

The within class variance of the transformed data from class C_k is given by

$$S_k^2 = \sum_{n \in C_k} (y_n - m_k)^2 \quad \text{where } y_n = w^T x_n$$

Fisher criterion is defined to be the ratio of the between class variance to the within class variance,

$$J(w) = \frac{(m_2 - m_1)^2}{S_1^2 + S_2^2}$$

$$\begin{aligned} & (m_2 - m_1)^2 \\ &= (w^T m_2 - w^T m_1)^2 \\ &= w^T [(m_2 - m_1)(m_1 - m_2)]^T w \\ &= w^T S_B w \end{aligned}$$

$$\begin{aligned} S_1^2 &= \sum_{n \in C_1} (w^T x_n - m_1)^2 \\ &= \sum_{n \in C_1} w^T [(x_n - m_1)(x_n - m_1)^T]^T w \\ &= w^T S_W w \\ S_2^2 &= w^T S_W w \quad \text{where } S_W = S_1 + S_2 \end{aligned}$$

$$S_1, J(w) = \frac{w^T S_B w}{w^T S_W w}$$

$$\frac{\partial J(w)}{\partial w} = \frac{(w^T S_W w) 2S_B w - (w^T S_B w) 2S_W w}{(w^T S_W w)^2} \geq 0$$

$$\Rightarrow S_B w - J(w) S_W w = 0 \quad \Rightarrow S_B^{-1} S_B w - S_W^{-1} S_W w = 0$$

scalar
so we can drop this

$$\Rightarrow w = S_W^{-1} S_B w$$

$$S_W^{-1} w \propto S_B^{-1} (m_1 - m_2)$$

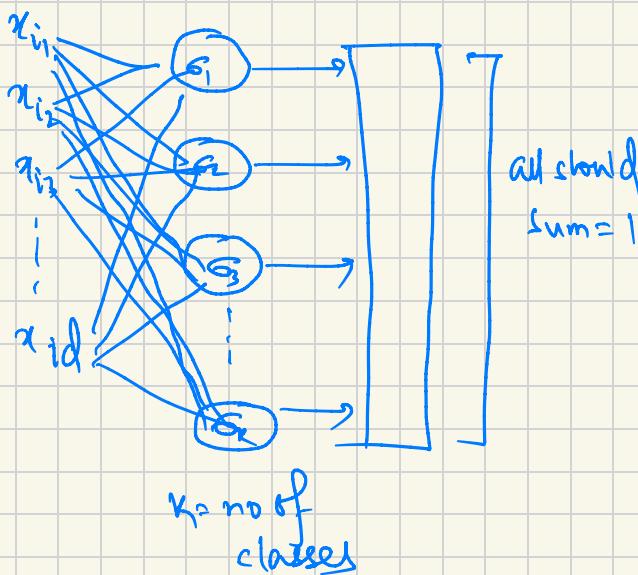
Multi Class Classification

$$P(y=c | x, w) = \frac{\exp(w_c^T x)}{\sum_{c=1}^C \exp(w_c^T x)}$$

log likelihood

$$L(w) = \log \prod_{i=1}^N \prod_{c=1}^C y_{ic} \log p_{ic}$$

this is cross entropy error function for multi class classification.



$$\sigma_j(z_j) = \frac{e^{z_j}}{\sum_{i=1}^K e^{z_i}}$$

where $z_j = \sum_{j=1}^d w_{ij} x_{ij}$

23 min

probabilistic Generative Models

$$P(G_1 | x) = \frac{P(x|G_1) P(G_1)}{P(x|G_1) P(G_1) + P(x|G_2) P(G_2)}$$

$$= \frac{1}{1 + \exp(-\alpha)} \quad \text{where } \alpha = \ln \frac{P(x|G_1) P(G_1)}{P(x|G_2) P(G_2)}$$

$$= \sigma(\alpha)$$

$$P(G_k | x) = \frac{P(x|G_k) P(G_k)}{\sum_{k=1}^K P(x|G_k) P(G_k)}$$

This is for binary classification.
for multiclass classification
we have below form
of the logit equation.

$$P(a_k | x) \Rightarrow \frac{\exp(\alpha_k)}{\sum_j \exp(\alpha_j)}$$

$$\alpha_k = \ln P(x|G_k) P(G_k)$$

→ similar to softmax function we have seen in multiclass classification where we have taken $w^T x_i$ inside exp function.

$P(x|y)$ is class conditioned distribution.

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

Property:

$$\sigma(-a) = 1 - \sigma(a)$$

$$\text{Inverse } a = \ln\left(\frac{\sigma}{1-\sigma}\right)$$

if $\sigma(a) = p(G_1 | x)$

$$a = \ln\left(\frac{p(G_1 | x)}{1 - p(G_1 | x)}\right)$$

$$a = \ln\left(\frac{p(G_1 | x)}{p(G_2 | x)}\right)$$

This is
also known
as **logit**
function.

It represents the log
of the ratio of probability
of two classes. It also

known as **log odds**.

$$p(x|c_k) \sim N(x; \mu_k, \Sigma_k)$$

$x \in \mathbb{R}^d$, $\mu = d \times 1$ mean vector

$\Sigma = \text{covariance matrix}$
($d \times d$) $|\Sigma| = \text{determinant}$

$$\frac{1}{\sqrt{(2\pi)^d |\Sigma|}} e^{-\frac{(x-\mu)^T \Sigma^{-1} (x-\mu)}{2}}$$

$$p(x|\mu, \Sigma_k) = \frac{1}{\sqrt{2\pi)^d |\Sigma|}} \exp \left(-\frac{1}{2} (x-\mu)^T \Sigma_k^{-1} (x-\mu) \right)$$

$$P(G|x) = \sigma(w^T x + w_0)$$

$$w_2 \in \Sigma^{-1}(\mu_1 - \mu_2)$$

$$w_0 = -\frac{1}{2} \mu_1^T \Sigma^{-1} \mu_1 + \frac{1}{2} \mu_2^T \Sigma^{-1} \mu_2 + \ln \frac{P(G)}{P(\bar{G})}$$

$$P(G|x) = \sigma \left(\ln \frac{P(x|G) P(G)}{P(x|\bar{G}) P(\bar{G})} \right) =$$

$$\ln \frac{\frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp \left\{ -\frac{1}{2} (x-\mu_1)^T \Sigma^{-1} (x-\mu_1) \right\}}{\frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp \left\{ -\frac{1}{2} (x-\mu_2)^T \Sigma^{-1} (x-\mu_2) \right\}} + \ln \frac{P(G)}{P(\bar{G})}$$

<https://regenerativetoday.com/univariate-and-bivariate-gaussian-distribution-clear-explanation-with-visuals/>

https://mmuratarat.github.io/2019-10-05/univariate-multivariate_gaussian#:~:text=Like%20the%20univariate%20normal%20distribution,they%20change%20together.

$$\ln \frac{\frac{1}{\sqrt{2\pi}\sigma} \cdot \frac{1}{\sqrt{|\Sigma|}} \exp \left\{ -\frac{1}{2} (x - \mu_1)^T \Sigma^{-1} (x - \mu_1) \right\}}{\frac{1}{\sqrt{2\pi}\sigma} \cdot \frac{1}{\sqrt{|\Sigma|}} \exp \left\{ -\frac{1}{2} (x - \mu_2)^T \Sigma^{-1} (x - \mu_2) \right\}} + \ln \frac{p(y)}{p(x)}$$

$$\frac{x^T \Sigma^{-1} x}{2} = q.$$

$$\Rightarrow \ln \frac{\exp \left\{ -\frac{1}{2} (x^T - \mu_1^T) \Sigma^{-1} (x - \mu_1) \right\}}{\exp \left\{ -\frac{1}{2} (x^T - \mu_2^T) \Sigma^{-1} (x - \mu_2) \right\}}$$

$$\Rightarrow \ln \frac{\exp \left\{ -\frac{1}{2} [x^T \Sigma^{-1} x - \mu_1^T \Sigma^{-1} x - x^T \Sigma^{-1} \mu_1 + \mu_1^T \Sigma^{-1} \mu_1] \right\}}{\exp \left\{ -\frac{1}{2} [x^T \Sigma^{-1} x - \mu_2^T \Sigma^{-1} x - x^T \Sigma^{-1} \mu_2 + \mu_2^T \Sigma^{-1} \mu_2] \right\}}$$

$$\Rightarrow \ln \exp \left\{ \frac{1}{2} \left[\cancel{x^T \Sigma^{-1} x} + \cancel{\mu_1^T \Sigma^{-1} x} + x^T \Sigma^{-1} \mu_1 - \mu_1^T \Sigma^{-1} \mu_1 \right. \right. \\ \left. \left. - \cancel{x^T \Sigma^{-1} x} - \cancel{\mu_2^T \Sigma^{-1} x} - x^T \Sigma^{-1} \mu_2 + \mu_2^T \Sigma^{-1} \mu_2 \right] \right\}$$

$$\Rightarrow \ln \exp \left\{ \frac{1}{2} \left[(\mu_1^T - \mu_2^T) \Sigma^{-1} x + (\mu_1 - \mu_2) \Sigma^{-1} x^T \right. \right. \\ \left. \left. - (\mu_1^T \Sigma^{-1} \mu_1 - \mu_2^T \Sigma^{-1} \mu_2) \right] \right\}$$

$$\Rightarrow \frac{1}{2} \left(\underset{\partial x^T}{\cancel{\Sigma^{-1}(\mu_1 - \mu_2)^T x}} + \underset{\partial x^T}{\cancel{\Sigma^{-1}(\mu_1 - \mu_2) x^T}} \right. \\ \left. - \mu_1^T \Sigma^{-1} \mu_1 + \mu_2^T \Sigma^{-1} \mu_2 \right)$$

$$w = \Sigma^{-1}(\mu_1 - \mu_2)$$

$$w_0 = -\frac{1}{2} \mu_1^T \Sigma^{-1} \mu_1 + \frac{1}{2} \mu_2^T \Sigma^{-1} \mu_2$$

For multiclass classification

$$P(c_k|x) \Rightarrow \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

$$a_k = \ln P(x|c_k) P(c_k)$$

$$a_k(x) = w_k^T x + w_{k0}$$

$$w_k = \Sigma^{-1} u_k$$

$$w_{k0} = -\frac{1}{2} u_k^T \Sigma^{-1} u_k + \ln P(c_k)$$

If covariance is not shared

Quadratic terms of x will be present.

It is also called quadratic discriminant analysis

$$P(y=c|x, w) = \frac{\frac{f_c}{\sqrt{2\pi\Sigma_c}} \exp\left[-\frac{1}{2}(x-u_c)^T \Sigma_c^{-1} (x-u_c)\right]}{\sum_c \frac{f_c}{\sqrt{2\pi\Sigma_c}} \exp\left[-\frac{1}{2}(x-u_c')^T \Sigma_c'^{-1} (x-u_c')\right]}$$

parameter estimation technique

$$P(x_n, q) = P(q) P(x_n|q) = \pi N(x_n | \mu_1, \Sigma)$$

$$P(x_n, q) = P(q) P(x_n|q) = (1-q) N(x_n | \mu_2, \Sigma)$$

$$P(y|q, \mu_1, \mu_2, \Sigma) = \prod_{n=1}^N \left[P N(x_n | \mu_1, \Sigma) \right]^{y_n} \left[(1-p) N(x_n | \mu_2, \Sigma) \right]^{(1-y_n)}$$

It is convenient to maximize the log of the likelihood function.

$$\ln P(y|q, \mu_1, \mu_2, \Sigma) = \sum_{n=1}^N \ln \left[P N(x_n | \mu_1, \Sigma) \right]^{y_n} + \ln \left[(1-p) N(x_n | \mu_2, \Sigma) \right]^{(1-y_n)}$$

$$\Rightarrow \sum_{n=1}^N y_n \ln p \left[-\frac{1}{2} (x_n - \mu_1)^T \Sigma^{-1} (x_n - \mu_1) \right]$$

$$(1-y_n) \ln(1-p) \left[-\frac{1}{2} (x_n - \mu_2)^T \Sigma^{-1} (x_n - \mu_2) \right]$$

$$L \rightarrow \sum_{n=1}^N y_n \ln p + (1-y_n) \ln(1-p)$$

$$\frac{\partial L}{\partial p} = \frac{y_n}{p} - \frac{(1-y_n)}{1-p} = 0$$

$$y_n - \cancel{y_n/p} - p + \cancel{y_n/p} = 0$$

$$p = \sum_{n=1}^N y_n$$

$$\frac{\partial L}{\partial \mu_i} \Rightarrow \sum_{n=1}^N y_n \left[-\frac{1}{2} (x_n - \mu_i)^T \Sigma^{-1} (x_n - \mu_i) \right]$$

$$= \sum_{n=1}^N y_n \left(-\frac{1}{2} \Sigma^{-1} (x_n - \mu_i) \Sigma^{-1} \right)$$

$$= \sum_{n=1}^N y_n \left[+ (x_n - \mu_i) \Sigma^{-1} \right]$$

$$\Rightarrow \sum_{n=1}^N y_n (x_n - \mu_i) \Sigma^{-1} = 0$$

$$\sum_{n=1}^N y_n (x_n - \mu_i) = 0$$

$$\boxed{\mu_i = \frac{1}{N} \left(\sum_{n=1}^N x_n y_n \right)}$$

$$\begin{aligned} \frac{\partial L}{\partial \mu} &= -\frac{1}{2} \left(\frac{\partial (y - \mu)^T \Sigma^{-1} (y - \mu)}{\partial \mu} \right) \\ &= -\frac{1}{2} (-2 \Sigma^{-1} (y - \mu)) \\ &= \Sigma^{-1} (y - \mu) \end{aligned}$$

PUBLIC
Questions
Tags
Users
Unanswered

TEAMS

Stack Overflow for Teams – Start collaborating and sharing organizational knowledge.



Create a free Team
[Why Teams?](#)

83

In chapter 2 of the [Matrix Cookbook](#) there is a nice review of matrix calculus stuff that gives a lot of useful identities that help with problems one would encounter doing probability and statistics, including rules to help differentiate the multivariate Gaussian likelihood.

If you have a random vector \mathbf{y} that is multivariate normal with mean vector $\boldsymbol{\mu}$ and covariance matrix Σ , then use equation (86) in the matrix cookbook to find that the gradient of the log likelihood L with respect to $\boldsymbol{\mu}$ is

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{\mu}} &= -\frac{1}{2} \left(\frac{\partial (\mathbf{y} - \boldsymbol{\mu})' \Sigma^{-1} (\mathbf{y} - \boldsymbol{\mu})}{\partial \boldsymbol{\mu}} \right) \\ &= -\frac{1}{2} (-2 \Sigma^{-1} (\mathbf{y} - \boldsymbol{\mu})) \\ &= \Sigma^{-1} (\mathbf{y} - \boldsymbol{\mu}) \end{aligned}$$

I'll leave it to you to differentiate this again and find the answer to be $-\Sigma^{-1}$.

As "extra credit", use equations (57) and (61) to find that the gradient with respect to Σ is

$$\begin{aligned} \frac{\partial L}{\partial \Sigma} &= -\frac{1}{2} \left(\frac{\partial \log(|\Sigma|)}{\partial \Sigma} + \frac{\partial (\mathbf{y} - \boldsymbol{\mu})' \Sigma^{-1} (\mathbf{y} - \boldsymbol{\mu})}{\partial \Sigma} \right) \\ &= -\frac{1}{2} (\Sigma^{-1} - \Sigma^{-1} (\mathbf{y} - \boldsymbol{\mu}) (\mathbf{y} - \boldsymbol{\mu})' \Sigma^{-1}) \end{aligned}$$

I've left out a lot of the steps, but I made this derivation using only the identities found in the matrix cookbook, so I'll leave it to you to fill in the gaps.

I've used these score equations for maximum likelihood estimation, so I know they are correct :)

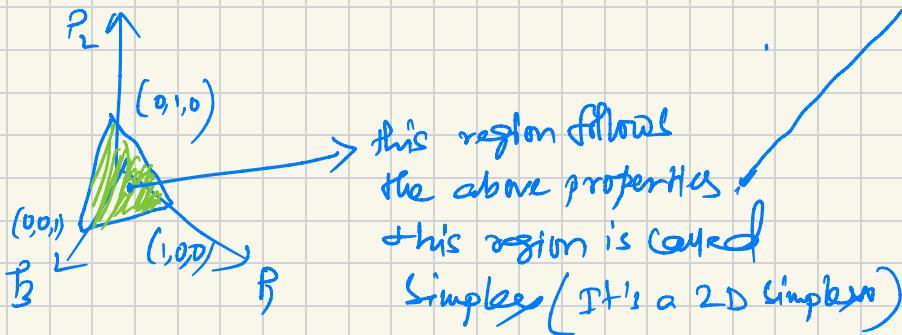
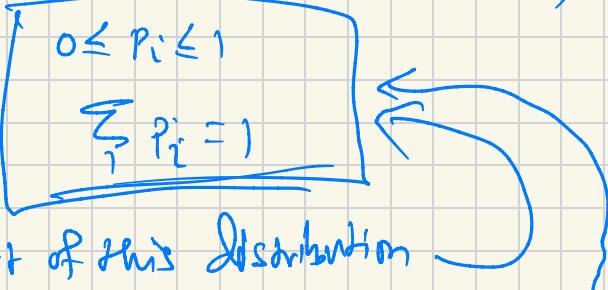
MLE can badly overfit in high dimension.

Dirichlet distribution

$$P(\boldsymbol{\alpha} | \boldsymbol{\alpha}) = \frac{\Gamma(\sum_{i=1}^K \alpha_i)}{\prod_{i=1}^K \Gamma(\alpha_i)} \alpha_1^{\alpha_1-1} \cdots \alpha_K^{\alpha_K-1}$$

Sample from probability distribution.

$$p_1, p_2, \dots, p_K \sim \text{Dir}(\alpha_1, \alpha_2, \dots, \alpha_K)$$



$$P_2 = \frac{N_c}{N}$$

Dim $\Rightarrow C$

$$\mu_i = \frac{1}{N_c} \sum_{y_i \leq c} x_i$$

Dim $\Rightarrow C \times D$

$$\Sigma_c = \sum_{i: y_i = c} (m_i - \mu_c) (m_i - \mu_c)^T$$

Dim = $C \times D \times D$
 $O(CD^2)$

so we need to estimate these many parameters.
& its very high & if dim is large then
no of parameters to estimate is large.

so high chance is it will overfit if data is less.

Can we reduce the number of parameters?
YES, using naive Bayes rule.

It assumes that x_i & x_j are conditionally independent given y , for all $i \neq j$.

$$P(x_1, x_2, \dots, x_n | y) = \prod_i P(x_i | y)$$

Reducing the number of parameters to estimate

$$P(y|x_1, x_2, \dots, x_n) = \frac{P(x_1, x_2, \dots, x_n | y) P(y)}{P(x_1, x_2, \dots, x_n)}$$

$$\begin{aligned} P(x_1, x_2, \dots, x_n | y) &= P(x_1 | x_2, x_3, \dots, x_n, y) * P(x_2 | x_3, \dots, x_n, y) * P(x_3 | x_4, \dots, x_n, y) \\ &\quad \vdots \\ &= P(x_1 | x_2, \dots, x_n, y) * P(x_2 | x_3, \dots, x_n, y) * \dots * P(x_n | y) \\ &\quad * P(y) \end{aligned}$$

Now the "naive" conditional independence assumption comes into play: assume that each feature x_i is conditionally independent of every other feature x_j for $i \neq j$, given y .

$$\begin{aligned} \text{so, } P(y|x_1, x_2, \dots, x_n) &\propto P(y) * P(x_1|y) * P(x_2|y) * \dots * P(x_n|y) \\ &\propto P(y) * \prod_{i=1}^n P(x_i|y) \end{aligned}$$

To estimate the parameter it requires 2D parameters.

$$\hat{y} = \operatorname{argmax}_{y_k} P(y=y_k) * \prod_i P(x_i | y=y_k)$$

$$\hat{y} = \operatorname{argmax}_{y_k} \sigma_k \prod_i \delta_{ijk}$$

MLB

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} P(D|\theta)$$

$$\hat{\pi}_k = \hat{P}(y=y_k) = \frac{\#D \{ y=y_k \}}{|D|}$$

$$\hat{\theta}_{ijk} = \hat{P}(x_i=x_j | y=y_k) = \frac{\#D \{ x_i=x_j \wedge y=y_k \}}{\#D \{ y=y_k \}}$$

MAP (Beta, Dirichlet prior) \rightarrow for binary classification
 \rightarrow when you have k -classes

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} P(\theta|D) = \underset{\theta}{\operatorname{argmax}} \frac{P(D|\theta) P(\theta)}{P(D)}$$

$$\hat{\pi}_k = \hat{P}(y=y_k) = \frac{\#D \{ y=y_k \} + (\beta_k - 1)}{|D| + \sum_m (\beta_m - 1)}$$

$$\hat{\theta}_{ijk} = \hat{P}(x_i=x_j | y=y_k) = \frac{\#D \{ x_i=x_j \wedge y=y_k \} + (\beta_{jk} - 1)}{\#D \{ y=y_k \} + \sum_m (\beta_m - 1)}$$

NBC Algorithm

$$N_c = 0, N_{jc} = 0$$

for $i = 1:N$ do

$c = \gamma_i$ // class i

$$N_c := N_c + 1$$

 for $j = 1:D$ do

 if $x_{ij} = 1$ then

$$N_{jc} := N_{jc} + 1$$

$$\hat{\pi}_c = \frac{N_c}{N}$$

$$\hat{\theta}_{jc} = \frac{N_{jc}}{N}$$

Complexity:

$O(ND)$ time.

Naïve Bayes Classifier

- Features are conditionally independent given the class label.

$$p(\mathbf{x}|y=c, \boldsymbol{\theta}) = \prod_{j=1}^D p(x_j|y=c, \theta_{jc})$$

V.W.F.
R

• class-conditional density $p(\mathbf{x}|y)$

- In the case of real-valued features, we can use the Gaussian distribution: $p(\mathbf{x}|y=c, \boldsymbol{\theta}) = \prod_{j=1}^D \mathcal{N}(x_j|\mu_{jc}, \sigma_{jc}^2)$, where μ_{jc} is the mean of feature j in objects of class c , and σ_{jc}^2 is its variance.
- In the case of binary features, $x_j \in \{0, 1\}$, we can use the Bernoulli distribution: $p(\mathbf{x}|y=c, \boldsymbol{\theta}) = \prod_{j=1}^D \text{Ber}(x_j|\mu_{jc})$, where μ_{jc} is the probability that feature j occurs in class c .
- In the case of categorical features, $x_j \in \{1, \dots, K\}$, we can model use the multinomial distribution: $p(\mathbf{x}|y=c, \boldsymbol{\theta}) = \prod_{j=1}^D \text{Cat}(x_j|\boldsymbol{\mu}_{jc})$, where $\boldsymbol{\mu}_{jc}$ is a histogram over the K possible values for x_j in class c .

We know $\hat{p} \equiv \frac{N_c}{N} \sim \text{Beta}(\alpha, \beta)$

$$P(\hat{p}|y) = \text{Beta}(N_c + \alpha, N - N_c + \beta)$$

mode of Beta distribution.

$$\text{mode} = \frac{N_c + \alpha - 1}{N + \alpha + \beta - 2}$$

when $\alpha = 1$
 $\beta = 1$

It's same as
MLE which
is uniform
distribution

$$\text{mode} = \frac{N_c}{N} = \text{MLE}$$

when $\alpha = 2$ $\beta = 2$

$$\text{mode} = \frac{N_c + 1}{N + 2}$$

Same as MAP.
So, it's like Laplace
smoothing.

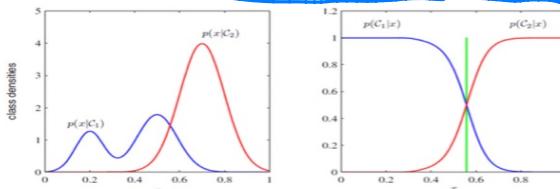
~~How~~ Generative model is useful for detecting outliers

Discriminative vs Generative

- **Generative Models**: model the joint distribution $p(x, C_k)$ directly and then normalize to obtain the posterior probabilities.

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)}$$

- Most demanding because it involves finding the joint distribution over both x and C_k . For many applications, x will have high dimensionality, and consequently we may need a large training set. If we only wish to make classification decisions, then it can be wasteful of computational resources but can be useful for detecting outliers or novel classes.



Let x be your data and y be some sort of label or decision variable that this data can be mapped to. Discriminative models learn (some approximation of) $p(y|x)$.

2 Generative models, depending on who you ask, learn $p(x)$, $p(x|y)$ or $p(y|x)$. These three are of course related. We can get the joint from the conditional as $p(x, y) = p(x|y)p(y)$, and we can get the marginal from the joint as $p(x) = \int p(x, y)dy$. In any case, the point is that they learn something about the probability of the data.

Now let's think about what an outlier is. An outlier is an observation in your data that is unlikely. That is, $p(x)$ is small (or $p(x|y)$ is small, if you are able to condition on an observed value of y). So you need some way of calculating $p(x)$ or $p(x|y)$. A generative model allows you to do that, through the math I described above. In words, a generative model was trained to accurately gauge the probability of your training data (assigning higher probabilities to more common values), and so given a new piece of data a generative model can tell you how probable that value is (under the model).

For a discriminative model, this is much harder, as we have just learned $p(y|x)$. How are you going to get from that to $p(x)$ or $p(x|y)$? To get the latter, you need:

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

(This is basically Bayes' rule in reverse.) Note that this depends on $p(x)$, which you don't know. Rearranging the same formula, we find:

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

Which again isn't helpful as it requires us to fill in terms on the r.h.s. that we don't know because they're not part of the model. In other words, because you never trained your discriminative model to tell you the probabilities of your data (only the probabilities of the labels *given* the data), you can't get it to do so at test time either.

For a generative model, you plug in either x or the combination of x and y , and the resulting probability, if it's small, will tell you that that particular observation is an outlier.

Parametric Supervised Learning

A parametric algorithm has a fixed number of parameters.

A parametric algorithm is computationally faster, but makes stronger assumptions about the data. the algo may work well if the assumptions turn out to be correct, but it may perform badly if the assumptions are wrong.

A learning model that summarises data with a set of parameters of fixed size (predefined mapped function) (independent of the numbers of training examples).

No matter how much data you throw at a parametric model, it won't change its mind about how many parameters it needs.

Ex of parametric algorithm is Linear Regression, Linear SVM, Perceptron, Logistic Regression.

Pros - need few data points to learn parameters.

Cons - strong distributional assumptions, not satisfied in practice.

◻ parametric models :

It assumes some finite set of parameters w . Given the parameters, future predictions, x , are independent of the observed data, D .

$$P(x|w, D) = P(x|w)$$

So, w capture everything there is to know about the data.

So, the complexity of the model is bounded even if the amount of data is unbounded. This makes them not very flexible.

◻ Non parametric models:

It assumes that the data distribution can't be defined in terms of such a finite set of parameters. But they can often be defined by assuming an infinite dimensional w . Usually we think of w as a function.

The amount of information that w can capture about the data D can grow as the amount of data grows. This makes them more flexible.

Ex: If non-parametric algorithm is KNN, DT, ANN, SVM with Gaussian kernels.

A model is called parameteric if it has finite and fixed number of parameters. Otherwise, it is called as non-parameteric.

For example, let us say that x_1, x_2, \dots, x_n are real valued samples from some hidden distribution. Now, I can model densities in two different ways:

- Parameteric Model:** I can choose a parameteric distribution to model the density, for example, Gaussian. I can compute mean as $\hat{\mu} = \frac{1}{n} \sum_{i=1}^n x_i$ and variance as $\hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \hat{\mu})^2$. Thus, using these parameters we can model the density as a Gaussian distribution i.e. $f(x) \propto \exp\left(-\frac{(x-\hat{\mu})^2}{2\hat{\sigma}^2}\right)$. Note that, no matter how many data points are given, we just have to estimate two parameters mean and variance for this model. Thus, the number of parameters for this model are finite and fixed and independent of the number of samples available to us.

- Non Parameteric Model:** We can also choose a non-parameteric model like Kernel Density Estimator.

Kernel Density Estimator defines density as: $f(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x-x_i}{h}\right)$ where h is the bandwidth parameter. Note that, the samples are directly part of the model which means they are the parameters of the model (different samples will give rise to different models). The number of parameters of this model grow linearly with the number of samples and theoretically, they could be infinite in number.

Non Parametric Density Estimation Methods in Machine Learning

Read

Discuss

Non-parametric methods: Similar inputs have similar outputs. These are also called instance-based or memory-based learning algorithms. There are 4 Non – parametric density estimation methods:

1. Histogram Estimator
2. Naive Estimator
3. Kernel Density Estimator (KDE)
4. KNN estimator (K – Nearest Neighbor Estimator)

Histogram Estimator

It is the oldest and the most popular method used to estimate the density, where the input space is divided into equal-sized intervals called **bins**. Given the training set $X = \{x^t\}_{t=1}^N$ an origin x_0 and the bin width h , the histogram density estimator function is:

$$\hat{p}(x) = \frac{\#\{x^t \text{ in the same bin as } x\}}{Nh}$$

Histogram estimator

The density of a sample is dependent on the number of training samples present in that bin. In constructing the histogram of densities we choose the origin and the bin width, the position of origin affects the estimation near the boundaries.

if Δ too small density model is too spiky

if Δ too large density model is too smooth.

* model complexity is decided by its bin size.

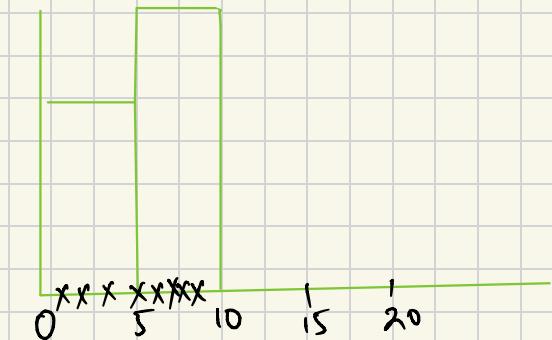
Ex of Histogram Estimator

Let's say values are

[2, 3, 5, 7, 9, 4, 6, 8]

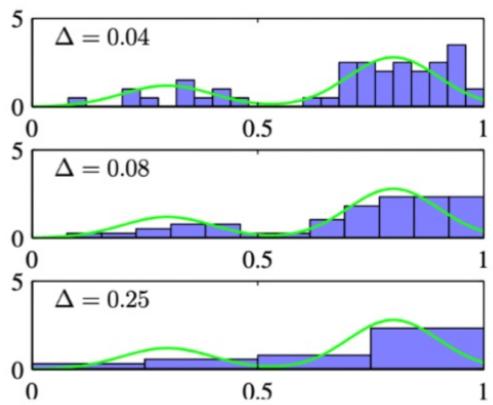
origin = 0

bin-width = 5



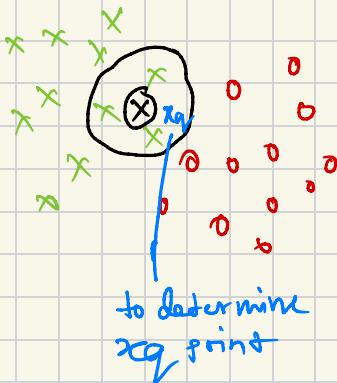
$$p(x_i) = \frac{\text{# of observations belong to same bin } i}{N \Delta i}$$

Δi = bin-width



KNN [Classification and Regression]

$$D =$$



$$D = \{(x_i, y_i) \mid x_i \in \mathbb{R}^n, y_i \in \{0, 1\}\}$$

steps:

1. find K-nearest points to x_q in D . let $K=3$ here.

$$\{x_1, x_2, x_3\} \text{ 3NN to } x_q$$

$$\downarrow \quad \downarrow \quad \downarrow$$

$$y_1, y_2, y_3$$

2. $\{y_1, y_2, y_3\} \rightarrow$ majority

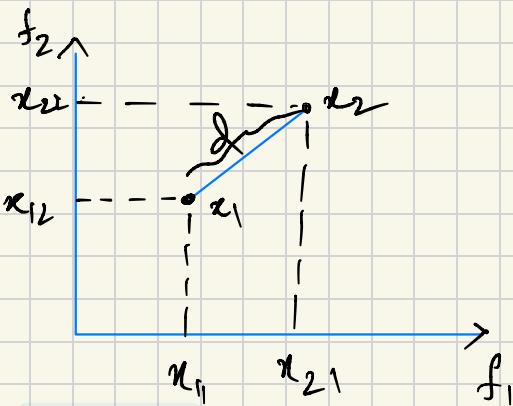
$$\{+, +, +\} \quad y_q = '+'$$

$$\{+, +, -\} \quad y_q = '+'$$

K should be odd no so that no draw happens.

Distance measures:

① Euclidean Distance



$$d = \sqrt{(x_{21} - x_{11})^2 + (x_{22} - x_{12})^2}$$

$$= \|x_1 - x_2\| \quad \begin{cases} \text{length of} \\ x_1, x_2 \end{cases}$$

$$x_1 \in \mathbb{R}^d$$

$$x_2 \in \mathbb{R}^d$$