CMPS 242: Machine Learning: Fall 2016: Homework 2 Solutions

Due: 11th October 2016

Question 1 (1 + 1 + 0.5 + 0.5 points): Consider the following 1-d dataset with 5 points $X = \{-1, 1, 10, -0.5, 0\}$, on which we are going to perform Gaussian density estimation. For the exercise below, you may use Python for plotting but all the calculations have to be done by hand.

- Compute the Maximum Likelihood Estimate (MLE) of the mean and variance. For the variance, compute both the unbiased and biased versions. Comment on what you observe. In particular, how does the presence of an outlier affect your estimates.

We will just use the following formulas for the estimates of the mean μ_{ML}, biased variance σ^2_{ML} and unbiased variance S^2_{ML}.

$$\mu_{\text{ML}} = \frac{1}{N}\sum_{n=1}^{N} x_n$$

$$\sigma^2_{\text{ML}} = \frac{1}{N}\sum_{n=1}^{N} (x_n - \mu_{ML})^2$$

$$S^2_{\text{ML}} = \frac{1}{N-1}\sum_{n=1}^{N} (x_n - \mu_{ML})^2$$

For the calculation itself, the variance decreases drastically once we remove the outlier. This can also be seen in the plots from the ipython notebook that accompanies this solution key, where the variance decreases from a value of approximately 20 to a value of 0.5! This just shows that the MLE estimate with a few data points is very sensitive to outliers.

- Assume that you have a $\mathcal{N}(0, 1)$ prior over the mean parameter and set the standard deviation $\sigma^2 = 1$. Compute the posterior distribution of the mean parameter and plot both the prior and the posterior distributions. Comment on what you observe.

This model is less sensitive to outliers. It can be seen that if we remove the outlier, the value of the variance decreases only slightly. However, we must also note that the prior we chose has a mean parameter that is very close to the mean of the dataset without the outlier.

Overall, our performance has still improved from the MLE case since now we have a Gaussian fit that is less sensitive to the outlier. Check the notebook for plots to convince yourself that this is the case.

- Now suppose we change the prior over the mean parameter to $\mathcal{N}(10, 1)$. Compute the new posterior distribution, plot it, and contrast it with what you observed previously.

Even though our distribution is less prone to overfit, we can see here very clearly that the influence of the prior can also be a negative one. The prior will completely skew the mean of the distribution towards the prior mean, and this is the case even when we don't have the outlier in the dataset.

- Suppose 2 more data points get added to your dataset:

$$X = \{-1, 1, 10, -0.5, 0, 2, 0.5\}$$

Using the same $\mathcal{N}(0, 1)$ prior over the mean parameter, compute and plot the posterior. How does observing new data points affect the posterior?

Adding two more datapoints to the dataset will help center the inferred mean around the true mean. The influence of the outlier is thus even smaller. See the notebook for more details.

Question 2 (1 + 0.5 + 2 + 0.5 points): Generate 100 data points as follows: Draw x uniformly at random[1] from $[-100, 100]$. For each x draw t from $\mathcal{N}\{f(x), 1\}$ where $f(x) = 0.1 + 2x + x^2 + 3x^3$. In order to fit this curve, we will make use of the following probabilistic model:

$$p(t|x, \mathbf{w}, \beta) = \mathcal{N}\left(t|y(x, \mathbf{w}), \beta^{-1}\right)$$

where $y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$.

[1] Do not forget to seed your random number generators!

- Perform MLE estimation of $\mathbf{w}$ and β. You may use the `optimize` module from `scipy` for this task. Comment on how well $\mathbf{w}$ and β match the true parameters used to generate the data. How do the estimates change when you use 1000 or 10,000 data points for your estimates?

> In this part of the problem, we use the function `curve_fit` from `scipy.optimize` to perform least squares curve fitting and obtain $\mathbf{w}_{ML}$ (you could use the `minimize` function instead, as well). The error function we need to minimize is given by (refer to the slides on Bayesian Curve Fitting):
>
> $$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2$$
>
> Then, β_{ML} can be then found by using the following expression:
>
> $$\frac{1}{\beta_{ML}} = \frac{1}{N}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}_{ML}) - t_n\}^2$$
>
> Please refer to the ipython notebook for the code. The observations from this experiment are that even with 100 data points, the MLE estimates obtained for $\mathbf{w}_{ML}$ and β_{ML} are pretty close to the true values. Therefore, increasing the number of data points to $10,000$ does not really show a drastic difference.

- Now suppose that you use $y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + w_5 x^5$ and repeat the above task. Comment on what changed.

> For the code, refer to the ipython notebook. Here we are using a higher order polynomial (of degree 5) instead of degree 3 as in the previous case. With 100 points, one observation we can make is that the estimate for degree 0 coefficient is not good while others are ok (the degree 4 and degree 5 coefficients get closer to zero, while the degree 1, degree 2 and degree 3 get close to the corresponding true values). Upon increasing the number of data points to $10,000$, all the estimates get better and close to the true values.

- Refer to the slides from the class, where we added a prior over $\mathbf{w}$ in order to derive Bayesian linear regression. Assume that we set the hyperparameter $\alpha = 1$ and plot the Bayesian estimate of the curve and the uncertainty around the estimate. How well does it match the observed data. How does the estimate change when you use 1000 or 10,000 data points?

This is mostly a programming exercise and you can see the code in the notebook. We picked the variance of the prior to be $\beta = 1$, even though one could use other values e.g. one could have used β_{MLE}. If we plot the behaviour on the full interval $[-100, 100]$, we cannot see much, so we restrict the plots on the interval $[-3, 3]$.

The difference between the 100 and 10000 data points is that the variance at each point is much smaller in the case of more data. Another important thing to note is that the estimator is much more certain around values where we have already seen some data points, which can also be seen in the plots in the form of a "thinner" uncertainty interval.

- Perform Bayesian linear regression but now set $\alpha = 100$. How does this change your estimates? Again, what happens when you use 1000 or 10,000 data points?

Setting $\alpha = 100$ shifts the mean closer to the value of the ML estimate as can also be seen from the plots. It still holds that a higher number of points will give a better estimate.

Question 3 (1 + 1 + 0.5 + 0.5 points): Let us assume the following generative model for the data:

$$p\left(\mathbf{x}|\mathcal{C}_k\right) = \prod_{i=1}^{D} \mathcal{N}\left(x_i|\mu_{ki}, \sigma_k^2\right)$$

In other words, each coordinate is independently drawn from a 1-d Gaussian distribution, where the variance of the coordinates for a given class k are fixed (σ_k^2), and known.

- Derive the MLE estimate for μ_{ki}.

The likelihood for the entire dataset (assume that there are N data points) can be expressed as:

$$\begin{aligned}\mathcal{L} &= p\left(x, t|\mathcal{C}, \mu, \sigma\right) \\ &= \prod_{n=1}^{N} p\left(x_n|\mathcal{C}_{nk}\right) \cdot \prod_{k=1}^{K} p\left(\mathcal{C}_k\right)^{\mathbb{I}(y_i=\mathcal{C}_k)}\end{aligned}$$

where $\mathbb{I}\left(\cdot\right)$ is an indicator function which equals 1 if the predicate holds, otherwise 0 (i.e: denoting whether data point x_n belongs to class $\mathcal{C}_k$ or not).

Plugging in the given generative model,

$$\mathcal{L} = \prod_{n=1}^{N} \left(\prod_{i=1}^{D} \mathcal{N}\left(x_i | \mu_{ki}, \sigma_k^2\right) \right) \cdot \prod_{k=1}^{K} p\left(\mathcal{C}_k\right)^{\mathbb{I}(y_i = \mathcal{C}_k)}$$

This is equivalent to the following log-likelihood,

$$\begin{aligned}
\log \mathcal{L} &= \sum_{n=1}^{N} \sum_{i=1}^{D} \log \mathcal{N}\left(x_i | \mu_{ki}, \sigma_k^2\right) + \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{I}\left(y_i = \mathcal{C}_k\right) \log p\left(\mathcal{C}_k\right) \\
&= \sum_{n=1}^{N} \sum_{i=1}^{D} \left(-\frac{1}{2} \log 2\pi - \frac{1}{2} \log \sigma_k^2 - \frac{1}{2\sigma_k^2} \left(x_i - \mu_{ik}\right)^2 \right) + \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{I}\left(y_i = \mathcal{C}_k\right) \log p\left(\mathcal{C}_k\right)
\end{aligned}$$

Taking derivative w.r.t μ_{ik} and setting to zero, we obtain,

$$\begin{aligned}
\frac{\partial \log \mathcal{L}}{\partial \mu_{ik}} &= \sum_{n=1}^{N_k} \frac{1}{\sigma_k^2} \left(x_i - \mu_{ik}\right) \\
&= 0
\end{aligned}$$

From above, it follows that,

$$\begin{aligned}
\sum_{n=1}^{N_k} x_i &= \sum_{n=1}^{N_k} \mu_{ik} \\
\Rightarrow \mu_{ik}^{\mathrm{ML}} &= \frac{\sum_{n=1}^{N_k} x_i}{\sum_{n=1}^{N_k} \mathbb{I}\left(y_i = \mathcal{C}_k\right)}
\end{aligned}$$

Step back for a moment and interpret the above result. It means that the estimate obtained of the mean for class k is nothing but the average of the i-th dimension of the data points which are assigned to that class, which makes intuitive sense. ■

- Assume that k (the number of classes) is equal to 2, and derive the decision rule of the Naive Bayes classifier.

The decision rule of Naive Bayes classifier is given by,

$$\begin{aligned}
p\left(\mathcal{C}_1 | x\right) &\geq p\left(\mathcal{C}_2 | x\right) \\
\Rightarrow p\left(x | \mathcal{C}_1\right) p\left(\mathcal{C}_1\right) &\geq p\left(x | \mathcal{C}_2\right) p\left(\mathcal{C}_2\right)
\end{aligned}$$

Denote $p(\mathcal{C}_1) = \pi$ and taking the log on both sides,

$$\log p(x|\mathcal{C}_1) + \log \pi \geq \log p(x|\mathcal{C}_2) + \log(1-\pi)$$
$$\Rightarrow \log p(x|\mathcal{C}_1) - \log p(x|\mathcal{C}_2) + \log\left(\frac{\pi}{1-\pi}\right) \geq 0$$
$$\Rightarrow \log p(x|\mathcal{C}_1) - \log p(x|\mathcal{C}_2) + \log\left(\frac{\pi}{1-\pi}\right) \geq 0$$

Substituting the likelihood normal distribution functional form,

$$\sum_{i=1}^{D} \log\left(-\frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(-\frac{1}{2\sigma_1^2}(x_i - \mu_{i1})^2\right)\right)$$
$$-\sum_{i=1}^{D} \log\left(-\frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left(-\frac{1}{2\sigma_2^2}(x_i - \mu_{i2})^2\right)\right)$$
$$+\log\left(\frac{\pi}{1-\pi}\right) \geq 0$$

Simplifying we obtain,

$$\sum_{i=1}^{D}\left(\frac{\sigma_1^2 - \sigma_2^2}{2\sigma_1^2\sigma_2^2}\right)\cdot x_i^2 + \sum_{i=1}^{D}\left(\frac{\mu_{i1}}{\sigma_1^2} - \frac{\mu_{i2}}{\sigma_2^2}\right)\cdot x_i + \sum_{i=1}^{D}\left(\frac{\mu_{i2}^2}{2\sigma_2^2} - \frac{\mu_{i1}^2}{2\sigma_1^2}\right)$$
$$+\frac{D}{2}\log\left(\frac{\sigma_2^2}{\sigma_1^2}\right) + \log\left(\frac{\pi}{1-\pi}\right) \geq 0$$

As can be observed, the above decision rule for Naive Bayes has a quadratic and a linear term. The quadratic term would disappear when both the classes have equal variances (i.e: $\sigma_1^2 = \sigma_2^2$). ■

- Suppose the misclassification costs are not equal. In particular, if the cost of misclassifying class 1 is 10 times more than the cost of misclassifying class 2, how does the decision rule change?

Let the costs of misclassifying classes $\mathcal{C}_1$ and $\mathcal{C}_2$ be γ_1 and γ_2 respectively. From the problem, we know that $\frac{\gamma_1}{\gamma_2} = 10$.

The risk / loss function $\mathcal{R}(\mathcal{C}_k|k)$ is given by:

$$\mathcal{R}(\mathcal{C}_1|x) = 0 \cdot p(\mathcal{C}_1|x) + \gamma_2 \cdot p(\mathcal{C}_2|x)$$
$$\mathcal{R}(\mathcal{C}_2|x) = \gamma_1 \cdot p(\mathcal{C}_1|x) + 0 \cdot p(\mathcal{C}_2|x)$$

From the above, the decision rule to predict class $\mathcal{C}_k$ will be such that it minimizes the above risk $\mathcal{R}(\mathcal{C}_k|x)$. In other words,

$$\begin{aligned}\mathcal{R}(\mathcal{C}_1|x) &\leq \mathcal{R}(\mathcal{C}_2|x) \\ \gamma_2 \cdot p(\mathcal{C}_2|x) &\leq \gamma_1 \cdot p(\mathcal{C}_1|x) \\ p(\mathcal{C}_1|x) &\geq \frac{1}{10} \cdot p(\mathcal{C}_2|x)\end{aligned}$$

Following the same algebra as in the above sub-problem, we obtain the final rule as,

$$\sum_{i=1}^{D} \left(\frac{\sigma_1^2 - \sigma_2^2}{2\sigma_1^2\sigma_2^2} \right) \cdot x_i^2 + \sum_{i=1}^{D} \left(\frac{\mu_{i1}}{\sigma_1^2} - \frac{\mu_{i2}}{\sigma_2^2} \right) \cdot x_i + \sum_{i=1}^{D} \left(\frac{\mu_{i2}^2}{2\sigma_2^2} - \frac{\mu_{i1}^2}{2\sigma_1^2} \right) + \frac{D}{2} \log \left(\frac{\sigma_2^2}{\sigma_1^2} \right) + \log \left(\frac{\pi}{1-\pi} \right) \geq -\log 10$$

■

- How does the decision rule change when $k = 3$?

In this sub-problem, we consider the cost of misclassifying different classes as being equal. In general for k classes, the decision rule to classify a data point into class $\hat{k}$ can be written as,

$$\hat{k} = \underset{j}{\operatorname{argmax}}\, p(\mathcal{C}_j|x)$$

Plugging in the expression for $p(\mathcal{C}_j|x)$ and expanding (as in the

previous sub-problems) we obtain,

$$\begin{aligned}
\hat{k} &= \underset{j}{\operatorname{argmax}}\, p\left(\mathcal{C}_j|x\right) \\
&= \underset{j}{\operatorname{argmax}}\, p\left(x|\mathcal{C}_j\right) p\left(\mathcal{C}_j\right) \\
&= \underset{j}{\operatorname{argmax}}\, \log p\left(x|\mathcal{C}_j\right) + \log p\left(\mathcal{C}_j\right) \\
&= \underset{j}{\operatorname{argmax}}\, \log \left(\prod_{i=1}^{D} \mathcal{N}\left(x_i|\mu_{ji}, \sigma_j^2\right) \right) + \log p\left(\mathcal{C}_j\right) \\
&= \underset{j}{\operatorname{argmax}} \left(\frac{D}{2} \log \frac{1}{2\pi\sigma_j^2} - \frac{1}{2\sigma_j^2} \sum_{i=1}^{D} \left(x_i - \mu_{ji}^2\right) \right) + \log p\left(\mathcal{C}_j\right)
\end{aligned}$$

Since for this problem $k = 3$,

$$\hat{k} = \underset{j \in \{1,2,3\}}{\operatorname{argmax}} \left(\frac{D}{2} \log \frac{1}{2\pi\sigma_j^2} - \frac{1}{2\sigma_j^2} \sum_{i=1}^{D} \left(x_i - \mu_{ji}^2\right) \right) + \log p\left(\mathcal{C}_j\right)$$

■