CSCI-GA.2565-001 Machine Learning: Part 2

Chuanyang Jin

Nov 10, 2022

1 Very Random Forest

Consider building a random forest by both subsampling the data and choosing a single feature per tree randomly. For example, consider a dataset $\mathcal{D} = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$ where $\mathbf{x}_i \in \mathbb{R}^D$ and $y_i \in \mathbb{R}$ for $i = 1, .., N$. We will carry out the following procedure:

1. Randomly sample one feature index $j \in \{1, \ldots, D\}$.
2. Draw a sample of the data $\mathcal{D}_{\mathbf{k}}$ of size $M \leq N$ with replacement. These datapoints will have indices $\mathbf{k} = k_1, \ldots, k_M$.
3. Keep only the j^{th} feature of the M samples: i.e. letting $x_{i,j}$ be the i^{th} datapoint and j^{th} feature, we use data
$$\mathcal{D}^j_{\mathbf{k}} = \{(x_{(k_1,j)}, y_{(k_1)}), ..., (x_{(k_M,j)}, y_{(k_M)})\}$$
4. Build a decision tree on $\mathcal{D}^j_{\mathbf{k}}$.
5. Repeat the above process R times so that r^{th} tree T_r uses feature $j^{(r)}$ and data $\mathbf{k}^{(r)}$ for $r \in \{1, \ldots, R\}$. Using this notation, the prediction of the r^{th} tree on new input $\mathbf{x}^\star$ is $T_r(\mathbf{x}^\star; \mathcal{D}^{j^{(r)}}_{\mathbf{k}^{(r)}})$.
6. Average these random trees to construct the random forest. That is, for input $\mathbf{x}^\star$, the random forest predicts $\hat{y} = \frac{1}{R}\sum_{r=1}^{R} T_r(\mathbf{x}^\star; \mathcal{D}^{j^{(r)}}_{\mathbf{k}^{(r)}})$.

Let us call this model a Very Random Forest (VRF). In this problem, we will attempt to answer the question: for which kinds of data distribution $y \mid \mathbf{x}$ are VRFs *unbiased* (in the context of bias-variance tradeoff)?

(A) Write down the two term that have to be equal for a model to be unbiased. One term should be some statistic of the true data distribution and the other should be a statistic of the model output.

Solution.

Generally, for a model to be unbiased, we will have $\mathbb{E}[y|x^*] = \mathbb{E}_D[\hat{y}(x; D)]$, where $\hat{y}(x; D)$ represents the prediction by a model trained on the dataset D and tested for an input x. □

(B) Let us first consider a single decision tree $T(\mathbf{x}, \mathcal{D}^j)$ that uses only a single feature j but the **entire** dataset (i.e. all N data points). Assuming that the tree was trained by minimizing squared loss, what will be the tree's prediction for a test point $\mathbf{x}^\star$? I.e. write down the function that $T(\mathbf{x}^\star, \mathcal{D}^j)$ corresponds to.

Solution.

Let c_k be chosen in the k^{th} node. We hope to minimize the loss $S = \sum (y_i - c)^2$. By setting the derivative of S to 0, we have

$$\nabla_c S = \sum 2(c - y_i) = 0$$

$$c_k = \frac{1}{|N_k|} \sum y_i$$

Thus our VRF will split at $x_k \in D$ with y_k closest to c_k. $T(\mathbf{x}, D^j)$ will split $\mathbb{R}^D$ into leaves where all training points have the same label values. Therefore, $T(\mathbf{x}^*, D^j) = \sum_i 1_{\mathbf{x}^* \in \text{leaf}_i} \mathbb{E}[y|\mathbf{x}_j^*]$. □

(C) Now, express the model-dependent term that you wrote in part (A) in terms of your answer in (B).

Solution.

The model-dependent term in part (A) can be expressed as $\mathbb{E}_D[\hat{y}(x; D)] = \frac{1}{R} \sum_{r=1}^{R} T_r(\mathbf{x}; \mathcal{D}_{\mathbf{k}^{(r)}}^{j^{(r)}})$. For $T_r(\mathbf{x}; \mathcal{D}_{\mathbf{k}^{(r)}}^{j^{(r)}})$, there will be at most $C(M) = \binom{N+M-1}{M}$ different trees with equal probabilities. We can factor out $1_{\mathbf{x}^* \in \text{leaf}_i}$ since we already include all the possible tree structures. So we have

$$\begin{aligned}
\mathbb{E}_D[\hat{y}(x; D)] &= \frac{1}{R} \sum_{r=1}^{R} T_r(\mathbf{x}; \mathcal{D}_{\mathbf{k}^{(r)}}^{j^{(r)}}) \\
&= \frac{1}{R} \sum_{r=1}^{R} \sum_{C(M)} \frac{1}{C(M)} \sum_i 1_{\mathbf{x}^* \in \text{leaf}_i} \mathbb{E}_{D_r^j}[y|\mathbf{x}_j^*] \\
&= \frac{1}{R \cdot C(M)} \sum_{r=1}^{R} \sum_{C(M)} \mathbb{E}_{D_r^j}[y|\mathbf{x}_j^*]
\end{aligned}$$

□

(D) As $N \to \infty$, what data-dependent function does your answer in part (C) converge to? You may assume that your VRF has sufficient capacity to model this function arbitrarily well.

Now equate this to the data-dependent term you describe in (A) and describe the distributions $y|\mathbf{x}$ for which the model will be unbiased.

Solution.

As $N \to \infty$, since our VRF is trained on all data and we assume it can model them arbitarily well, each $\mathbb{E}_{D_r^j}[y|\mathbf{x}_j^*]$ will converge to $\mathbb{E}[y|\mathbf{x}_j^*]$. Our answer in part (C) will converge to $\frac{1}{R} \sum_{r=1}^{R} \mathbb{E}[y|\mathbf{x}_j^*]$. For our VRF to be unbiased, we need

$$\mathbb{E}[y|x^*] = \mathbb{E}_D[\hat{y}(x; D)]$$

Then

$$\mathbb{E}[y|x^*] = \frac{1}{R} \sum_{r=1}^{R} \mathbb{E}[y|\mathbf{x}_j^*]$$

This requires all features of $\mathbf{x}$ to be independent from each other. □

(E) Compare the bias and variance of the VRF with the traditional random forest, where we select a random subset of the data and a random subset of features to build each tree.

Hint: Look at the generalization bound from the lecture on random forests. You only need to look at the final result, not the derivation.

Solution.

According to the lecture 6 slide 48, a generalization error bound is

$$ge \leq \frac{\bar{\rho}(1-s^2)}{s^2}$$

where $\bar{\rho}$ represents the correlation between trees, and s represents the strength of trees.

Each tree in our VRF is built on only one feature and thus has weaker strength. Therefore, compared to the traditional random forest, our VRF will have weaker strength of trees, and thus having higher bias.

Since each tree in our VRF is built on only one feature, the likelihood of two trees using the same feature is greatly decreased. Therefore, compared to the traditional random forest, our VRF will have weaker correlation between trees, and thus having lower variance. □

2 Conditional Modelling with Gaussians

For any finite set of points $\mathcal{D} = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$, assume the following conditional model:

$$\begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \Bigg| \, \mathbf{x}_1, ..., \mathbf{x}_N \sim \mathcal{N}(\vec{0}, K)$$

Here, K is a covariance matrix such that $K_{ij} := k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{1}{2}\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2)$. You may assume that K is positive-definite for any dataset $\mathcal{D}$.

(A) For any set of $(N+1)$ points $\mathbf{x}_1, \ldots, \mathbf{x}_{N+1}$, write the down the distribution of $y_1, \cdots y_N, y_{N+1} \mid \mathbf{x}_1, \cdots \mathbf{x}_N, \mathbf{x}_{N+1}$ under the assumed model.

Solution.

$$\begin{bmatrix} y_1 \\ \vdots \\ y_{N+1} \end{bmatrix} \Bigg| \, \mathbf{x}_1, ..., \mathbf{x}_{N+1} \sim \mathcal{N}(\vec{0}, K_{N+1})$$

where $\vec{0}$ contains $N+1$ components, and K_{N+1} is a $(N+1) \times (N+1)$ covariance matrix such that $K_{N+1\ ij} := k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{1}{2}\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2)$. □

(B) Given a dataset $\mathcal{D} = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$ and a test point $\mathbf{x}_{N+1}$, how do you make a prediction using this model?

Solution.

$$\begin{aligned}
P(y_{N+1}|\mathbf{x}_1, ..., \mathbf{x}_N, \mathbf{x}_{N+1}, y_N) &= \frac{p(\mathbf{x}_1, ..., \mathbf{x}_N, \mathbf{x}_{N+1}, y_1, ..., y_N, y_{N+1})}{p(\mathbf{x}_1, ..., \mathbf{x}_{N+1}, y_1, ..., y_N)} \\
&= \frac{p(\mathbf{x}_1, ..., \mathbf{x}_N, \mathbf{x}_{N+1}, y_1, ..., y_N, y_{N+1})}{p(\mathbf{x}_1, ..., \mathbf{x}_N, y_1, ..., y_N)p(\mathbf{x}_{N+1})} \\
&= \frac{p(\mathbf{x}_1, ..., \mathbf{x}_N, \mathbf{x}_{N+1}, y_1, ..., y_N, y_{N+1})/p(\mathbf{x}_1, ..., \mathbf{x}_N, \mathbf{x}_{N+1})}{p(\mathbf{x}_1, ..., \mathbf{x}_N, y_1, ..., y_N)/p(\mathbf{x}_1, ..., \mathbf{x}_N)} \\
&= \frac{p(y_1, ..., y_N, y_{N+1} \mid \mathbf{x}_1, ..., \mathbf{x}_N, \mathbf{x}_{N+1})}{p(y_1, ..., y_N \mid \mathbf{x}_1, ..., \mathbf{x}_N} \\
&= \frac{\frac{1}{\sqrt{(2\pi)^{N+1}|K_{N+1}|}} \exp\left(-\frac{1}{2}[y_1, ..., y_N, y_{N+1}]^\top K_{N+1}^{-1}[y_1, ..., y_N, y_{N+1}]\right)}{\frac{1}{\sqrt{(2\pi)^N|K_N|}} \exp\left(-\frac{1}{2}[y_1, ..., y_N]^\top K_{N+1}^{-1}[y_1, ..., y_N]\right)} \\
&= \frac{1}{\sqrt{(2\pi)|1 - \mathbf{k}_{N+1}^\top K_N^{-1}\mathbf{k}_{N+1}|}} \exp\Big(-\frac{1}{2}([y_1, ..., y_N, y_{N+1}]^\top K_{N+1}^{-1}[y_1, ..., y_N, y_{N+1}] \\
&\quad - [y_1, ..., y_N]^\top K_N^{-1}[y_1, ..., y_N])\Big) \\
&\sim \mathcal{N}(\mathbf{k}_{N+1}^\top K_N^{-1}[y_1, ..., y_N], 1 - \mathbf{k}_{N+1}^\top K_N^{-1}\mathbf{k}_{N+1})
\end{aligned}$$

where

$$K_{N+1} = \begin{bmatrix} K_N & \mathbf{k}_{N+1} \\ \mathbf{k}_{N+1}^\top & k(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}) \end{bmatrix}$$

$$\mathbf{k}_{N+1} = [k(\mathbf{x}_1, \mathbf{x}_{N+1}), k(\mathbf{x}_2, \mathbf{x}_{N+1}), ..., k(\mathbf{x}_N, \mathbf{x}_{N+1})]$$

Therefore we predict using

$$P(y_{N+1}|\mathbf{x}_1, ..., \mathbf{x}_N, \mathbf{x}_{N+1}, y_N) \sim \mathcal{N}(\mathbf{k}_{N+1}^\top K_N^{-1}[y_1, ..., y_N], 1 - \mathbf{k}_{N+1}^\top K_N^{-1}\mathbf{k}_{N+1})$$

□

(C) Suppose that the test point $\mathbf{x}_{N+1}$ is an outlier, e.g. assume $\min_{1\le i\le N} \|\mathbf{x}_{N+1} - \mathbf{x}_i\| > 1000$. Using your answer in part (B), what is your prediction for $\mathbf{x}_{N+1}$?

Solution.

Assume $\min_{1\le i\le N} \|\mathbf{x}_{N+1} - \mathbf{x}_i\| > 1000$, then $k(\mathbf{x}_i, \mathbf{x}_{N+1}) = \exp(-\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_{N+1}\|^2)$ approaches 0 for any $i = 1, ..., N$. So the mean of the Normal distribution $\mathbf{k}_{N+1}^\top K_N^{-1}[y_1, ..., y_N]$ approaches 0, and the variance $1 - \mathbf{k}_{N+1}^\top K_N^{-1}\mathbf{k}_{N+1})$ approaches 1. Therefore, our prediction for $\mathbf{x}_{N+1}$ approaches $\mathcal{N}(0, 1)$. □

(D) Now, imagine you fit a flexible neural network f_θ on the same dataset $\mathcal{D}$. Consider the same outlier $\mathbf{x}_{N+1}$ as in part (C). What can you say about the prediction $f_\theta(\mathbf{x}_{N+1})$? Does your answer change depending on your choice of network f_θ?

Hint: Think about the constraints a neural network may impose on the prediction for $\mathbf{x}_{N+1}$.

Solution.

It is hard to say about the prediction $f_\theta(\mathbf{x}_{N+1})$.
In our conditional model with Gaussians, the prediction for input data is more influenced by the observation made on close input data. However, a flexible neural network f_θ contains more layers, and its prediction is more complicated. Sometimes the output is sensitive to the input, so the output of the outlier $\mathbf{x}_{N+1}$ may be far away from $y_1, ..., y_N$.
On the other hand, the activation functions in the neural network impose some constrains on the prediction, and a lot of extreme values will become less extreme after the activation functions. So even though the outlier $\mathbf{x}_{N+1}$ is far away from other input data, its output may be not far away. So it is hard to say about the prediction. □

(E) Compare the predictions you made in parts (C) and (D). Are they the same? If not, explain the difference. If one has a problem, suggest a way to solve it.

Solution.

They are not the same. In part (C), the prediction of the conditional Gaussian model always approaches 0, so it is inaccurate. In part (d), the neural network gives different predictions for different outliers, and it may be either good or bad.
The prediction in part (c) is directly related to the kernel function $k(\mathbf{x}_i, \mathbf{x}_{N+1}) = \exp(-\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_j\|^2)$, which always approaches 0 for an outlier. This makes the prediction always approaches 0. To solve this problem, we can choose a different kernel function that diminishes the effect of the distance between training and test data. For example, we can let $k(\mathbf{x}_i, \mathbf{x}_{N+1}) = \exp(-\frac{\epsilon}{2}\|\mathbf{x}_i - \mathbf{x}_j\|^2)$ where ϵ is a small value such as $\frac{1}{1000}$.
To improve the prediction in part (d), we may use a larger training dataset, change the activation function, or add data normalization schemes in the neural network. □

3 Neural Networks for Reconstruction

In class, we saw how neural networks are a flexible model class that can be used for supervised regression or classification tasks, i.e. predicting y from $\mathbf{x}$. Let us consider a neural network that tries to predict $\mathbf{x}$ from $\mathbf{x}$ instead. In other words, given an input $\mathbf{x} \in \mathbb{R}^D$, the neural network will be trained to reconstruct the same $\mathbf{x}$.

Let us consider the network's architecture as follows:

$$\mathbf{h} = a(\mathbf{V}\mathbf{x})$$
$$\hat{\mathbf{x}} = \mathbf{W}\mathbf{h}$$

Here, our network contains a single hidden layer of size H, hence $\mathbf{V}$ is a $H \times D$ matrix and $\mathbf{W}$ is a $D \times H$ matrix. $a(\cdot)$ is an element-wise non-linear activation function that we will leave undecided for now. We have omitted bias terms to simplify the math.

I Gradients

(A) Write down the loss function L of our network for a single data point $\mathbf{x}$, assuming element-wise squared loss. Compute the derivative $\frac{\mathrm{d}L}{\mathrm{d}\mathbf{h}}$. Do not leave any symbolic expressions of the form $\frac{dy}{dx}$, i.e. actually compute the derivative in terms of known quantities like $\mathbf{W}$ or $\mathbf{x}$.

Solution.

The loss function is $L = ||\hat{\mathbf{x}} - \mathbf{x}||_2^2$. Thus we have

$$\begin{aligned}\frac{\mathrm{d}L}{\mathrm{d}\mathbf{h}} &= \frac{\mathrm{d}L}{\mathrm{d}\hat{\mathbf{x}}}\frac{\mathrm{d}\hat{\mathbf{x}}}{\mathrm{d}\mathbf{h}} \\ &= 2(\hat{\mathbf{x}} - \mathbf{x})^\top \mathbf{W} \\ &= 2(\mathbf{W} \cdot a(\mathbf{V}x) - \mathbf{x})^\top \mathbf{W}\end{aligned}$$

□

(B) Let $v := \mathbf{V}_{1,1}$ be the element in the first row and first column of $\mathbf{V}$. Compute the derivative $\frac{\mathrm{d}\mathbf{h}}{\mathrm{d}v}$. You can assume that $\frac{\mathrm{d}a(x)}{\mathrm{d}x} = a'(x)$. Next, using the chain rule and your answer in (A), compute the derivative $\frac{\mathrm{d}L}{\mathrm{d}v}$. As in (A), do not leave any symbolic expressions of the form $\frac{dy}{dx}$.

Solution.

Let $\mathbf{V}_{1,:}$ be the first row of matrix $\mathbf{V}$, and $\mathbf{W}_{:,1}$ be the first column of matrix $\mathbf{W}$. Then we have

$$\frac{\mathrm{d}\mathbf{h}}{\mathrm{d}v} = \begin{bmatrix} a'(\mathbf{V}_{1,:}\mathbf{x})x_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{H \times 1}$$

$$\begin{aligned}\frac{dL}{dv} &= \frac{dL}{d\mathbf{h}}\frac{d\mathbf{h}}{dv} \\ &= 2a'(\mathbf{V}_{1,:}\mathbf{x})x_1(\hat{\mathbf{x}} - \mathbf{x})^\top \mathbf{W}_{:,1}\end{aligned}$$

□

II Representation

(C) How does our choice of H affect the neural network's performance? Assuming that our optimization procedure successfully finds a network with minimum loss, describe the characteristics of an optimal network in the cases: (i) $H > D$, (ii) $H = D$, and (iii) $H < D$.

In each of these cases, is zero loss always possible? If not, why not? If sometimes, give examples of data distributions where zero loss is or is not possible.

Solution.

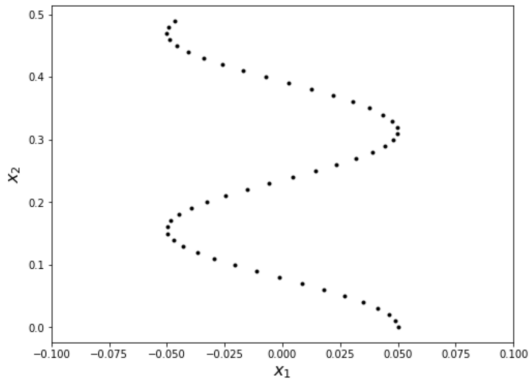
(i) $H > D$, zero loss is always possible. For $\mathbf{Wh} = \mathbf{x}$ to have a solution, the column space of $\mathbf{W}$ must span $\mathbb{R}^D$ for given $\mathbf{x}$. So $\mathbf{W}$ is of rank D, and the row space of $\mathbf{W}$ span $\mathbb{R}^D$. For $\mathbf{Vx} = a^{-1}\mathbf{h}$ to have a solution, since a^{-1} can be viewed as a basis transformation, $\mathbf{V}$ has full rank, and the row space of $\mathbf{V}$ span $\mathbb{R}^D$. Since both the column space of $\mathbf{W}$ and the row space of $\mathbf{V}$ span $\mathbb{R}^D$, for any given $x \in \mathbb{R}^D$, the linear system will be solvable, thus achieving zero loss.

(ii) $H = D$, zero loss is always possible. Both $\mathbf{V}$ and $\mathbf{W}$ will be invertible, and both the column space of $\mathbf{W}$ and the row space of $\mathbf{V}$ span $\mathbb{R}^D$. Then the linear system will be solvable, thus achieving zero loss.

(iii) $H < D$, zero loss is sometimes possible. For $\mathbf{Wh} = \mathbf{x}$ to have a solution, x must be in the column space of $\mathbf{W}$, which is a subspace with H basis. Given $\mathbf{h}$ in the row space of $\mathbf{V}$, x must also be in the row space of $\mathbf{V}$, which is also a subspace with H basis. Therefore, x needs to be in a subspace with H basis, then it is possible to achieve zero loss.

For example, if the model is trained with data in the space spanned by $\{e_1, ..., e_H\}$, where $e_i \in \mathbb{R}^D$ with i^{th} entry equal to 1 and other entries equal to 0. Then the column space of $\mathbf{W}$ and the row space of $\mathbf{V}$ can both be the space spanned by $\{e_1, ..., e_H\}$, then it can achieve zero loss. If the data distribution spans a space with more than H basis, then it is not possible to achieve zero loss. □

(D) How does our choice of a affect our neural network's performance? Consider two activation functions: (i) $a(x) = \frac{1}{1+\epsilon^{-x}}$ (the sigmoid function), and (ii) $a(x) = x$ (the identity function). For the 2D input data described by the plot below, what might the hidden representation $\mathbf{h}$ look like for each of these activation functions? Why?



Solution.

When our model achieves zero loss, then $\hat{x} = \mathbf{Wh}$ looks the same as the curve in the plot. (i) For the sigmoid function, all entries of $\mathbf{h}$ will be in the first quadrant with the same relative position, but only be rotated and compressed. Multiplying $\mathbf{W}$ will rotate and re-scale them to the positions in the plot. (ii) For the identity function, h will be some pattern transformed from the plot by rotation and compression. □

4 Building a Latent Variable Model

In this question, you will build a latent variable model by making certain assumptions about your data. Now, you have data about N patients that were treated with a new drug to control their blood cholesterol (BC) level. For each patient $1 \leq i \leq N$ in the dataset, you observe a single scalar change in BC and no other data; let us denote this value as x_i. A biologist collaborator tells you that there potentially exists K patient characteristics that affect how BC changes due to drug intake. Since these factors are unmeasured, you build a model with latent variables $z_{i,k}$ that indicate the presence of characteristic k for patient i.

Denote the characteristic vector for patient i as $\mathbf{z}_i = [z_{i,1}, \cdots, z_{i,K}]$ where $z_{i,k} \in \{0, 1\}$ (i.e. assume that the latent variables are binary). Furthermore, assume that $(x_i, \mathbf{z}_i) \perp\!\!\!\perp (x_j, \mathbf{z}_j)$ for $i \neq j$.

(A) Choose a prior distribution for the characteristic vectors $\mathbf{z}_i \sim p(\mathbf{z})$ and justify your choice, i.e. describe the assumptions that you made.

Solution.

We choose the prior

$$\begin{aligned} p_{\mathbf{z}}(\mathbf{z}_i) &= \prod_{z_{i,k}=1} p_k \prod_{z_{i,k}=0} (1 - p_k) \\ &= p_k^{|\mathbf{z}_i|^2} (1 - p_k)^{K - |\mathbf{z}_i|^2} \end{aligned}$$

where $|\mathbf{z}_i|^2 = \sum_k z_{i,k}^2 = \sum_{z_{i,k}=1} 1$.

It is under the assumptions that
1. $z_{i,1}, ..., z_{i,k}$ are pairwise independent;
2. $z_{i,k} \sim \text{Bernoulli}(p_k)$ for $k = 1, 2, ..., K$.

□

(B) The biologist tells you that given any patient's characteristics $\mathbf{z}_i$, the observed BC can be modeled as:

$$v_{i,k} \sim \mathcal{N}(\mu_k, 1), \qquad \epsilon_i \sim \mathcal{N}(0, 1), \quad x_i = \sum_{k=1}^{K} z_{i,k} v_{i,k} + \epsilon_i$$

where $\{\mu_k\}_{1 \leq k \leq K}$ are known quantities. Write down the the likelihood distribution $p(x_i \mid \mathbf{z}_i)$ explicitly.

Solution.

Given a $\mathbf{z}_i$, we have

$$x_i \mid \mathbf{z}_i = \sum_{k=1}^{K} z_{i,k} v_{i,k} + \epsilon_i = \sum_{z_{i,k}=1} v_{i,k} + \epsilon_i$$

It is a linear combination of Normal distributions, which follows a new Normal distribution. If we denote $\boldsymbol{\mu} := [\mu_1, \mu_2, ..., \mu_K]$, then $\mathbf{z}_i \mu = \sum_{z_{i,k}=1} \mu_k$ and $|z_i|^2 = \sum_{z_{i,k}=1} 1$. We have

$$x_i \mid \mathbf{z}_i \sim \mathcal{N}(z_i \cdot \mu, |\mathbf{z}_i|^2 + 1)$$

Therefore we have

$$p(x_i | \mathbf{z}_i) = \frac{1}{\sqrt{2\pi(|\mathbf{z}_i|^2 + 1)}} \exp\left[-\frac{(x_i - z_i \cdot \mu)^2}{2(|\mathbf{z}_i|^2 + 1)} \right]$$

□

(C) Under your choice of prior in part (A) and the likelihood specified by the biologist in part (B), write down the implied marginal distribution $p(x_i)$ in terms of known quantities like $\{\mu_k\}_{1\le k\le K}$. How many modes can this distribution have?

Solution.

Let S be the sample space of z_i. Let $\boldsymbol{\mu}$ follow our definition in (b). We have

$$p(x_i) = \sum_{\mathbf{z}_i \in S} \left(p_k^{|\mathbf{z}_i|^2} (1-p_k)^{K-|\mathbf{z}_i|^2} \frac{1}{\sqrt{2\pi(|\mathbf{z}_i|^2+1)}} \exp\left[-\frac{(x_i - z_i \cdot \mu)^2}{2(|\mathbf{z}_i|^2+1)} \right] \right)$$

The distribution can have at most 2^K modes since there are at most 2^K different $\mathbf{z}_i$. □

(D) How would you infer $\mathbf{z}_i$ for any patient i? Write down one way to do it in terms of observed data $\{x_i\}_{1\le i\le N}$ and known quantities $\{\mu_k\}_{1\le k\le K}$.

Solution.

To infer $\mathbf{z}_i$ for any patient i, we hope to find $\mathbf{z}_i$ with the maximum posterior $p(\mathbf{z}_i \mid x_i)$.

$$\operatorname*{argmax}_{\mathbf{z}_i} p(\mathbf{z}_i \mid x_i) = \operatorname*{argmax}_{\mathbf{z}_i} \frac{p(\mathbf{z}_i, x_i)}{p(x_i)} = \operatorname*{argmax}_{\mathbf{z}_i} \frac{p(x_i \mid \mathbf{z}_i) p(\mathbf{z}_i)}{p(x_i)}$$

We further assume that $p_k = \frac{1}{2}$ for any k in the Bernoulli distribution, so $p(z_i)$ share the same prior likelihood. Since $\{x_i\}_{1\le i\le N}$ are observed data, we only need to find

$$\begin{aligned}\operatorname*{argmax}_{\mathbf{z}_i} p(x_i \mid \mathbf{z}_i) &= \operatorname*{argmax}_{\mathbf{z}_i} \frac{1}{\sqrt{2\pi(|\mathbf{z}_i|^2+1)}} \exp\left[-\frac{(x_i - \mathbf{z}_i \cdot \mu)^2}{2(|\mathbf{z}_i|^2+1)} \right] \\ &= \operatorname*{argmin}_{\mathbf{z}_i} \frac{1}{\sqrt{\phi(\mathbf{z}_i)}} \exp\left[\frac{(x_i - \mathbf{z}_i \cdot \mu)^2}{\phi(\mathbf{z}_i)} \right]\end{aligned}$$

where $\phi(\mathbf{z}_i) = |\mathbf{z}_i|^2 + 1$.
For each fixed $\phi(\mathbf{z}_i) \in [1, 2, ...K+1]$, we can design fast algorithms to find the $\mathbf{z}_i$ that minimizes $|x_i - z_i \cdot \mu|$. There are at most 2^K possibilities. Then we compare their results to determine the final $\mathbf{z}_i$. □

(E) Now consider a different likelihood model than in (B). Instead of giving you exact values, suppose that the biologist now tells you a range for each μ_k in $\{\mu_k\}_{1\le k\le K}$. What prior would you place on these means and why?

Solution.

We can place a uniformly distributed prior over the range of each μ_k. Since we have no information about which value in the range is more likely, we assume that each value in the range has the equal probability. □

(F) Now consider yet another likelihood model, where we let each possible value of $\mathbf{z}_i$ parameterize the likelihood with a separate mean $\mu_{\mathbf{z}_i} = \mathbb{E}[x_i|\mathbf{z}_i]$? What are the trade-offs of this model compared to our original one in (B)? For example, what happens when K is large?

Solution.

When K is large, the model in (B) requires 2^K summations, which is computationally expensive. In the new model, we only need to take care of each feature independently, so it decreases the computational difficulty. We can also update a particular mean without influencing the value of other means.

However, the model in (B) considers the dependency between features, but the new model does not consider this.

□