Recitation 2
Linear Regression, Naive Bayes, MLE & MAP

10-701: Introduction to Machine Learning

09/22/2023

1 Linear Regression

In this section, we will consider the following linear regression model:

For each data point in $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$,

$$y_i = \boldsymbol{w}^T \boldsymbol{x}_i + \epsilon \text{ where } y_i, \epsilon \in \mathbb{R} \text{ and } \boldsymbol{w}, \boldsymbol{x}_i \in \mathbb{R}^{d+1}$$

In matrix notation, we can express this linear relationship for all data points as:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w} + \boldsymbol{\epsilon} \text{ where } \boldsymbol{y}, \boldsymbol{\epsilon} \in \mathbb{R}^n, \boldsymbol{X} \in \mathbb{R}^{n\times(d+1)}, \text{ and } \boldsymbol{w} \in \mathbb{R}^{d+1}$$

1.1 Ordinary Least Squares (OLS)

In class, we saw that one way to optimize $\boldsymbol{w}$ is to minimize the least squares error:

$$\begin{aligned}\boldsymbol{w}^*_{\text{LS}} &= \arg\min_{\boldsymbol{w}} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}||_2^2 \\ &= \arg\min_{\boldsymbol{w}} (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})\end{aligned}$$

1. Derive the least squares optimal solution $\boldsymbol{w}^*_{\text{LS}}$. You may assume any matrix inversion that naturally appears is possible.

   We can solve by setting the derivative w.r.t $\boldsymbol{w}$ of the minimized statement to 0

$$\frac{\partial}{\partial \boldsymbol{w}}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) = \frac{\partial}{\partial \boldsymbol{w}}(\boldsymbol{y}^T\boldsymbol{y} - (\boldsymbol{X}\boldsymbol{w})^T\boldsymbol{y} - \boldsymbol{y}^T(\boldsymbol{X}\boldsymbol{w}) + (\boldsymbol{X}\boldsymbol{w})^T\boldsymbol{X}\boldsymbol{w})$$

   In this situation, we can look at how to derive and/or combine the matrix products above. For $\boldsymbol{y}^T\boldsymbol{y}$ the derivative is just 0, due to it not involving w. Since $\boldsymbol{X}\boldsymbol{w}$ is size n x 1 and $\boldsymbol{y}$ is size n x 1, $(\boldsymbol{X}\boldsymbol{w})^T\boldsymbol{y}$ and $\boldsymbol{y}^T(\boldsymbol{X}\boldsymbol{w})$ are both scalars so they can just be combined.

$$\begin{aligned}&= \frac{\partial}{\partial \boldsymbol{w}}(\boldsymbol{y}^T\boldsymbol{y} - (\boldsymbol{X}\boldsymbol{w})^T\boldsymbol{y} - \boldsymbol{y}^T(\boldsymbol{X}\boldsymbol{w}) + (\boldsymbol{X}\boldsymbol{w})^T\boldsymbol{X}\boldsymbol{w}) \\ &= \frac{\partial}{\partial \boldsymbol{w}}(-2(\boldsymbol{X}\boldsymbol{w})^T\boldsymbol{y} + (\boldsymbol{X}\boldsymbol{w})^T\boldsymbol{X}\boldsymbol{w})\end{aligned}$$

If we expand out $\boldsymbol{Xw}$, we get a n x 1 vector $[\boldsymbol{x_1 w}^T, \boldsymbol{x_2 w}^T, ..., \boldsymbol{x_n w}^T]$ where x_i is the ith row of x.

Then, expanding $(\boldsymbol{Xw})^T\boldsymbol{Xw}$, we get the scalar $(\boldsymbol{x_1 w}^T)^2 + ... + (\boldsymbol{x_n w}^T)^2$

Deriving by w for each i, we get $2(\boldsymbol{x_i w}^T)x_i$. This scalar sum is equivalent to $2\boldsymbol{X}^T\boldsymbol{Xw}$

Thus, we have the final derivative:

$$\begin{aligned} &= \frac{\partial}{\partial \boldsymbol{w}}(-2(\boldsymbol{Xw})^T\boldsymbol{y} + (\boldsymbol{Xw})^T\boldsymbol{Xw}) \\ &= -2\boldsymbol{X}^T\boldsymbol{y} + 2\boldsymbol{X}^T\boldsymbol{Xw} \end{aligned} \tag{1}$$

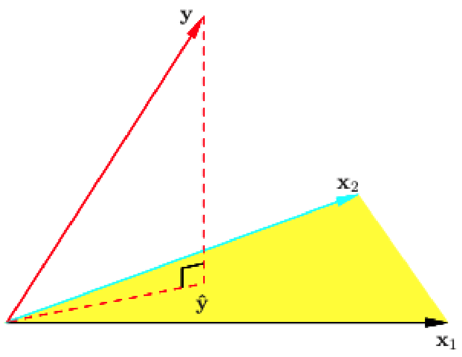
Solving for $-2\boldsymbol{X}^T\boldsymbol{y} + 2\boldsymbol{X}^T\boldsymbol{Xw} = 0$, we get that $\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$

2. Now let us consider the following: In general, when we have some matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^n$, the orthogonal projection of $\boldsymbol{b}$ onto the column space of $\boldsymbol{A}$ can be done using the projection matrix $\boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T$. With this in mind, what can we say about $\boldsymbol{w}^*_{\text{LS}}$?

   Notice that since $\boldsymbol{w}^*_{\text{LS}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$, we have

$$\hat{\boldsymbol{y}} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

   This shows that $\hat{\boldsymbol{y}}$ is an orthogonal projection of $\boldsymbol{y}$ onto the column space of $\boldsymbol{X}$. That is, from a geometric perspective, we are finding the value of $\boldsymbol{w}$ that gives us this orthogonal projection matrix when minimizing the least squares error.



2 Gradient Descent

Gradient descent (GD) is one of the most commonly used optimization algorithms in machine learning. Here we will go over 1) why we are using gradients in the first place and 2) why stochastic gradient descent (SGD) works.

2.1 Gradient Points in the Direction of Steepest Ascent

In class, we saw a pictorial sketch of why gradient descent makes sense; we will discuss it more formally here.

One way of thinking about this is that for a differentiable multivariate function $f : \mathbb{R}^d \to \mathbb{R}$, the gradient at point $\boldsymbol{x} \in \mathbb{R}^d$ (i.e. $\nabla f(\boldsymbol{x})$) points in the direction of *steepest ascent* at $\boldsymbol{x}$.

Recall from calculus that we define the directional derivative with respect to some unit vector $\boldsymbol{u} \in \mathbb{R}^d$ as

$$D_{\boldsymbol{u}} f(\boldsymbol{x}) = \lim_{h \to 0} \frac{f(\boldsymbol{x} + h\boldsymbol{u}) - f(\boldsymbol{x})}{h}$$

In words, the directional derivative describes how the function value instantaneously changes if we step along the direction of $\boldsymbol{u}$ from point $\boldsymbol{x}$. With some calculation, one can show that

$$\begin{aligned} D_{\boldsymbol{u}} f(\boldsymbol{x}) &= \boldsymbol{u}^T \nabla f(\boldsymbol{x}) \\ &= ||\boldsymbol{u}|| ||\nabla f(\boldsymbol{x})|| \cos\theta \ (\because \text{inner product}) \end{aligned}$$

where θ is the angle between $\boldsymbol{u}$ and $\nabla f(\boldsymbol{x})$. We know $||\boldsymbol{u}|| = 1$, and it is easy to see that when $\theta = 0$, $D_{\boldsymbol{u}} f(\boldsymbol{x})$ is maximized.

Thus, at each point, we should step in the direction of the gradient to maximally *increase* the function value (gradient ascent). Meanwhile, we should step in the direction opposite of the gradient to maximally *decrease* the function value (gradient descent). Whether we want to use gradient ascent or descent will depend on whether we want to maximize or minimize some objective function.

2.2 Stochastic Gradient Descent

In lecture, you are introduced with the concept of gradient descent (GD). However, the *stochastic* gradient descent (SGD) is more common in practice than the vanilla GD, as the gradient updates in SGD are computationally cheaper when working with large datasets but still lead to good, generalizable solutions (often even better). An in-depth discussion of SGD is beyond the scope of this course, but here we will see why it makes sense at a high level.

Suppose we have some ML model (e.g., linear regression, logistic regression, neural network) and we want to optimize the parameters $\boldsymbol{w}$ of that model using data $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$. And let's say that our loss function is

$$\mathcal{L}_{\text{total}} = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\boldsymbol{x}_i, \boldsymbol{y}_i)$$

which is an average of the loss $\mathcal{L}$ for each data point $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ across our dataset.

Now, we want to use the following SGD algorithm to iteratively update $\boldsymbol{w}_t$:

1. Randomly sample (without replacement) m indices from $\{1, \ldots, n\}$. Call the set of sampled indices $\mathcal{B}$.
2. Calculate the loss using the sampled data points

   $\tilde{\mathcal{L}} = \frac{1}{m} \sum_{i=1}^{n} \mathcal{L}(\boldsymbol{x}_i, \boldsymbol{y}_i) \underbrace{1[i \in \mathcal{B}]}_{\text{indicator}}$
3. Update $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \eta \frac{\partial \tilde{\mathcal{L}}}{\partial \boldsymbol{w}}$

Based on this setup, answer the question below.

1. In statistics, the bias of an estimator (or bias function) is the difference between this estimator's expected value and the true value of the parameter being estimated.

$$Bias(\hat{\theta}, \theta) = \mathbb{E}_{x|\theta}[\hat{\theta} - \theta]$$

An estimator or decision rule with zero bias is called unbiased. Show that the stochastic gradient is an unbiased estimator of the gradient, i.e. show that:

$$\mathbb{E}\left[\frac{\partial \tilde{\mathcal{L}}}{\partial \boldsymbol{w}}\right] = \frac{\partial \mathcal{L}}{\partial \boldsymbol{w}}$$

(Hint: $\mathbb{E}\big[1[i \in \mathcal{B}]\big] = p(i \in \mathcal{B})$. Also check out the remark below if you need better intuition.)

$$\begin{aligned}\mathbb{E}\left[\frac{\partial \tilde{\mathcal{L}}}{\partial \boldsymbol{w}}\right] &= \frac{1}{m} \sum_{i=1}^{n} \frac{\partial \mathcal{L}(\boldsymbol{x}_i, \boldsymbol{y}_i)}{\partial \boldsymbol{w}} \mathbb{E}\big[1[i \in \mathcal{B}]\big] \\ &= \frac{1}{m} \sum_{i=1}^{n} \frac{\partial \mathcal{L}(\boldsymbol{x}_i, \boldsymbol{y}_i)}{\partial \boldsymbol{w}} \frac{m}{n} \\ &= \frac{1}{n} \sum_{i=1}^{n} \frac{\partial \mathcal{L}(\boldsymbol{x}_i, \boldsymbol{y}_i)}{\partial \boldsymbol{w}} \\ &= \frac{\partial \mathcal{L}}{\partial \boldsymbol{w}}\end{aligned}$$

The fact that the stochastic gradient is an unbiased estimator of the full-batch gradient is an important justification for using SGD. In fact, showing unbiasedness is generally important in many stochastic algorithms.

Remark: Randomly choosing the data points for updating the parameters at each iteration is what makes SGD *stochastic*. Try comparing $\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}}$ and $\frac{\partial \tilde{\mathcal{L}}}{\partial \boldsymbol{w}}$; you should notice that since we are only using a random subset of all of our training points to calculate the gradient, $\frac{\partial \tilde{\mathcal{L}}}{\partial \boldsymbol{w}}$ can be thought of as an approximation of $\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}}$.

3 Naive Bayes

3.1 Review

Simple probabilistic classifier most commonly used for text classification. Called Naive Bayes because it applies Bayes Theorem with a naive assumption of conditional independence: features in $X = (X_1, X_2, \dots, X_d)$ are conditionally independent given the label $Y = \{1, 2, \dots, k\}$, where k are the number of classes. The Naive Bayes model assigns probabilities $p(Y_k|X)$ for each class k.

$$p(Y_k|X) = \frac{p(X|Y_k)p(Y_k)}{p(X)} = \frac{p(X, Y_k)}{p(X)} = \frac{p(X_1, X_2, \dots, X_d, Y_k)}{p(X)}, \text{ by Bayes Thm}$$

$$p(X_1, X_2, \dots, X_d, Y_k) = p(X_1|X_2, \dots, X_d, Y_k)p(X_2, \dots, X_d, Y_k) = p(X_1|\cdot)p(X_2|\cdot)\dots p(X_d|Y_k)$$

After chain rule, apply naive assumption all features in X are conditionally independent, given the label Y_k.

$$p(X_i|X_{i+1}, \dots, X_d, Y_k) = p(X_i|Y_k)$$

$$\text{Thus, } p(Y_k|X) = \frac{p(Y_k, X)}{p(X)} = \frac{p(Y_k)}{p(X)} \prod_{i=1}^{d} p(X_i|Y_k)$$

3.2 Simple Example

Suppose that there are d binary features $(X_1, X_2, \dots, X_d)$. Assume d is even. Consider the following pairing of the d features:

$$(X_1, X_2), (X_3, X_4), \dots, (X_{d-1}, X_d)$$

For each of the above pairs (X_i, X_{i+1}), assume X_i and X_{i+1} are **dependent**. However, assume the 2 pairs (X_i, X_{i+1}) and (X_j, X_{j+1}) are themselves **independent** when $i \neq j$ given the class.

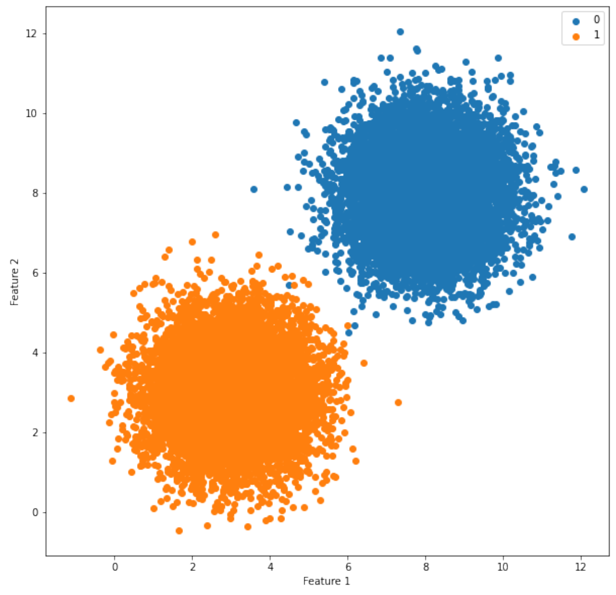
If the class-conditional distribution of each pair of features is known (as well as the class prior), is it possible to construct a classification algorithm using the Naive Bayes approach? If it's possible, how would you do it? If it's not possible, why is it not?

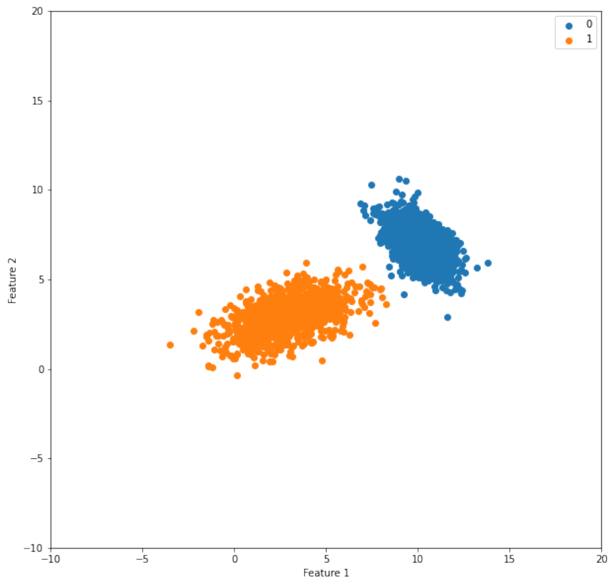
Since the features are all pairwise independent, we can construct a truth table for each of the pairs of features, since the distribution is known. I.e $P(X_1 = 0, X_2 = 0, Y_1 = 1), P(X_1 = 0, X_2 = 1, Y_1 = 1)$, etc. Then we can use the Naive Bayes assumption to calculate $P(Y_1|X_1, ...X_n) = P(X_1, X_2|Y_1)P(X_3, X_4|Y_1)...P(X_{n-1}X_n|Y_1$, where each of the values are known since the distribution of each pair is known. This is analogous to letting a separate random variable represent the distribution X_k, X_{k+1} for each k.

3.3 Gaussian Contour Plots

For a one-dimensional Gaussian, the probability density looks similar to bell curve. For a two-dimensional Gaussian, if both coordinates are independent of one another then the density concentrates in circles. If the two coordinates are not independent, then the density will look elliptical like in the figure above.

For each dataset below, determine if the Naive Bayes assumption is valid. Assume that the data given the class label is distributed as a multivariate Gaussian.





FIRST DATASET: Given that the class label is 1 (all the orange points), we see that the data appears to be distributed circularly, meaning the two coordinates are independent. From our assumption that it is a multivariate Gaussian, we conclude that conditioning on the class label being 1 does indeed make the features independent of one another. The same logic holds for the blue points.

SECOND DATASET: Given that the class label is 1, we see that the coordinates are not independent of one another. The Naive Bayes assumption will build a linear decision boundary assuming that it is a circle, which will diminish our performance. Similar logic holds for the blue points. Covariance matrix has non-zero values for $\text{cov}(x_1, x_2)$ meaning not independent.

3.4 Exam Style Practice Problems

1. In a Naive Bayes problem, suppose we are trying to compute $P(Y|X_1, X_2, X_3, X_4)$. Furthermore, suppose X_2 and X_3 are identical (i.e., X_3 is just a copy of X_2). Finally, assume X_2 is not independent of Y. Which of the following are true in this case? **Select all correct answers.**

   (a) Naive Bayes will learn identical parameter values for $P(X_2|Y)$ and $P(X_3|Y)$.

   (b) Naive Bayes will predict $P(Y|X_1, X_2, X_3, X_4) < P(Y|X_1, X_2, X_4)$.

   (c) Naive Bayes will predict $P(Y|X_1, X_2, X_3, X_4) > P(Y|X_1, X_2, X_4)$.

   (d) None of the above

   (a),(b)

   Naive Bayes will learn identical parameter values for $P(X_2|Y)$ and $P(X_3|Y)$ - this is because in Naive Bayes features are considered independent of each other, hence the estimated probabilities for these identical features will be the same. Naive Bayes will output probabilities $P(Y|X_1, X_2, X_3, X_4)$ that are closer to 0 and 1 than they would be if we removed the feature corresponding to X_3 - consider the expression.

$$P(Y|X_1, X_2, X_3, X_4) = \frac{P(Y, X_1, X_2, X_3, X_4)}{P(X_1, X_2, X_3, X_4)} = \frac{P(X_1|Y)P(X_2|Y)P(X_3|Y)P(X_4|Y)P(Y)}{P(X_1, X_2, X_3, X_4)}$$

   The denominator does not change - as the features are identical, $P(X_1, X_2, X_3, X_4)$ is the same as $P(X_1, X_2, X_4)$. The numerator however is multiplied by a number $P(X_3|Y)$ which is lesser than 1. Hence the output probabilities are different.

2. Gaussian Naive Bayes, in general, can learn non-linear decision boundaries. Consider the simple case where we have just one real-valued feature $X_1 \in \mathbb{R}$ from which we wish to infer the value of label $Y \in \{0, 1\}$. The corresponding generative story would be:

   $Y \sim \text{Bernoulli}(\phi)$
   $X_1 \sim \text{Gaussian}(\mu_y, \sigma_y^2)$
   where the parameters are the Bernoulli parameter ϕ and the class-conditional Gaussian parameters $\mu_0, \sigma_0^2, \mu_1, \sigma_1^2$ corresponding to Y = 0 and Y = 1 , respectively.

   Consider a linear decision boundary in one dimension described by the rule: if $X_1 > c$, then $Y = 1$, else $Y = 0$, where c is a real-valued threshold. Is it possible (in the 1D case) to construct a Gaussian Naive Bayes classifier with a decision boundary that cannot be expressed by a rule in the above form? **Select all correct answers.**

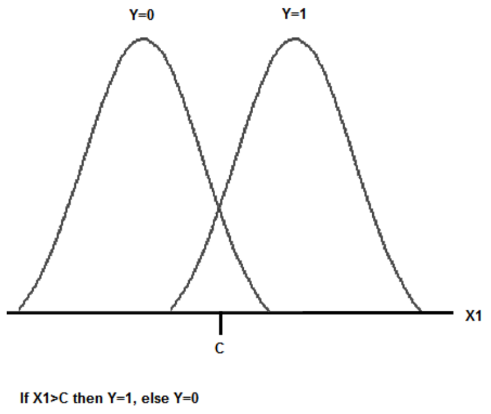
   (a) Yes, this can occur if the Gaussians are of equal means and equal variances.

   (b) Yes, this can occur if the Gaussians are of equal means and unequal variances.

   (c) Yes, this can occur if the Gaussians are of unequal means and equal variances.

   (d) Yes, this can occur if the Gaussians are of unequal means and unequal variances.

(e) No, it is not possible.



(b),(d)

Yes, this can occur if the Gaussians are of equal means and unequal variances. We then have two decision boundaries, as you can see from the figure below. Recall that we choose the class to be the one that gives higher probability at any x.

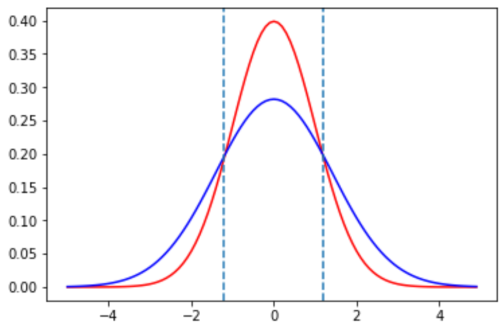


Figure 1: Equal mean, unequal variance

4 MLE and MAP

4.1 Definitions

- Likelihood: $\mathcal{L}(\theta) = \mathbb{P}(\mathcal{D}|\theta)$ and $l(\theta) = \log \mathbb{P}(\mathcal{D}|\theta)$
- Posterior: $\mathbb{P}(\theta|\mathcal{D}) = \frac{\mathbb{P}(\mathcal{D}|\theta)\mathbb{P}(\theta)}{\mathbb{P}(\mathcal{D})} \propto \mathbb{P}(\mathcal{D}|\theta)\mathbb{P}(\theta)$
- MLE estimate: $\theta_{MLE} = \arg\max_\theta \mathbb{P}(\mathcal{D}|\theta) = \arg\max_\theta \log \mathbb{P}(\mathcal{D}|\theta)$
- MAP estimate: $\theta_{MAP} = \arg\max_\theta \mathbb{P}(\mathcal{D}|\theta)\mathbb{P}(\theta) = \arg\max_\theta \log \mathbb{P}(\mathcal{D}|\theta) + \log \mathbb{P}(\theta)$

4.2 Gaussian MLE

Given that we have i.i.d samples $D = \{x_1, ..., x_N\}$, where each point is identically distributed according to a Gaussian distribution, find the MLE for the mean and variance. Hint: $p(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

Solution:

$$\begin{aligned}
\mu^* &= \arg\max_\mu \mathbb{P}(\mathcal{D}|\mu, \sigma) \\
&= \arg\max_\mu \prod_{i=1}^n \mathbb{P}(x_i|\mu, \sigma) \\
&= \arg\max_\mu \prod_{i=1}^n \left(\frac{1}{\sigma\sqrt{2\pi}} \cdot \exp\left(-\frac{(x_i-\mu)^2}{2\sigma^2} \right) \right) \\
&= \arg\max_\mu \left\{ \ln\left(\left(\frac{1}{\sigma\sqrt{2\pi}} \right)^n \right) + \sum_{i=1}^n -\frac{(x_i-\mu)^2}{2\sigma^2} \right\}
\end{aligned}$$

Then we can solve for μ^* by solve for $\partial/\partial\mu$ (since the function is concave)

$$\frac{\partial}{\partial\mu} \left\{ n \ln\left(\frac{1}{\sigma\sqrt{2\pi}} \right) + \sum_{i=1}^n - \left(\frac{(x_i-\mu)^2}{2\sigma^2} \right) \right\} = \frac{1}{2\sigma^2} \cdot 2 \sum_{i=1}^n (x_i - \mu)$$

Hence, we have μ^* be the solution of

$$\sum_{i=1}^n (x_i - \mu) = 0$$

That is, we have $\mu^* = \frac{1}{n} \cdot \sum_{i=1}^n x_i$.

For σ^*, we have something similar here

$$\begin{aligned}\sigma^* &= \arg\max_{\sigma} \mathbb{P}(\mathcal{D}|\mu,\sigma)\\ &= \cdots \ \text{(same as above)}\\ &= \arg\max_{\sigma}\left\{ n\ln\left(\frac{1}{\sigma\sqrt{2\pi}}\right) + \sum_{i=1}^{n}\left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right)\right\}\end{aligned}$$

We can solve σ^* by solving $\partial/\partial\sigma = 0$.

$$\frac{\partial}{\partial\sigma}\left\{ n\ln\left(\frac{1}{\sigma\sqrt{2\pi}}\right) + \sum_{i=1}^{n}\left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right)\right\} = -\frac{n}{\sigma} + \left(\sum_{i=1}^{n}(x_i-\mu)^2\right)\cdot\frac{1}{\sigma^3}$$

Hence, we have the solution of σ^* being the solution of

$$-\frac{n}{\sigma} + \left(\sum_{i=1}^{n}(x_i-\mu)^2\right)\cdot\frac{1}{\sigma^3}$$

That is,

$$(\sigma^*)^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^2$$