1 Difference to discriminative modeling

1.1 Discriminative modeling

Most machine learning problems that are encountered in the real world are of discriminative nature. Discriminative modeling is a form of *supervised learning*. It aims to estimate a function that "fits" the labeled training data.

One example of such a problem would be: "Given a painting, determine if the painting was made by Banksy or not." For that, the model is given a training set, which contains paintings of Banksy and aintings made by other artists (or totally different images). Each image in the training set has a label, e.g. 1 if the painting was made by Banksy, and 0 if it's something different.
The model is then trained, and can be fed new images which it tries to deduct a probability for it being a painting by Banksy or not.

Discriminative modeling estimates $p(y|x)$

In other words: The probability of a label y given data (observation) x.

1.2 Generative modeling

A generative model describes how the dataset X is generated, in terms of a probabilistic model (some distribution). By sampling from this model, we can generate new data.

Generative modeling estimates $p(x)$

In other words: The probability of observing observation x.

If the dataset is labeled, we can also build a model that estimates $p(x|y)$.

1.3 Difference

The general difference is that a *discriminative model* can only output probabilities of labels against a new datapoint. Even if it has an accuracy of 100% (perfect discriminative model), it can never generate new data.

A *generative model* on the other hand can output a new datapoint, which has a high chance of belonging to the original data set (the model can predict the distribution from which the dataset was generated from really well).

2 Generative modeling basics

2.1 Generative modeling in a nutshell

Given a dataset X (which may or may not be labeled), our goal is the following:

- We have some dataset X
- We assume that the data in X has been generated by some (unknown) distribution p_{data}
- A generative model p_{model} tries to mimic p_{data} as closely as possible. By using the model (distribution) p_{model} we can generate new datapoints $\hat{X}$ that appear to originate from p_{data}
- p_{model} is close enough to the original p_{data} if:
  1. It can generate new datapoints that appear to have been drawn from p_{data}
  2. It can generate examples that are different from the datapoints in X, that is the model shouldn't simply reproduce the datapoints from X.

2.2 Parametric modeling

Because the true distribution of the data p_{data} is unknown, we have to approximate it using a model. However, there are infinitely many models p_{model} that we can use to approximate the original distribution.

To approach the problem in a structured manner, we use *parametric modeling*:

A *parametric model* $p_\theta(x)$ is a family of density functions that can be described (characterized) by a finite number of parameters θ.

For example if we have a 2D plane with rectangle on it, where the probability of a point being in the box is uniform, and a point being outside has probability 0, we can identify each rectangle by four parameters: The coordinates on the bottom-left (θ_1, θ_2) and top-right (θ_3, θ_4).

This means not that every density function $p_\theta(x)$ in this parametric model can be identified by the set of parameters $\theta = \{\theta_1, \theta_2, \theta_3, \theta_4\}$.

2.3 Likelihood

The *likelihood* $\mathcal{L}(\theta|x)$ of a parameter set θ gives us the plausability of the parameter set θ given data point x.

It is defined as

$$\mathcal{L}(\theta|x) = p_\theta(x)$$

In other words: The likelihood of θ given some observed datapoint x is defined to be the value of the probability function parameterized by θ at the point x.

If the likelihood for a given datapoint x if higher for θ_1 that for θ_2 ($\mathcal{L}(\theta_1|x) > \mathcal{L}(\theta_2|x)$), this means that the model characterized by θ_1 is more likely to be close to the real model (at least for the point x).

If we have a dataset X consisting of several independent datapoints, we can write

$$\mathcal{L}(\theta|X) = \prod_{x \in X} p_\theta(x)$$

Which can be rewritten as

$$\mathcal{L}(\theta|X) = \sum_{x \in X} \log(p_\theta(x))$$

Intuitively, we are defining the likelihood of a set of parameters θ to be equal to the probability of seeing the data X under the model parameterized by θ.

2.4 Maximum likelihood estimation (MLE)

Because of the way likelihood was defined in the previous chapter, we want to find a parameter set $\hat{\theta}$ that maximizes the likelihood of observing the dataset X. This is called *maximum likelihood estimation (MLE)*:

$$\hat{\theta} = \underset{\theta}{\text{argmax}}\ \mathcal{L}(\theta|X)$$

Where $\hat{\theta}$ is called the *maximum likelihood estimate*.

3 Naive Bayes Generative model

Suppose we have a dataset with k features, with $\{x_1, x_2, \ldots, x_k\}$ being the features (for each datapoint $x \in X$).

The *naive Bayes assumption* makes the assumption that every feature x_j is independent of every other feature x_h. More formally, for each pair of features x_j, x_h:

$$p(x_j|x_h) = p(x_j)$$

Using the chain rule of probability, we can write the density function as a product of conditional probabilities:

$$p(x) = p(x_1, \ldots, x_k) = p(x_2, \ldots, x_k|x_1)$$

which results in

$$p(x_1) = p(x_3, \ldots, x_k|x_1, x_2) \cdot p(x_2|x_1) \cdot p(x_1) = \prod_{n=1}^{k} p(x_n|x_1, \ldots, x_{n-1})$$

Using the naive bayes rule we can simplify this as:

$$p(x) = \prod_{n=1}^{k} p(x_k)$$

This is the *Naive bayes model*. The problem is reduced to estimating the parameters $\theta_{k,l} = p(x_k = l)$ for each feature seperately and multiplying these to find the probability for any possible combination.

For each feature we need to estimate a parameter for each value that the feature can take. Therefore, if feature x_k can take h_k values, we have a total of $\sum_{n=1}^{k} h_k$ parameters we need to estimate.

The maximum likelihood estimation estimates $\hat{\theta}_{k,l}$ as follows:

$$\hat{\theta}_{k,l} = \frac{n_{kl}}{N}$$

where n_{kl} is the number of times that the feature k takes on the value l in the dataset and N is the total number of observations.

3.1 Probability of observation x

After calculating the probability of each feature with the *naive Bayes model*, we can calculate the probability of a new (unseen) datapoint $\hat{x}$ by multiplying the probabilities of all the k features of the new feature:

$$p(\hat{x}) = p(x_1, x_2, \ldots, x_k) = \prod_{n=1}^{k} p(x_n)$$

where $p(x_k)$ is the probability of the k-th feature occuring the way we chose it for $\hat{x}$.

4 Gaussian Bayes classifier

4.1 Gaussian Naive Bayes classifier

As a generative model tries to infer the process according to which examples are generated, we can also use this to classify data. The typical approach is:

1. Estimate the prior on labels $P(y)$
2. Estimate the conditional distribution $P(x|y)$ for each class y
3. Obtain predictive distribution $P(y|x)$ using the Bayes' rule:

$$P(y|x) = \frac{1}{Z}P(y) \cdot P(x|y) \text{ where } Z = \sum_y P(y) \cdot P(x|y)$$

In other words: First generate class label $P(y)$, and then generate features given the class $P(x|y)$.

Here, we model the *class label* as generated from *categorical* variable:

$$P(Y = y) = p_y \qquad y \in Y = \{1, \ldots, c\}$$

The d *features* are then modeled as *conditionally independent* given Y:

$$P(X_1 \ldots, X_d|Y) = \prod_{i=1}^{d} P(X_i|Y)$$

I.e. given a class label, each feature is "generated" independently of the other features. However, we still need to specify the feature distributions $P(X_i|Y)$.

We can do that by modeling the features as *(conditionally) independent Gaussians*:

$$P(x_i|y) = \mathcal{N}(x_i; \mu_{y,i}, \sigma^2_{y,i})$$

Where $\mu_{y,i}$ and $\sigma^2_{y,i}$ depend on class y and feature i.

Using the naive Bayes approach, we can calculate these as follows:

Given *training data* $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, calculate:

MLE for class prior: $\hat{P}(Y = y) = \hat{p}_y = \frac{Count(Y=y)}{n}$ (how likely is class label y)

MLE for feature distribution: $\hat{P}(x_i|y) = \mathcal{N}(x_i; \hat{\mu}_{y,i}, \sigma^2_{y,i})$:

$$\hat{\mu}_{y,i} = \frac{1}{Count(Y = y)} \sum_{j:y_y=y} x_{j,i}$$

where $x_{j,i}$ is the value of feature i for instance j (x_j, y_j)

$$\sigma^2_{y,i} = \frac{1}{Count(Y = y)} \sum_{j:y_y=y} (x_{j,i} - \hat{\mu}_{y,i})^2$$

We can then use these to make a prediction for a new point x:

$$y = \underset{y'}{\text{argmax}}\ \hat{P}(y'|x) = \underset{y'}{\text{argmax}}\ P(y') \prod_{i=1}^{d} \hat{P}(x_i|y')$$

4.2 Decision rules for binary classification

As stated before, we want to predict $y = \underset{y'}{\text{argmax}}\ P(y'|x)$.

For a binary classification task (i.e. $c = 2,\ y \in \{+1, -1\}$) this is equivalent to

$$Y = \text{sign}\left(\log \frac{P(Y=1|x)}{P(Y=-1|x)}\right)$$

which gives $+1$ if $P(Y=1|x) > 0.5$ and -1 otherwise.

The function $f(x) = \log \frac{P(Y=1|x)}{P(Y=-1|x)}$ is called the *discriminant function.*

4.3 Generalized MLE for Gaussian Naive Bayes Classifier

As before, in section 4, we have a datamatrix x and label vector y. Again, the labels are generated from categorical variable $P(Y = y) = p_y$.
And again, we model the features as generated by a multivariate Gaussian:

$$P(x|y) = \mathcal{N}(x; \mu_y, \Sigma_y)$$

where μ_y is the mean and Σ_y is the covariance matrix:

$$\Sigma_y = \begin{pmatrix} \sigma_{y,1}^2 & & \\ & \ddots & \\ & & \sigma_{y,d}^2 \end{pmatrix}$$

As before, the MLE for class label distribution is given by: $\hat{P}(Y = y) = \hat{p}_y = \frac{Count(Y=y)}{n}$

The MLE for feature distribution is now $\hat{P}(x|y) = \mathcal{N}(x; \hat{\mu}_y, \hat{\Sigma}_y)$ with:

$$\hat{\mu}_y = \frac{1}{Count(Y=y)} \sum_{i:y_i=y} x_i$$

$$\hat{\Sigma}_y = \frac{1}{Count(Y=y)} \sum_{i:y_i=y} (x_i - \hat{\mu}_y)(x_i - \hat{\mu}_y)^T$$

To now use this model as a binary classifier ($y \in \{+1, -1\}$), we can write the discriminant function $f(x) = \log \frac{P(Y=1|x)}{P(Y=-1|x)}$ as:

$$f(x) = \log \frac{p}{1-p} + \frac{1}{2}\left[\log \frac{|\hat{\Sigma}_-|}{|\hat{\Sigma}_+|} + \left((x - \hat{\mu}_-^T)\hat{\Sigma}_-^{-1}(x - \hat{\mu}_-)\right) - \left((x - \hat{\mu}_+^T)\hat{\Sigma}_+^{-1}(x - \hat{\mu}_+)\right)\right]$$

With this, we can predict the sign $f(x)$ for $Y : \{+1, -1\}$. This is also called a **Quadratic discriminant analysis**

4.4 Fisher's linear discriminant analysis (LDA)

When we have $c = 2$ (number of labels), we fix $p = 0.5$ and we assume the covariances are equal ($\hat{\Sigma}_- = \hat{\Sigma}_+ = \hat{\Sigma}$), we can simplify the discriminant function $f(x)$:

$$f(x) = x^T\hat{\Sigma}^{-1}(\hat{\mu}_+ - \hat{\mu}_-) + \frac{1}{2}(\hat{\mu}_-^T\Sigma^{-1}\hat{\mu}_- - \hat{\mu}_+^T\Sigma^{-1}\hat{\mu}_+)$$

With these assumptions, we predict:

$$y = \text{sign}(f(x)) = \text{sign}(w^Tx + w_0) \qquad w = \hat{\Sigma}^{-1}(\hat{\mu}_+ - \hat{\mu}_-) \quad w_0 = \frac{1}{2}(\hat{\mu}_-^T\Sigma^{-1}\hat{\mu}_- - \hat{\mu}_+^T\Sigma^{-1}\hat{\mu}_+)$$

This linear classifier is called *Fisher's linear discriminant analysis.*

As Fisher's LDA uses the discriminant function $f(x) = \log \frac{P(Y=1|x)}{P(Y=-1|x)}$, we can derive the class distribution:

$$P(Y = 1|x) = \frac{1}{1 + \exp(-f(x))} = \sigma(w^Tx + w_0)$$

which is the same form as logistic regression. This means that if model assumptions are met, LDA will make the same predictions as Logistic Regression.

5 Discrete Features

So far all features were $x \in \mathbb{R}^d$. However, if we suppose some X_i take discrete values, such as Gender, Nationality, etc., it might not make sense to model X_i as a Gaussian.

In that case, the generative models allow to easily swap the Gaussian distribution to different distributions, e.g. model $P(X_i|Y)$ as Bernoulli, Categorical, Multinomial, ect.

5.1 MLE for Categorial Naive Bayes Classifier

Model class labels as generated from categorical variable:

$$P(Y = y) = p_y \qquad y \in Y = \{1, \dots, c\}$$

Model features by conditionally independent categorical random variables:

$$P(X_i = c|Y = y) = \theta_{c|y}^{(i)}$$

Estimation of the parameters are as follows:

Given data set $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$

MLE for class label distribution $\hat{P}(Y = y) = \hat{p}_y = \frac{Count(Y=y)}{n}$

MLE for distribution of feature i $\hat{P}(X_i = c|y) = \theta_{c|y}^{(i)} = \frac{Count(X_i=c,Y=y)}{Count(Y=y)}$

Prediction for a new point x:

$$y = \underset{y'}{\text{argmax}}\hat{P}(y'|x) = \underset{y'}{\text{argmax}}\hat{P}(y')\prod_{i=1}^{d}\hat{P}(x_i|y')$$

5.2 Discrete and continuous features

The (naive) Bayes classifier does not require each feature to follow the same type of conditional distribution. For example, can model some features as Gaussian and some other as categorical.

The training (MLE) and prediction remains the same.

6 Overfitting

As MLE is prone to overfitting, we need a strategy to avoid overfitting. This can be done by:

- Restricting the model class (e.g. assumptions on covariance structure, Gaussian Naive Bayes) → fewer parameters
- Using priors → "smaller" parameters

6.1 Prior over parameters

For this subsection, we have $c = 2$ (number of labels).

For the prior of our class probabilities, we have assumed that $P(Y = 1) = \theta$. The MLE for this gives us $\hat{\theta} = \frac{Count(Y=1)}{n}$. In the extreme case that $n = 1$ (only one datapoint), this is an extremely inaccurate approximation of the whole distribution.

In that case, we may want to put the prior distribution $P(\theta)$ and compute the posterior distribution $P(\theta|y_1, \ldots, y_n)$ (i.e. if we already know the prior distribution, e.g. we know that we recieve a lot of spam, so the label "Spam" has a rather high probability).

The posterior distribution is:

$$P(\theta|z_1, \ldots, z_n) = \frac{1}{Z} P(\theta) \prod_{i=1}^{n} P(y_i|\theta)$$

where $P(\theta)$ is the prior, $P(y_i|\theta)$ is the likelihood (assuming the data is independent) and

$$Z = \int P(\theta) \prod_{i=1}^{n} P(y_i|\theta) \; d\theta$$

In other words: We believe that the distribution follows some probability $P(\theta)$ (learned from previous data, assumptions, etc.) and model the distribution in such a way.

6.2 Beta prior over parameters

Again, we consider a binary classification task ($c = 2$). We can use the *Beta Prior distribution* to model the prior distribution.

The *Beta Prior* has two parameters: α_+ and α_-. It is defined as follows:

$$Beta(\theta; \alpha_+, \alpha_-) = \frac{1}{B(\alpha_+, \alpha_-)} \; \theta^{\alpha_+ - 1} \; (1 - \theta)^{\alpha_- - 1}$$

where $B(\alpha_+, \alpha_-)$ is just a regularization function such that the integral over the function in the interval $[0, 1]$ equals 1 (such that the function is a distribution function).

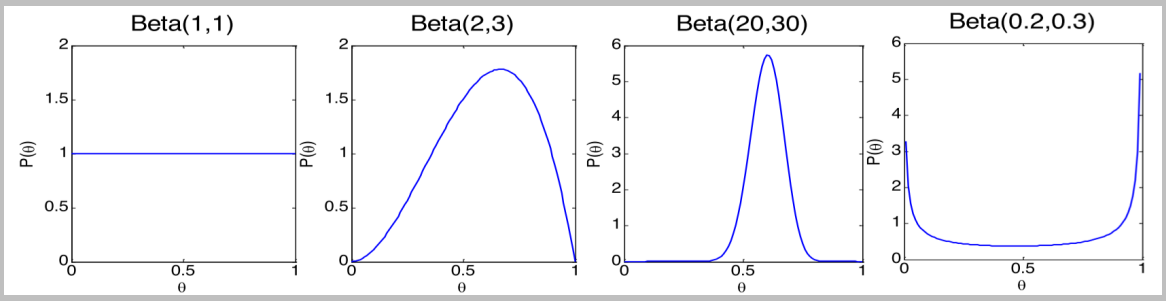

Fig.1: Beta Prior with different parameters. As α_+ and α_- get larger, the function peaks more extreme.$\alpha_+ = \alpha_- = 1$ we have an uniform distribution.

6.3 Conjugate distributions

A pair of prior distributions and likelihood functions is called *conjugate* if the posterior distribution remains in the same distribution family as the prior.

Example: Beta Priors and Binomial likelihood:

Prior: $Beta(\theta; \alpha_+, \alpha_-)$
Observations: Suppose we observe n_+ positive and n_- negative labels
Posterior: $Beta(\theta; \alpha_+ + n_+, \alpha_- + n_-)$

α_+ and α_- act as *pseudo-counts*.

MAP estimate:

$$\hat{\theta} = \underset{\theta}{\text{argmax}}\ P(\theta | y_1, \ldots, y_n; \alpha_+, \alpha_-) = \frac{\alpha_+ + n_+ - 1}{\alpha_+ + n_+ + \alpha_- + n_- - 2}$$

We have the following list of conjugated priors:

Prior / Posterior	**Likelihood function**
Beta	Bernoulli/Binomial
Dirichlet	Categorical / Multinomial
Gaussian (fixed covariance)	Gaussian
Gaussian-inverse Wishart	Gaussian
Gaussian process	Gaussian

7 Sources

- Generative Deep Learning by D. Foster (https://www.oreilly.com/library/view/generative-deep-learning/9781492041931/ch01.html)
- Introduction to Machine Learning Class at ETH (Prof A. Krause) (https://las.inf.ethz.ch/teaching/introml-s19)

7.1 Images

1. Introduction to Machine Learning, Lecture on the 14.05.2019, Page 34