CS 181 Spring 2020 Section 2 Notes: Probabilistic Regression, Classification

1 Probabilistic Regression

1.1 Takeaways

1.1.1 Generative Model

"Story" for how the data were created.

For a model parameterized by θ, we have the **likelihood** for data $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $\mathbf{x}_i \in \mathbb{R}^m$, $y_i \in \mathbb{R}$

$$p(D|\theta)$$

The value of θ that maximizes the likelihood is called the maximum likelihood estimate or **MLE**.

We can further add a generative story for θ. This is called the **prior**

$$p(\theta)$$

Given the generative model, we can find the **posterior** distribution for θ

$$p(\theta|D) \propto p(D|\theta)p(\theta)$$

The value θ that maximizes the posterior is called the maximum aposteriori or **MAP** estimate.

1.1.2 Log Trick

When we are maximizing the likelihood or the posterior, often, we apply log on both sides of the equation. This serves to reduce the complexity of the expressions by converting products into summations and removing exponents while resulting in the same solution.

Example:

$$\log p(\theta|D) \propto \log p(D|\theta) + \log p(\theta)$$

We can do this because log is a monotonically increasing function.

1.2 Concept Question

When is MAP the same as MLE?

In addition to the likelihood, MAP incorporates a prior distribution (that quantifies the additional information available through prior knowledge) over θ. When we have no additional information about θ, i.e., we have a uniform prior, the MAP estimate is equal to the MLE.

$$\begin{aligned} \theta_{MAP} &= \text{argmax} \log p(D|\theta) + \log p(\theta) \\ &= \text{argmax} \log p(D|\theta) + constant \\ &= \text{argmax} \log p(D|\theta) \\ &= \theta_{MLE} \end{aligned}$$

2 Linear Classification

2.1 Takeaways

2.1.1 Classification

Goal : Given an input vector $\mathbf{x}$, assign it to one of K dicrete classes C_k.

Input space divided into **decision regions** whose boundaries are called **decision boundaries or surfaces**.

2.1.2 Binary Linear Classification

- Two classes divided by a linear separator in feature space.
- **Discriminant function** : Function that directly assigns each vector to a specific class

$$\hat{y} = \text{sign}(h(\mathbf{x}; \mathbf{w}, w_0)) = \text{sign}(\mathbf{w}^\top \mathbf{x} + w_0)$$

- $\mathbf{w}$ is orthogonal to every point on the decision surface. It determines orientation of decision boundary.

To formulate the loss, we use the hinge loss / rectified linear function:

$$ReLU(z) = \max\{0, z\}$$

2.1.3 Perceptron

- Algorithm used to train the linear discriminant model
- Perceptron Loss :

$$\begin{aligned}\mathcal{L}(\mathbf{w}) &= \sum_{i=1}^{n} ReLU(-h(\mathbf{x}_i; \mathbf{w}, w_0) y_i) \\ &= -\sum_{i=1:y_i \neq \hat{y}_i}^{n} (\mathbf{w}^\top \mathbf{x}_i + w_0) y_i\end{aligned}$$

- Update using gradient descent:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \frac{\partial}{\partial \mathbf{w}} \mathcal{L}^{(i)}(\mathbf{w}) = \mathbf{w}^{(t)} + \eta y_i \mathbf{x}_i,$$

2.2 Concept Question

Why do we choose the $ReLU$ function over the $0/1$ function when formulating the loss function?

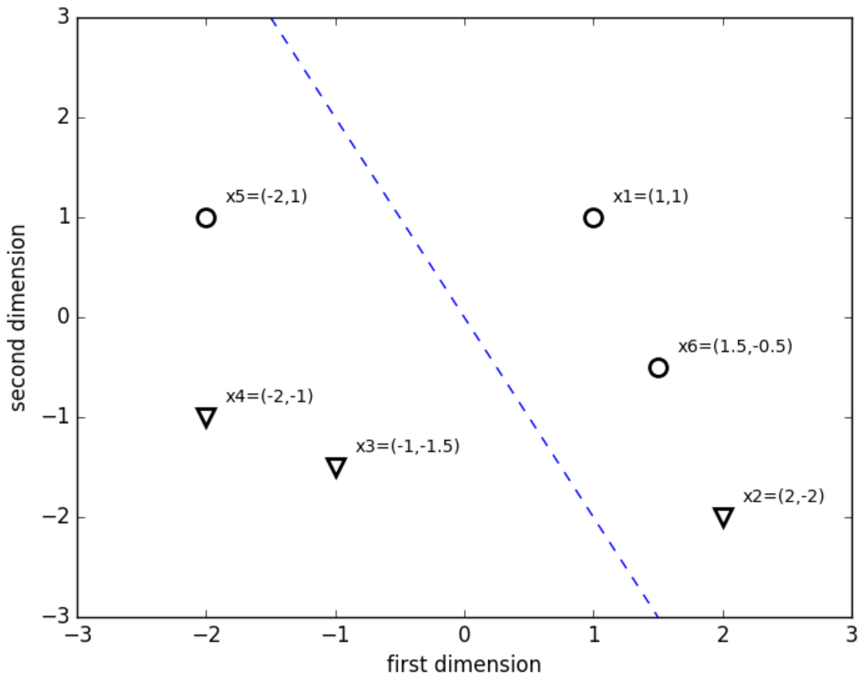
The gradient for the $0/1$ function is uninformative. We are either right or wrong.
$0/1$ is not differentiable at 0 and its derivative is 0 at all other points.

2.3 Exercise: Small Perceptron Example

Let's train a perceptron on a small data set. Consider data $\{\mathbf{x}_i\}_{i=1}^n, \mathbf{x}_i \in \mathbb{R}^2$. Let the learning rate $\eta = 0.2$ and let the weights be initialized as:

$$\mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0.5 \end{pmatrix}, w_0 = 0$$

Let the circles have $y_i = 1$ and the triangles $y_i = -1$. The data and initial separation boundary (determined by $\mathbf{w}$) is illustrated below.



Proceed by iterating over each example until there are no more classification errors. When in doubt, refer to the notes above. We know a priori that we will be able to train the classifier and have no classification errors because one can see visually that the data is linearly separable (note: as mentioned above, if the data were not so obviously linearly separable, a new basis could make it so). How many updates do you have to make? Is this surprising?

Solution:

1. Consider $\mathbf{x_1}$: $\mathbf{w}^\top \mathbf{x_1} + w_0 = 1 \cdot 1 + 0.5 \cdot 1 + 0 > 0$. This is a correct classification, so we take no action.
2. Consider $\mathbf{x_2}$: $\mathbf{w}^\top \mathbf{x_2} + w_0 = 1 \cdot 2 + 0.5 \cdot (-2) + 0 > 0$. This is an incorrect classification, so we need to update our weight parameters:

$$\begin{aligned}\mathbf{w} &\leftarrow \mathbf{w} + (0.2)(-1)\begin{pmatrix} 2 \\ -2 \end{pmatrix} = \begin{pmatrix} 0.6 \\ 0.9 \end{pmatrix} \\ w_0 &\leftarrow w_0 + 0.2(-1) = -0.2\end{aligned}$$

3. Consider $\mathbf{x_3}$: $\mathbf{w}^\top \mathbf{x_3} + w_0 = 0.6 \cdot (-1) + 0.9 \cdot (-1.5) + (-0.2) < 0$. This is a correct classification.
4. Consider $\mathbf{x_4}$: Check that this is a correct classification.
5. Consider $\mathbf{x_5}$: Check that this is an incorrect classification and our weight parameters are updated to:

$$\begin{aligned}\mathbf{w} &\leftarrow \begin{pmatrix} 0.2 \\ 1.1 \end{pmatrix} \\ w_0 &\leftarrow 0\end{aligned}$$

6. Consider $\mathbf{x_6}$: Check that this is again an incorrect classification:

$$\begin{aligned}\mathbf{w} &\leftarrow \begin{pmatrix} 0.5 \\ 1 \end{pmatrix} \\ w_0 &\leftarrow 0.2\end{aligned}$$

All data points are correctly classified now. If the data is linearly separable, we are guaranteed to converge to a solution using the perceptron algorithm in a finite number of steps.

3 Probabilistic Classification

3.1 Takeaways

3.1.1 Discriminative Model

We need to model $p(y|\mathbf{x})$.

In the binary case, we can use the sigmoid function:

$$\sigma(z) = \frac{1}{1 + \exp\{-z\}}$$

and the loss function:

$$\mathcal{L}(\theta) = -\sum_{n=1}^{N} (y_n \log p(y_n = 1|\mathbf{x}_n; \theta) + (1 - y_n) \log p(y_n = 0|\mathbf{x}_n; \theta))$$

where θ are the parameters of the model.

Logistic Regression: Use a linear function

$$z = \mathbf{w}^T \mathbf{x} + w_0$$

3.1.2 Generative Model

Models the joint distribution $p(x, y)$. This factors into $p(x|y)p(y)$.

- $p(y)$ is called the **class prior** and is always a categorical distribution.
  Gives an a priori probability of an observation being a certain class, without even considering the observation's features.
- $p(x|y)$ is called the **class-conditional distribution** and its form is model-specific.
  Specifies how likely an observation (set of features) is given a class.

We are interested in picking the class k that maximizes $p(y = k|\mathbf{x})$.

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} \propto p(\mathbf{x}|y)p(y)$$

$\mathbf{x}$ can be either discrete or continuous.

3.1.3 Naive Bayes

Naive Bayes is one type of generative model for classification.

It's "naive" because we assume that each feature in $\mathbf{x}$ is independently distributed, conditioned on the class.

$$p(\mathbf{x}|y = k) = \prod_{j=1}^{D} p(x_j|y = k)$$

where D is the number of features and k is the class.

We must choose the particular distribution $p(x_j|y)$. Let's choose a Multinomial class-conditional distribution over discrete feature counts.

$$p(\mathbf{x}_i|y=k;\boldsymbol{\pi}_k) = \mathrm{Mu}(\mathbf{x}_i|N_i = \sum_j x_{ij}, \boldsymbol{\pi}_k) = \frac{N_i!}{\prod_{j=1}^D x_{ij}!}\prod_{j=1}^D \pi_{kj}^{x_j} \propto \prod_{j=1}^D \pi_{kj}^{x_j}$$

In this model, each class k has a vector of probabilities over each of D features.

$$\boldsymbol{\pi}_k = [\pi_{k1}, \pi_{k2}, \ldots, \pi_{kD}], \sum_{j=1}^D \pi_{kj} = 1$$

Let's consider just two classes. Let $p(y=1)=\theta$ and $p(y=0)=(1-\theta)$.

During prediction, we pick the k that maximizes $p(y=k|\mathbf{x})$ for an observation $\mathbf{x}$. To be able to do this, we must estimate the parameters of our model from data.

For MLE, we want to find $\theta, \boldsymbol{\pi}_0, \boldsymbol{\pi}_1$ that maximize the log-likelihood:

$$\begin{aligned}
\theta^*, \boldsymbol{\pi}_0^*, \boldsymbol{\pi}_1^* &= \mathrm{argmax}_{\theta,\boldsymbol{\pi}_0,\boldsymbol{\pi}_1} \sum_{i=1}^N \log p(\mathbf{x}_i, y_i) \\
&= \mathrm{argmax}_{\theta,\boldsymbol{\pi}_0,\boldsymbol{\pi}_1} \sum_{i=1}^N (\log p(\mathbf{x}_i|y_i;\boldsymbol{\pi}_0,\boldsymbol{\pi}_1) + \log p(y_i;\theta)) \\
&= \mathrm{argmax}_{\theta,\boldsymbol{\pi}_0,\boldsymbol{\pi}_1} \sum_{i=1}^N \left(\log \prod_{k\in\{0,1\}} p(\mathbf{x}_i|y_i=k;\pi_k)^{\mathbb{I}\{y_i=k\}} + \log \prod_{k\in\{0,1\}} p(y_i=k;\theta)^{\mathbb{I}\{y_i=k\}}\right) \\
&= \mathrm{argmax}_{\theta,\boldsymbol{\pi}_0,\boldsymbol{\pi}_1} \sum_{i=1}^N \left(\log \prod_{k\in\{0,1\}} \left(\prod_{j=1}^D \pi_{kj}^{x_{ij}}\right)^{\mathbb{I}\{y_i=k\}} + \log \prod_{k\in\{0,1\}} \left(\theta^k(1-\theta)^{(1-k)}\right)^{\mathbb{I}\{y_i=k\}}\right) \\
&= \mathrm{argmax}_{\theta,\boldsymbol{\pi}_0,\boldsymbol{\pi}_1} \sum_{i=1}^N \sum_{k\in\{0,1\}} \mathbb{I}\{y_i=k\}\left[\left(\sum_{j=1}^D x_{ij}\log\pi_{kj}\right) + k\log\theta + (1-k)\log(1-\theta)\right]
\end{aligned}$$

The indicator $\mathbb{I}\{y_i=k\}$, the k that multiplies $\log\theta$ and the $1-k$ that multiplies $\log(1-\theta)$ in the last line are all just used for notation tricks to make sure that our likelihood only evaluates $p(\mathbf{x}, y)$ for each $\mathbf{x}$'s true class rather than both classes. This is used a lot, so make sure you see what is going on.

3.2 Concept Question

Why do we use the "naive" (independence) assumption?

We use the "naive" assumption in order to keep the number of parameters small.
Naive bayes requires a number of parameters linear in the number of variables (classes, features).

3.3 Exercise: Shapes of Decision Boundaries I

Consider now a generative model with $K > 2$ classes, and output label $\mathbf{y}$ encoded as a "one hot" vector of length K. We adopt class prior $p(\mathbf{y} = C_k; \boldsymbol{\pi}) = \pi_k$ for all $k \in \{1, \ldots, K\}$ (where π_k is a parameter of the prior). Let $p(\mathbf{x} \,|\, \mathbf{y} = C_k)$ denote the class-conditional density of features $\mathbf{x}$ (in this case for class C_k). Let the class-conditional probabilities be Gaussian distributions

$$p(\mathbf{x} \,|\, \mathbf{y} = C_k) = \mathcal{N}(\mathbf{x} \,|\, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \text{ for } k \in \{1, \ldots, K\}$$

We will predict the class of a new example $\mathbf{x}$ as the class with the highest conditional probability, $p(\mathbf{y} = C_k \,|\, \mathbf{x})$. Luckily, a little bird came to the window of your dorm, and claimed that you can classify an example $\mathbf{x}$ by finding the class that maximizes the following function:

$$f_k(\mathbf{x}) = \log(\pi_k) - \frac{1}{2}\log(|\boldsymbol{\Sigma}_k|) - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k).$$

Derive this formula by comparing two different classes' conditional probabilities. What can we claim about the shape of the decision boundary given this formula?

Solution:

Let's start with the conditional probability:

$$p(\mathbf{y} = C_k \,|\, \mathbf{x}) = \frac{p(\mathbf{x} \,|\, \mathbf{y} = C_k)p(\mathbf{y} = C_k)}{\sum_\ell^c p(\mathbf{x} \,|\, \mathbf{y} = C_\ell)p(\mathbf{y} = C_\ell)}$$

{We can take out the denominator since it will be the same for each class}

$$\begin{aligned}&\propto p(\mathbf{x} \,|\, \mathbf{y} = C_k)p(\mathbf{y} = C_k)\\ &= \frac{1}{(2\pi)^{\frac{m}{2}}|\boldsymbol{\Sigma}_k|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right)\pi_k\end{aligned}$$

{We can factor out constants that will be shared across classes}

$$\propto \frac{\pi_k}{|\boldsymbol{\Sigma}_k|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right)$$

{We take the logarithm, which is a monotonically increasing function and won't change the maximum across classes}

$$\propto \ln(\pi_k) - \frac{1}{2}\ln(|\boldsymbol{\Sigma}_k|) - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)$$

Now, what can we say about the shape of our decision boundary? $(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)$ gives us intuition about the shape of our decision boundary. We see it is quadratic.

For further intuition, we can look at the other two terms to see what affects the probability of an example being classified to a given class. As the prior π_k increases, we see that the probability increases, which fits our intuition. As the covariance matrix $\boldsymbol{\Sigma}_k$ increases, we see that the probability decreases, which also fits our intuition.

3.4 Exercise: Shapes of Decision Boundaries II

Let's say the little bird comes back and now tells you that every class has the same covariance matrix, and so $\boldsymbol{\Sigma}_\ell = \boldsymbol{\Sigma}'_\ell$ for all classes C_ℓ and $C_{\ell'}$. Simplify this formula down further. What can we claim about the shape of the decision boundaries now?

Solution:

Here was our original formula.

$$\ln(\pi_k) - \frac{1}{2}\ln(|\boldsymbol{\Sigma}_k|) - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k) \tag{1}$$

Given that all of the classes have the same covariance matrix, which we will write as $\boldsymbol{\Sigma}$, we have:

$$\begin{aligned}
f_k(\mathbf{x}) &= \ln(\pi_k) - \frac{1}{2}\ln(|\boldsymbol{\Sigma}|) - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k) \\
&= \ln(\pi_k) - \frac{1}{2}\ln(|\boldsymbol{\Sigma}|) - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T (\boldsymbol{\Sigma}^{-1}\mathbf{x} - \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k) \\
&= \ln(\pi_k) - \frac{1}{2}\ln(|\boldsymbol{\Sigma}|) - \frac{1}{2}(\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - \mathbf{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k - \boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\mathbf{x} + \boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k) \\
&= \ln(\pi_k) - \frac{1}{2}\ln(|\boldsymbol{\Sigma}|) - \frac{1}{2}(\mathbf{x}\boldsymbol{\Sigma}^{-1}\mathbf{x}^T - 2\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k) \\
&= \ln(\pi_k) - \frac{1}{2}\ln(|\boldsymbol{\Sigma}|) - \frac{1}{2}\mathbf{x}\boldsymbol{\Sigma}^{-1}\mathbf{x}^T + \mathbf{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k
\end{aligned}$$

{We can drop $-\frac{1}{2}\ln(|\boldsymbol{\Sigma}|)$ and $-\frac{1}{2}\mathbf{x}\boldsymbol{\Sigma}^{-1}\mathbf{x}^T$ since they are class independent}

$$\propto \ln(\pi_k) + \mathbf{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k$$

Thus, we conclude that we can adopt function

$$\tilde{f}_k(\mathbf{x}) = \mathbf{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k^T\Sigma^{-1}\boldsymbol{\mu}_k + \ln(\pi_k) \tag{2}$$

Looking at this formula, we can see that our decision boundaries are no longer quadratic but linear. We've proven that if the classes share the same covariance matrix, our decision boundaries will be linear!