Q2. Written Problems [23 marks]

In Q3, you will be using Gaussian Class Conditional (GCC) and Naive Bayes (NB) to classify images. To that end, we want to understand the effects of dimensionality and explore alternative algorithms/models. The written solutions should be saved as `ExamSol_<utorid>.pdf` in the `Exam` directory.

Curse of Dimensionality. Before you considered GCC and NB, you thought about using K-Nearest Neighbours. You recall that your amazing instructors and teaching assistants mentioned the "curse of dimensionality" multiple times. To figure out why, you came up with the following toy example. A hypersphere is a generalization of the concept of a sphere (not just 3 dimensional). Consider a d dimensional hypersphere with radius r. Its (hyper)volume is

$$V_d(r) = \begin{cases} \frac{\pi^k}{k!} r^d, & d = 2k \\ \frac{2(k!)(4\pi^k)}{(2k+1)!} r^d, & d = 2k+1 \end{cases}. \tag{1}$$

The fraction of its hypervolume lying between values $r - c$ and r, where $0 < c < r$ is given by

$$f = 1 - \left(1 - \frac{c}{r}\right)^d. \tag{2}$$

1. For any fixed c, f tends to 1 as $d \to \infty$. Show this numerically with $\frac{c}{r} = 0.01$, for $d = 2, 10, 500$.

Given $\frac{c}{r} = 0.01$, $f = 1 - (1 - 0.01)^d = 1 - 0.99^d$. When $d=2$, $f = 1 - 0.99^2 = 0.0199$

$d=10$, $f = 1 - 0.99^{10} = 0.0956179\cdots$

$d=500$, $f = 1 - 0.99^{500} = 0.993429517\cdots$

We can see that with d increase, f is getting close to 1.

WTS: $\lim_{d\to\infty} (1 - X^d) = 1$.

Proof: $X = \frac{c}{r}$. since $0 < c < r$, $\therefore 0 < X < 1$.

$\lim_{d\to\infty} (1 - X^d) = \lim_{d\to\infty} 1 - \lim_{d\to\infty} X^d = 1 - \lim_{d\to\infty} X^d = 1 - 0 = 1$. As wanted. ☑

2. For any fixed r, evaluate the fraction of the hypervolume which lies inside the radius $\frac{r}{2}$ for $d = 2, 10, 500$.

Its fraction lies inside radius $r/2$ can be calculated by $c = r/2$:

$f = 1 - (1 - \frac{c}{r})^d = 1 - (1 - \frac{1}{2})^d = 1 - (\frac{1}{2})^{10500} \approx 1$.

3. Suppose we sample points according to a uniform distribution inside a very high-dimensional hypersphere centered at the origin, where in the hypersphere would we most likely find the sampled points? Why is this a problem for some algorithms such as K-Means and K-Nearest Neighbours?

Most likely all points would cumulate at the surface of this high-dimentional sphere. As we can see in (1) f tends to 1 rapidly, and in (2) we got f eventually goes to 1. For KNN and KMeans, as all points are cumulated together, and they are likely to have similar distance, centers cannot be choosed efficiently. Most likely it will never reach an end where the result converges. So these kind of methods are not efficient.

Naive Bayes. In Chapter 9 of the online course lecture notes, we introduced GCC and NB for classification. The generative model for a data point $\mathbf{x}$, in the two class case, is formulated as

$$p(\mathbf{x}) \;=\; \sum_{j=0}^{1} p(\mathbf{x}\,|\,c=j)\;p(c=j). \tag{3}$$

where c is the class random variable; it takes on the value 0 for one class, and 1 for the other. The decision function for the CC model is the log posterior ratio,

$$a(\mathbf{x}) \;=\; \log p(c=1\,|\,\mathbf{x}) - \log p(c=0\,|\,\mathbf{x})\;. \tag{4}$$

To learn the model, given N data points $\{\mathbf{x}_i, c_i\}_{i=1}^{N}$, we estimate the class priors, $p(c=0)$ and $p(c=1)$, and likelihoods, $p(\mathbf{x}|c=0)$ and $p(\mathbf{x}|c=1)$. For GCC models the likelihoods are Gaussian,

$$p(\mathbf{x}|c=j) \;=\; G(\mathbf{x};\mu_j,\Sigma_j)\;, \tag{5}$$

for which we estimate the means and covariances. In NB, we further assume that the features are conditionally independent of the class (i.e. $p(\mathbf{x}|c) = \prod_{i=1}^{d} P(x_i|c)$).

Consider the following problems in the context of binary classification:

1. Compute the prior class probability $\pi = p(c=1)$ for the NB model using Maximum Likelihood (ML) estimate (i.e. $\pi_{ML} = \arg_\pi \max P(\{\mathbf{x}_i, c_i\}_{i=1}^{N}|\pi)$).

$$\pi_{ML} = \frac{\sum_{i=1}^{N} \mathbb{1}(c_i=1)}{N} = \frac{\sum_{i=1}^{N} c_i}{N} \quad (\text{Since } c_i = 1 \text{ or } 0.)$$

2. We had seen that ML is prone to overfitting. As a result we would want to use the Bayes' estimate instead. Let the prior of the prior class probability be distributed following Beta distribution (i.e. $\pi \sim \text{Beta}(\alpha=1, \beta=1)$). Compute the prior class probability π for the NB model using Bayes' estimate. How do α and β affect the prior class probability and reduce overfitting?

$$p(\theta, \alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1} \propto \theta^{\alpha-1}(1-\theta)^{\beta-1}$$

In general Beta distribution, when joining with this binary C,

if we denot $c=1$ as m, $c=0$ as l,

$$p(\theta \mid m, l, \alpha, \beta) = \frac{p(m,\theta|l,\alpha,\beta)\;p(\theta|\alpha,\beta)}{p(m|l,\alpha,\beta)} = \frac{\text{Binomial}(m|N,\theta)\,\text{Beta}(\theta|\alpha,\beta)}{\int \text{Binomial}(m|N,\theta)\,\text{Beta}(\theta|\alpha,\beta)\,d\theta}$$

$$\propto \theta^{\alpha+m-1}(1-\theta)^{\beta+l-1} \propto \text{Beta}(\theta \mid \alpha+m, \beta+l)$$

$\pi = 1$ in this case as $\alpha=1$, $\beta=1$. So π here $\propto \frac{\alpha}{\alpha+\beta}$

The bigger $\frac{\alpha}{\alpha+\beta}$ is, the bigger the probability is.

Like when $\alpha=1$ & $\beta=1$, it means we have no prior knowledge.

It can helps to smooth the fitting & reduce the effection of noise.

3. Compute the parameters of each Gaussian for the NB model using ML estimate.

$$\log L(c) = \log P(X|C) = \sum_{i=1}^{N} \left\{ \log \pi_{ci} + \sum_{j=1}^{D} x_{ji} \log \mu_j + (1-x_{ji}) \log (1-\mu_j) \right\}$$

$$\text{Also} = \sum^{N} \log \pi_{ci} + \log\left(\frac{1}{\sqrt{2\pi \det \Sigma}}\right) - \frac{1}{2}(x_i - \mu_{ci})^T \Sigma_{ci}^{-1} (x_i - \mu_{ci})$$

$$\frac{\partial \log L(c)}{\partial \mu_j} = 0 \Rightarrow \sum_{i=1}^{N} \mathbb{1}(c_i = 1)\left(\frac{x_{ji}}{\mu_j} - \frac{1-x_{ji}}{1-\mu_j}\right) = 0$$

$$\sum_{i=1}^{N} \mathbb{1}(c_i = 1)\left[x_{ji}(1-\mu_j) - (1-x_{ji})\mu_j\right] = 0$$

$$\sum_{i=1}^{N} \mathbb{1}(c_i = 1)\mu_j = \sum_{i=1}^{N} \mathbb{1}(c_i = 1) x_{ji}$$

$$\mu_{j\,MLE} = \frac{\sum_{i=1}^{N} \mathbb{1}(c_i = 1) x_{ji}}{\sum_{i=1}^{N} \mathbb{1}(c_i = 1)} = \frac{\sum_{i=1}^{N} c_i x_{ji}}{\sum c_i}$$

$$\frac{\partial \log L(\theta)}{\partial \Sigma_c^{-1}} = -\sum^{N} \mathbb{1}(c_i = k)\left\{\frac{-\partial \log \sqrt{2\pi \det \Sigma}}{\partial \Sigma_{ci}^{-1}} - \frac{1}{2}(x_i - \mu_{ci})(x_i - \mu_{ci})^T\right\} = 0$$

$$\Sigma_{c\,MLE} = \frac{\sum^{N} \mathbb{1}(c_i = k)(x_i - \mu_{ci})(x_i - \mu_{ci})^T}{\sum^{N} \mathbb{1}(c_i = k)} = \frac{\sum^{N} c_i (x_i - \mu_{ci})(x_i - \mu_{ci})^T}{\sum^{N} c_i}$$

4. Naive Bayes is useful when features are conditionally independent and it can also reduce the number of parameters. Suppose you know that only some features are conditionally independent, what can you do instead to model some correlations while reducing number of parameters? Explain how your approach reduces number of parameters.

I would only use the conditionally independent parameters (suppose there are k of them. $0 < K < N$) & ignore their covariances. For example, given $P(x_1 \cdots x_n | Y) = \prod_i P(x_i | Y)$, it will be:

$$= P(x_1|Y) P(x_2|Y) \cdots P(x_n|Y)$$

And $P(Y | x_1 \cdots x_n) = \frac{P(x_1 \cdots x_n | Y) P(Y)}{P(x_1 \cdots x_n)}$.

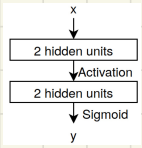
So I only need to estimate $P(x_1|Y)$, $P(x_2|Y) \cdots P(x_k|Y)$, $P(Y)$

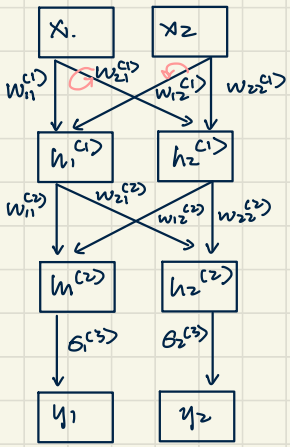
Originally I need $2(2^n - 1) + 1$ parameters.

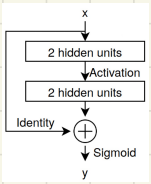
Now with the conditional independent assumption I need $(2k+1)$ parameters.

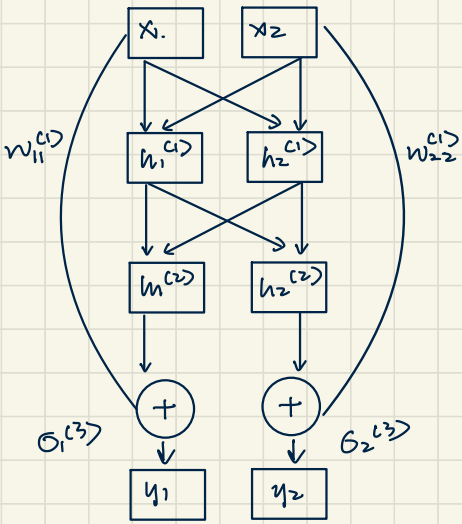
Figure 1: **Left:** Input x is fed into a fully connection network with two hidden layers, each with two hidden units. The first hidden layer's activation is any arbitrary non-linear function and the second hidden layer's activation is a sigmoid. The output of the sigmoid is the prediction y. **Right:** Input x is fed into a residual network. The network has two hidden layers, each with two hidden units. The first hidden layer's activation is any arbitrary non-linear function. The output of the second layer is summed with input x (i.e. a **skip connection**) right before being fed into a sigmoid. The output of the sigmoid is the prediction y.

1. Draw the computational graphs with networks f and g. For simplicity, ignore the bias term. You may also find it helpful to denote the j'th activation in the i'th layer as $\sigma_j^{(i)}$, the linear combination between i'th layer inputs and parameters of the j'th hidden unit as $h_j^{(i)}$, and the k'th parameter in the j'th hidden unit in the i'th as $w_{jk}^{(i)}$.









2. Let t be the label, y be the prediction, and $\mathcal{L}(y, t)$ be a loss function. Compute the derivative of $\mathcal{L}$ with respect to x using f and g. For simplicity, you may write the gradient in terms of partial derivatives.

Gradient $= y - t$

$$\partial \mathcal{L}/\partial x = \partial \mathcal{L}/\partial f \cdot \partial f/\partial x + \partial \mathcal{L}/\partial g \cdot \partial g/\partial x$$

But here $y = x + f(x)$

$$= \partial \mathcal{L}/\partial f \cdot \begin{bmatrix} \partial f_1/\partial x_1 & \partial f_1/\partial x_2 \\ \partial f_2/\partial x_1 & \partial f_2/\partial x_2 \end{bmatrix} + \partial \mathcal{L}/\partial g \cdot \begin{bmatrix} \partial g_1/\partial x_1 & \partial g_1/\partial x_2 \\ \partial g_2/\partial x_1 & \partial g_2/\partial x_2 \end{bmatrix}$$

$$= \partial \mathcal{L}/\partial f \cdot \begin{bmatrix} \partial f_1/\partial x_1 & \partial f_1/\partial x_2 \\ \partial f_2/\partial x_1 & \partial f_2/\partial x_2 \end{bmatrix} + \partial \mathcal{L}/\partial g \cdot \begin{bmatrix} 1+(\partial f_1/\partial x_1) & 1+(\partial f_1/\partial x_2) \\ 1+(\partial f_2/\partial x_1) & 1+(\partial f_2/\partial x_2) \end{bmatrix}$$

3. Using the derivatives above, provide an example where f experiences **vanishing gradient** (i.e. $\frac{\partial \mathcal{L}}{\partial x} = 0$) whereas g does not suffer from this problem. **Hint:** Define the activation in the first hidden layer.

So if in some case $f(x) = 0$.

The first para in (2) is going to vanish as $\partial f_i / \partial x_i = 0$

But in skip connection there's activation in 1st layer, and the algorithm is like $y = x + f(x)$. Hence $x^i = x^{i+1} + x^{i+1}(\partial f / \partial x)$

$= x^{i+1}(1 + \partial f / \partial x)$

which still exist. The skip connection is quite stable as the gradient $\partial f / \partial x \to 0$ or very small. In this kind of case, g works but f fails.