \mathbb{R}^n$ be $p(y; \theta)$.

(a) **Score function.** The *score function* with $p(y;\theta)$ is defined as $\nabla_\theta \log p(y;\theta)$, which signifies the sensitivity of the likelihood function with respect to the parameters.

Show that the expected value of the score is 0.

Proof. The expected value of the score

$$\begin{aligned} \mathop{\mathbb{E}}_{y \sim p(y;\theta)}[[\nabla_{\theta'} \log p(y;\theta)]_{\theta'=\theta}] &= \int p(y;\theta)[\nabla_{\theta'} \log p(y;\theta)]_{\theta'=\theta} dy \\ &= \int p(y;\theta) \frac{1}{p(y;\theta)} [\nabla_{\theta'} p(y;\theta)]_{\theta'=\theta} dy \\ &= \left[\nabla_{\theta'} \int p(y;\theta) dy\right]_{\theta'=\theta} \\ &= 0 \end{aligned}$$

□

(b) **Fisher information.** *Fisher information* is defined as the covariance matrix of the score function,

$$\mathcal{I}(\theta) = \operatorname{cov}_{y\sim p(y;\theta)} \left[\nabla_{\theta'} \log p(y;\theta')\right]_{\theta'=\theta}.$$

Intuitively, the Fisher information represents the amount of information that a random variable Y carries about a parameter θ of interest. Show that the Fisher information can be equivalently given by

$$\mathcal{I}(\theta) = \mathop{\mathbb{E}}_{y\sim p(y;\theta)} \left[\left[\nabla_{\theta'} \log p(y;\theta')\nabla_{\theta'} \log p(y;\theta')^T\right]_{\theta'=\theta}\right].$$

Note that the fisher information is a funciton of the parameter. The parameter is both a) the parameter value at which the score function is evaluated, and b) the parameter of the distribution with respect to which the expectation and variance is calculated.

Proof. Since $\mathbb{E}_{y\sim p(y;\theta)}[[\nabla_{\theta'} \log p(y;\theta)]_{\theta'=\theta}] = 0$, we have

$$\operatorname{cov}_{y\sim p(y;\theta)} \left[\nabla_{\theta'} \log p(y;\theta')\right]_{\theta'=\theta} = \mathop{\mathbb{E}}_{y\sim p(y;\theta)} \left[\left[\nabla_{\theta'} \log p(y;\theta')\nabla_{\theta'} \log p(y;\theta')^T\right]_{\theta'=\theta}\right]$$

by the definition of covariance. This completes the proof. □

(c) **Fisher information (alternate form).** It turns out that Fisher information can not only be defined as the covariance of the score functino, but in most situations it can also be represented as the expected negative Hessian of the log-likelihood. Show that

$$\mathop{\mathbb{E}}_{y\sim p(y;\theta)} \left[\left[-\nabla^2_{\theta'} \log p(y;\theta')\right]_{\theta'=\theta}\right] = \mathcal{I}(\theta).$$

Proof. From (b), we know that

$$\begin{aligned}\mathcal{I}_{ij}(\theta) &= \mathbb{E}\left[\frac{\partial}{\partial\theta_i}\log p(y;\theta')\frac{\partial}{\partial\theta_j}\log p(y;\theta')\right]\\ &= \mathbb{E}\left[\frac{1}{p(y;\theta')^2}\frac{\partial}{\partial\theta_i}p(y;\theta')\frac{\partial}{\partial\theta_j}p(y;\theta')\right].\end{aligned}$$

Also, for the left hand side of the expression we need to prove, we have

$$\begin{aligned}\text{LHS} &= \mathbb{E}\left[-\frac{\partial}{\partial\theta_i}\frac{\partial}{\partial\theta_j}\nabla^2_{\theta'}p(y;\theta')\right]\\ &= \mathbb{E}\left[\frac{\partial}{\partial\theta_i}\frac{1}{p(y;\theta')}\frac{\partial}{\partial\theta_i}p(y;\theta')\right]\\ &= \mathbb{E}\left[-\frac{1}{p(y;\theta')}\frac{\partial}{\partial\theta_i}p(y;\theta')\frac{\partial}{\partial\theta_j}p(y;\theta') + \frac{1}{p(y;\theta')}\frac{\partial}{\partial\theta_i}\frac{\partial}{\partial\theta_j}p(y;\theta')\right]\end{aligned}$$

For the second term, we have

$$\begin{aligned}
\mathbb{E}\left[\frac{1}{p(y;\theta')}\frac{\partial}{\partial\theta_i}\frac{\partial}{\partial\theta_j}p(y;\theta')\right] &= \int \frac{\partial}{\partial\theta_i}\frac{\partial}{\partial\theta_j}p(y;\theta')dy \\
&= \frac{\partial}{\partial\theta_i}\frac{\partial}{\partial\theta_j}\int p(y;\theta')dy \\
&= 0.
\end{aligned}$$

Hence,

$$\mathop{\mathbb{E}}_{y\sim p(y;\theta)}\left[\left[-\nabla^2_{\theta'}\log p(y;\theta')\right]_{\theta'=\theta}\right] = \mathbb{E}\left[-\frac{1}{p(y;\theta')}\frac{\partial}{\partial\theta_i}p(y;\theta')\frac{\partial}{\partial\theta_j}p(y;\theta')\right] = \mathcal{I}(\theta).$$

This completes the proof. □

Remark. This shows that the expected curvature of the log-likelihood function is also equal to the Fisher information matrix. If the curvature of the log-likelihood is steep, this generally means you need fewer number of data samples to estimate that parameter well, and vice versa. The fisher information matrix associated with a statistical model parameterized by θ is extremely important in determining how a model behaves as a function of the number of traning set examples. △

(d) **Approximatin D_{KL} with Fisher information.** We are interested in the set of all distributions that are at a small fixed D_{KL} distance away from the current distribution. To calculate KL divergence between $p(y;\theta)$ and $p(y;\theta+d)$, we approximate with Fisher information at θ. Show that

$$D_{\mathrm{KL}}(p_\theta \parallel p_{\theta+d}) \approx \frac{1}{2}d^T\mathcal{I}(\theta)d.$$

Proof. Towards Taylor expansion, we have

$$\begin{aligned}
&D_{\mathrm{KL}}(p_\theta \parallel p_\theta) = 0 \\
&\nabla_{\theta'}D_{\mathrm{KL}}(p_\theta \parallel p_{\theta'}) = \nabla_{\theta'}\mathbb{E}[\log p_\theta] - \mathbb{E}[\log p_{\theta'}] = 0 \\
&\nabla^2_{\theta'}D_{\mathrm{KL}}(p_\theta \parallel p_{\theta'}) = \nabla^2_{\theta'}\mathbb{E}[\log p_\theta] - \mathbb{E}[\log p_{\theta'}] = \mathcal{I}(\theta).
\end{aligned}$$

Hence,

$$D_{\mathrm{KL}}(p_\theta \parallel p_{\tilde{\theta}}) \approx \frac{1}{2}d^T\mathcal{I}(\theta)d.$$

□

(e) **Natural gradient.** Want to maximize the log-likelihood by moving only by a fixed D_{KL} distance from the current position. Now set up the constrained optimization problem that will yield the natural gradient update d. Let the log-likelihood objective be $\ell(\theta) = \log p(y;\theta)$. Let the D_{KL} distance we want to move by be some small positive constant c. The natural gradient update d^* is

$$d^* = \operatorname*{argmax}_d \ell(\theta + d) \text{ subject to } D_{\text{KL}}(p_\theta \parallel p_{\theta+d}) = c.$$

In order to solve this, use Taylor expansion and Lagrangian multipliers.

For the optimization problem, consider the Lagrangian

$$\begin{aligned} L(d,\lambda) &= \ell(\theta+d) - \lambda(D_{\text{KL}}(p_\theta \parallel p_{\theta+d}) - c) \\ &= \log p(y;\theta+d) - \lambda(D_{\text{KL}}(p_\theta \parallel p_{\theta+d}) - c) \\ &= \log p(y;\theta) + d^T \frac{\nabla_\theta p(y;\theta)}{p(y;\theta)} - \lambda\Big(\frac{1}{2} d^T \mathcal{I}(\theta) d - c\Big). \end{aligned}$$

Set

$$\begin{cases} \nabla_d L(d,\lambda) = 0, \\ \nabla_\lambda L(d,\lambda) = 0. \end{cases} \implies \begin{cases} \frac{\nabla_\theta p(y;\theta)}{p(y;\theta)} = \lambda \mathcal{I}(\theta) d, \\ \frac{1}{2} d^T \mathcal{I}(\theta) d = c. \end{cases}$$

This gives

$$d = \sqrt{\frac{2c}{\nabla_\theta p(y;\theta)^T (\mathcal{I}^{-1})^T \nabla_\theta p(y;\theta)}} \mathcal{I}^{-1} \nabla_\theta p(y;\theta).$$

(f) **Relation to Newton's Method.** Show that the direction of update of Newton's method, and the direction of natural gradient, are exactly the same for GLMs. Refer to results in problem set 1 question 4. For natural gradient, it is sufficient to use $\tilde{d}$, the unscaled natural gradient.

Proof. Recall that for GLMs, we have

$$p(y;\eta) = b(y)\exp\big(\eta^T T(y) - a(\eta)\big).$$

It follows that

$$\begin{aligned} \log p(y;\eta) &= \log b(y) + \eta^T T(y) - a(\eta), \\ \nabla_\eta \log p(y;\eta) &= T(y) - \nabla_\eta a(\eta), \\ \nabla_\eta^2 \log p(y;\eta) &= -\nabla_\eta^2 a(\eta). \end{aligned}$$

When using Newton's method to update, we have

$$\begin{aligned} \eta &:= \eta - \big(\nabla_\eta^2 \log p(y;\eta)\big)^{-1} \nabla_\eta \log p(y;\eta) \\ &= \eta + \frac{1}{p(y;\eta)} \big(\nabla_\eta^2 a(\eta)\big)^{-1} \nabla_\eta p(y;\eta). \end{aligned}$$

Compare this the natural gradient

$$d = \frac{1}{\lambda p(y;\eta)} \mathcal{I}^{-1} \nabla_\eta p(y;\eta).$$

Recall that Fisher information matrix is the negative expected value of Hessian, and notice that the Hessian here is independent of y. Hence,

$$\mathcal{I} = \mathop{\mathbb{E}}_{y\sim p(y;\theta)}\left[-\nabla_\eta^2 \log p(y;\eta)\right] = \mathop{\mathbb{E}}_{y\sim p(y;\theta)}\left[\nabla_\eta^2 a(\eta)\right] = \nabla_\eta^2 a(\eta).$$

From this, we can clearly see that the direction of update of Newton's method and the direction of natural gradient are exactly the same for GLMs. This completes the proof. □

■

Problem 4 Semi-supervised EM

Expectation maximization (EM) is a classical algorithm for unsupervised learning. In this problem we explore one ways in which EM can be adapted to semi-supervised setting, where we have some labelled examples along with unlabelled examples.

In unsupervised learning, we have $\{x^{(i)}\}_{i=1}^m$ unlabelled examples and we wish to learn $p(x, z; \theta)$, but $z^{(i)}$ are not observed. EM allow us to maximize the intractable $p(x;\theta)$ indirectly by iteratively performing the E-step and M-step, each time maximizing a tractable lower bound. The objective

$$\ell_{\text{unsup}}(\theta) = \sum_{i=1}^m \log p(x^{(i)};\theta) = \sum_{i=1}^m \log \sum_{z^{(i)}} p(x^{(i)}, z^{(i)};\theta).$$

Now, extend EM to semi-supervised setting. Suppose *additional* $\tilde{m}$ labelled examples $\{(\tilde{x}^{(i)}, \tilde{z}^{(i)})\}_{i=1}^{\tilde{m}}$ where x and z are both observed. Want to simultaneously maximize the marginal likelihood of the parameters using the unlabelled examples, and full likelihood of the parameters using the labelled examples, by optimizing their weighted sum with some hyperparameter α. More concretely, the semi-supervised objective $\ell_{\text{semi-sup}}$ can be written as

$$\ell_{\text{sup}}(\theta) = \sum_{i=1}^{\tilde{m}} \log p(\tilde{x}^{(i)}, \tilde{z}^{(i)};\theta),$$

$$\ell_{\text{semi-sup}}(\theta) = \ell_{\text{unsup}} + \alpha\ell_{\text{sup}}.$$

First, we derive the EM steps for semi-supervised learning.

- **E-step (semi-supervised)**

  For $i = 1, 2, \ldots, m$, set
  $$Q_i^{(t)}(z^{(i)}) = p(z^{(i)} \mid x^{(i)};\theta^{(t)}).$$

- **M-step (semi-supervised)**

$$\theta^{(t+1)} = \operatorname*{argmax}_\theta \left[\sum_{i=1}^m \left(\sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)};\theta)}{Q_i(z^{(i)})}\right) + \alpha\left(\sum_{i=1}^{\tilde{m}} \log p(\tilde{x}^{(i)}, \tilde{z}^{(i)};\theta)\right)\right].$$

(a) **Convergence.** First show that this algorithm converges. It suffices to show that the objective function $\ell_{\text{semi-sup}}(\theta)$ with each iteration of E-step and M-step. That is, we need to show that $\ell_{\text{semi-sup}}(\theta^{(t+1)}) \geq \ell_{\text{semi-sup}}(\theta^{(t)})$

Proof. For $\theta^{(t)}$ and the $Q_i^{(t)}$ chosen, it is guaranteed that

$$\begin{aligned}\ell_{\text{semi-sup}}(\theta^{(t)}) &= \sum_{i=1}^{m} \log \sum_{z^{(i)}} p(x^{(i)}, z^{(i)}; \theta^{(t)}) + \alpha \sum_{i=1}^{\tilde{m}} \log p(\tilde{x}^{(i)}, \tilde{z}^{(i)}; \theta^{(t)}) \\ &= \sum_{i=1}^{m} \sum_{z^{(i)}} Q_i^{(t)}(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \theta^{(t)})}{Q_i^{(t)}(z^{(i)})} + \alpha \sum_{i=1}^{\tilde{m}} \log p(\tilde{x}^{(i)}, \tilde{z}^{(i)}; \theta^{(t)}).\end{aligned}$$

Since $Q_i^{(t)}$ is a probability measure, for $\theta^{(t+1)}$, we also have

$$\begin{aligned}\ell_{\text{semi-sup}}(\theta^{(t+1)}) &= \sum_{i=1}^{m} \log \sum_{z^{(i)}} p(x^{(i)}, z^{(i)}; \theta^{(t+1)}) + \alpha \sum_{i=1}^{\tilde{m}} \log p(\tilde{x}^{(i)}, \tilde{z}^{(i)}; \theta^{(t+1)}) \\ &\geq \sum_{i=1}^{m} \sum_{z^{(i)}} Q_i^{(t)}(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \theta^{(t+1)})}{Q_i^{(t)}(z^{(i)})} + \alpha \sum_{i=1}^{\tilde{m}} \log p(\tilde{x}^{(i)}, \tilde{z}^{(i)}; \theta^{(t+1)})\end{aligned}$$

by Jensen's inequality.

As $\theta^{(t+1)}$ maximizes the expression in M-step, we have

$$\begin{aligned}\ell_{\text{semi-sup}}(\theta^{(t+1)}) &\geq \sum_{i=1}^{m} \sum_{z^{(i)}} Q_i^{(t)}(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \theta^{(t+1)})}{Q_i^{(t)}(z^{(i)})} + \alpha \sum_{i=1}^{\tilde{m}} \log p(\tilde{x}^{(i)}, \tilde{z}^{(i)}; \theta^{(t+1)}) \\ &\geq \sum_{i=1}^{m} \sum_{z^{(i)}} Q_i^{(t)}(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \theta^{(t)})}{Q_i^{(t)}(z^{(i)})} + \alpha \sum_{i=1}^{\tilde{m}} \log p(\tilde{x}^{(i)}, \tilde{z}^{(i)}; \theta^{(t)}) \\ &= \ell_{\text{semi-sup}}(\theta^{(t)}).\end{aligned}$$

This completes the proof. □

Semi-supervised GMM

Now we revisit the Gaussian Mixture Model (GMM) to apply our semi-supervised EM algorithm. Consider a schenario where data is generated from k Gaussian distributions with unknown means $\mu_j \in \mathbb{R}^d$ and covariances $\Sigma_j \in \mathbb{S}_+^d$ where $j = 1, \ldots, k$. We have m data points $x^{(i)} \in \mathbb{R}^d$ and each has a corresponding latent $z^{(i)} \in \{1, \ldots, k\}$ indicating which distribution $x^{(i)}$ belongs to. Specifically, $z^{(i)} \sim \text{Multinomial}(\phi)$ and $x^{(i)} \mid z^{(i)} \sim \mathcal{N}(\mu_{z^{(i)}}, \Sigma_{z^{(i)}})$ i.i.d. We also have additional $\tilde{m}$ points $\tilde{x}^{(i)} \in \mathbb{R}^d$ and an associated *observed* variable $\tilde{z}^{(i)} \in \{1, \ldots, k\}$. As before, we assume $\tilde{x}^{(i)} \mid \tilde{z}^{(i)} \sim \mathcal{N}(\mu_{\tilde{z}^{(i)}}, \Sigma_{\tilde{z}^{(i)}})$ i.i.d.

(b) **Semi-supervised E-step.** In E-step, $z^{(i)}$ need to be re-estimated. For this specific model, we have

$$\begin{aligned}
w_j^{(i)} &= p(z^{(i)} = j \mid x^{(i)}) \\
&= \frac{p(x^{(i)} \mid z^{(i)} = j)p(z^{(i)} = j)}{\sum_l p(x^{(i)} \mid z^{(i)} = l)p(z^{(i)} = l)} \\
&= \frac{\frac{1}{|\Sigma_j|^{1/2}} \exp\big(-\frac{1}{2}(x^{(i)} - \mu_j)^T \Sigma_j^{-1} (x^{(i)} - \mu_j)\big)\phi_j}{\sum_l \frac{1}{|\Sigma_l|^{1/2}} \exp\big(-\frac{1}{2}(x^{(i)} - \mu_l)^T \Sigma_l^{-1} (x^{(i)} - \mu_l)\big)\phi_l}.
\end{aligned}$$

(c) **Semi-supervised M-step.** The model parameters μ, Σ, ϕ need to be re-estimated in the M-step. For this specific model, we have the optimization problem

$$\operatorname*{argmax}_{\mu,\Sigma,\phi} \ell_{\text{semi-sup}} = \operatorname*{argmax}_{\mu,\Sigma,\phi} \ell_{\text{unsup}} + \alpha \ell_{\text{sup}},$$

where

$$\begin{aligned}
\ell_{\text{unsup}} &= \sum_{i=1}^{m} \sum_{z^{(i)}} w_j^{(i)} \log \frac{\exp\big(-\frac{1}{2}(x^{(i)} - \mu_j)^T \Sigma_j^{-1} (x^{(i)} - \mu_j)\big)\phi_j}{(2\pi)^{d/2}|\Sigma_j|^{1/2} w_j^{(i)}}, \\
\ell_{\text{sup}} &= \sum_{i=1}^{\tilde{m}} \log \frac{1}{(2\pi)^{d/2}|\Sigma_{\tilde{z}^{(i)}}|^{1/2}} \exp\bigg(-\frac{1}{2}(\tilde{x}^{(i)} - \mu_{\tilde{z}^{(i)}})^T \Sigma_{\tilde{z}^{(i)}}^{-1} (\tilde{x}^{(i)} - \mu_{tzz})\bigg)\phi_{\tilde{z}^{(i)}}.
\end{aligned}$$

To get a closed form expression, we use Lagrangian multipliers and construct Lagrangian

$$L(\mu, \Sigma, \phi, \lambda) = \ell_{\text{unsup}} + \alpha \ell_{\text{sup}} + \lambda(\Sigma_j \phi_j - 1).$$

For matrix derivatvies, we have

$$\begin{aligned}
\nabla_A |A| &= \frac{1}{|A|} A^{-T}, \\
\nabla_A x^T A^{-1} x &= -A^{-1} x x^T A^{-1}.
\end{aligned}$$

Notice that the covariance matrices Σ_l are symmetric, so $\Sigma_l^{-T} = \Sigma^{-1}$. It follows that

$$\begin{aligned}
\nabla_{\mu_l} \ell_{\text{semi-sup}} &= \sum_{i=1}^{m} w_l^{(i)} \Sigma_l^{-1} (x^{(i)} - \mu_l) + \alpha \sum_{i=1}^{\tilde{m}} \mathbb{I}\{\tilde{z}^{(i)} = l\} \Sigma_l^{-1} (\tilde{x}^{(i)} - \mu_l), \\
\nabla_{\phi_l} \ell_{\text{semi-sup}} &= \sum_{i=1}^{m} \frac{w_l^{(i)}}{\phi_l} + \alpha \sum_{i=1}^{\tilde{m}} \mathbb{I}\{\tilde{z}^{(i)} = l\} \frac{1}{\phi_l} + \lambda, \\
\nabla_{\Sigma_l} \ell_{\text{semi-sup}} &= \sum_{i=1}^{m} -\frac{w_l^{(i)}}{2} \Sigma_l^{-1} + \frac{w_l^{(i)}}{2} \Sigma_l^{-1} (x^{(i)} - \mu_l)(x^{(i)} - \mu_l)^T \Sigma_l^{-1} \\
&\quad + \alpha \sum_{i=1}^{\tilde{m}} -\frac{\mathbb{I}\{\tilde{z}^{(i)} = l\}}{2} \Sigma_l^{-1} + \frac{\mathbb{I}\{\tilde{z}^{(i)} = l\}}{2} \Sigma_l^{-1} (\tilde{x}^{(i)} - \mu_l)(\tilde{x}^{(i)} - \mu_l)^T \Sigma_l^{-1}.
\end{aligned}$$

Setting the gradients to zero, we get

$$\begin{aligned}
\mu_l &= \frac{\sum_{i=1}^m w_l^{(i)} x^{(i)} + \alpha \sum_{i=1}^{\tilde{m}} \mathbb{I}\{\tilde{z}^{(i)} = l\} \tilde{x}^{(i)}}{\sum_{i=1}^m w_l^{(i)} + \alpha \sum_{i=1}^{\tilde{m}} \mathbb{I}\{\tilde{z}^{(i)} = l\}}, \\
\phi_l &= -\frac{1}{\lambda}\left(\sum_{i=1}^m w_l^{(i)} + \alpha \sum_{i=1}^{\tilde{m}} \mathbb{I}\{\tilde{z}^{(i)} = l\}\right), \\
\Sigma_l &= \frac{\sum_{i=1}^m w_l^{(i)} (x^{(i)} - \mu_l)(x^{(i)} - \mu_l)^T + \alpha \sum_{i=1}^{\tilde{m}} \mathbb{I}\{\tilde{z}^{(i)} = l\} (x^{(i)} - \mu_l)(x^{(i)} - \mu_l)^T}{\sum_{i=1}^m w_l^{(i)} + \alpha \sum_{i=1}^{\tilde{m}} \mathbb{I}\{\tilde{z}^{(i)} = l\}}.
\end{aligned}$$

As $\sum_j \phi_j = 1$, we have

$$\lambda = -\sum_{i=1}^m \sum_j w_j^{(i)} - \alpha \sum_{i=1}^{\tilde{m}} \sum_j \mathbb{I}\{\tilde{z}^{(i)} = j\} = -m - \alpha \tilde{m}.$$

Hence,

$$\phi_l = \frac{\sum_{i=1}^m w_l^{(i)} + \alpha \sum_{i=1}^{\tilde{m}} \mathbb{I}\{\tilde{z}^{(i)} = l\}}{m + \alpha \tilde{m}}.$$

These give the closed form update rules for the model parameters based on the semi-supervised objective.

(d) **Coding problem. Classical (Unsupervised) EM Implementation.** After updating the model parameters, we can calculate the log-likelihood

$$\begin{aligned}
\ell_{\text{semi-sup}} &= \sum_{i=1}^m \log \sum_{z^{(i)}} p(x^{(i)}, z^{(i)}) + \alpha \sum_{i=1}^{\tilde{m}} \log p(\tilde{x}^{(i)}, \tilde{z}^{(i)}) \\
&= \sum_{i=1}^m \log \sum_{z^{(i)}} p(x^{(i)} \mid z^{(i)}) p(z^{(i)}) + \alpha \sum_{i=1}^{\tilde{m}} \log p(\tilde{x}^{(i)}, \tilde{z}^{(i)}) \\
&= \sum_{i=1}^m \log \sum_{z^{(i)}} \frac{\phi_{z^{(i)}}}{(2\pi)^{d/2} |\Sigma_{z^{(i)}}|^{1/2}} \exp\left(-\frac{1}{2}(x^{(i)} - \mu_{z^{(i)}})^T \Sigma_{z^{(i)}} (x^{(i)} - \mu_{z^{(i)}})\right) \\
&\qquad + \alpha \sum_{i=1}^{\tilde{m}} \log \frac{1}{(2\pi)^{d/2} |\Sigma_{\tilde{z}^{(i)}}|^{1/2}} \exp\left(-\frac{1}{2}(\tilde{x}^{(i)} - \mu_{\tilde{z}^{(i)}})^T \Sigma_{\tilde{z}^{(i)}} (\tilde{x}^{(i)} - \mu_{\tilde{z}^{(i)}})\right)
\end{aligned}$$

(e) **Coding problem. Semi-supervised EM Implementation.** Now consider both labelled and unlabelled examples (a total of $m + \tilde{m}$) . with 5 labelled examples per cluster.

(f) **Comparison of unsupervised and semi-supervised EM.** Compared to classical unsupervised EM, semi-supervised EM usually takes fewer iterations to converge, has more stability against random initilaizations, and has better overall quality.

∎

Problem 5 K-means for compression
Use k-means algorithm to lossy image compression, by reducing the number of colors used in an image.
A 512x512 image 24-bit colors takes about $512 \times 512 \times 3 = 786432$ bytes . To compress, we use k-means to reduce the image to $k = 16$ colors. More specifically, each pixel in the image is considered a point in the 3-dimensional (r, g, b) space. To compress, we will cluster these points in colo-space into 16 clusters and replace each pixel with the closest cluster centroid.

(a) **Coding problem. K-means compression Implementation.** See `data/p05_kmeans`.

(b) **Compression factor.** If we represent the image with these reduced (16) colors, we only need to store the 24-bit RGB value for the 16 centroids and the index of centroids for every pixel. We can use a 16-bit integer for the latter, and that would be

$$16 \times 3 + 512 \times 512 \times 2 = 524336 \text{ bytes}$$

in total. The compression factor is about $0.66 \approx \frac{2}{3}$, since we only need 2 bytes instead of 3 for each pixel.

∎