$$\frac{\partial L}{\partial w} = \sum_{i=1}^{n} -\frac{1}{2\sigma^2} 2(w^T x_i - y_i) x_i - 2\frac{w}{2\beta^2}$$

$$\Rightarrow -\frac{1}{\sigma^2}(w^T x - y) x^T - \frac{w}{\beta^2} = 0$$

$$\Rightarrow -\frac{w^T x^T x}{\sigma^2} + \frac{y x^T}{\sigma^2} - \frac{w}{\beta^2} = 0$$

$$\Rightarrow w\left[-\frac{x^T x}{\sigma^2} - \frac{1}{\beta^2}\right] = -\frac{x^T y}{\sigma^2}$$

$$\Rightarrow w\left(\frac{\beta^2 x^T x + \sigma^2}{\sigma^2 \beta^2}\right) = \frac{x^T y}{\sigma^2}$$

$$\Rightarrow w\left(x^T x + \frac{\sigma^2}{\beta^2}\right) = x^T y$$

$$\Rightarrow w_{MAP} = (x^T x + \lambda I)^{-1} x^T y$$

$$\boxed{\lambda = \frac{\sigma^2}{\beta^2}}$$

Bias Variance Decomposition:

$y(x|D)$ ⇒ function learnt from training data

$h(x)$ ⇒ actual regression function (we don't know this but we can assume this to be an optimal function)

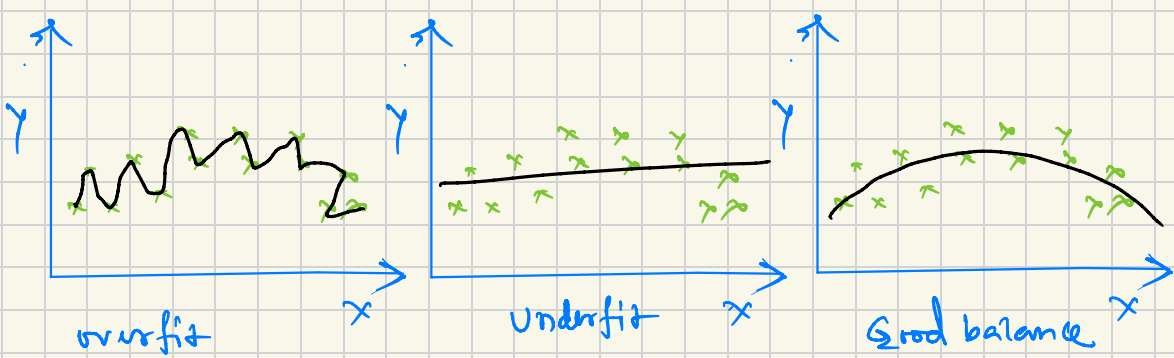
$$E_D\left[\{y(x;D) - h(x)\}^2\right]$$

$$\Rightarrow E_D\left[y(x;D) - E_D(y(x;D)) + E_D(y(x;D)) - h(x)\right]^2$$

$$\Rightarrow E_D\left[\{E_D(y(x;D)) - h(x)\}^2 + \{y(x;D) - E_D(y(x;D))\}^2 - 2\{E_D(y(x;D)) - h(x)\}\{y(x;D) - E_D(y(x;D))\}\right]$$

$$\Rightarrow E_D\left[\{E_D(y(x;D) - h(x)\}^2\right] + E_D\left[\{y(x;D) - E_D(y(x;D)\}^2\right] - 2\,E_D\{E_D(y(x;D)) - h(x)\}\,\underbrace{E_D\{y(x;D) - E_D(y(x;D))\}}_{\text{this becomes } 0.}$$

$$\Rightarrow \underbrace{E_D\left[\{E_D(y(x;D) - h(x)\}^2\right]}_{\text{Bias}^2} + \underbrace{E_D\left[\{y(x;D) - E_D(y(x;D)\}^2\right]}_{\text{variance.}}$$



Linear ML algo often have a high bias & a low variance

Non linear ML algo often have a low bias but a high variance.

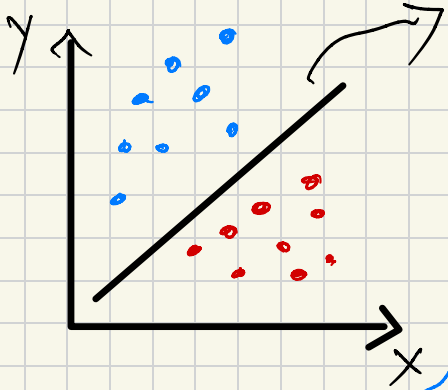
Bias variance trade off?

1. Dimensionality reduction & feature selection can decrease variance by simplifying model.
2. Similarly a larger training set tends to decrease variance.
3. Adding features tends to decrease bias, at the expense of introducing additional variance.
4. Linear & Generalized linear models can be regularized to decrease their variance at the cost of increasing their bias.

Logistic Regression:

Classification Problem

Linear Model



this is called discernment function or decision boundary. We need to learn this function.

Why Linear Regression can't be used for classification

→ For classification problem using linear regression, the probability is a Normal distribution. so, output $P(y=0|x) + P(y=1|x) \neq 1$.

→ even if you provide binary output to linear regression, it may provide output in real number like 1, 2, -2 etc which doesn't make sense in case of binary classification.

→ to make proper decision we need probabilistic output which sum upto 1 for all classes.

So, use of linear regression for classification problem is not a good option. We need good model which can distribute our probability value within 0 & 1 and for K-class classification within (1, K).

our task is to,

$$P(Y=1|X) \Rightarrow \mathbb{R}^D \xrightarrow{w^T x} \mathbb{R} \xrightarrow{\text{sigmoid fn}} \{0,1\}$$

linear transformation

also called probit function.

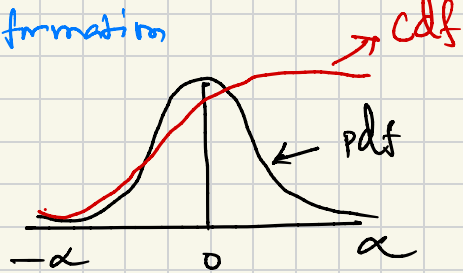
If, $P(x) = N(x|0,1)$

$$CDF(x) = \int_{-\infty}^{\alpha} P(x)\,dx$$

$$CDF(-\infty) = 0$$

$$CDF(0) = \int_{-\infty}^{0} P(x)\,dx = 0.5$$

$$CDF(\infty) = 1.$$



Logistic Sigmoid:

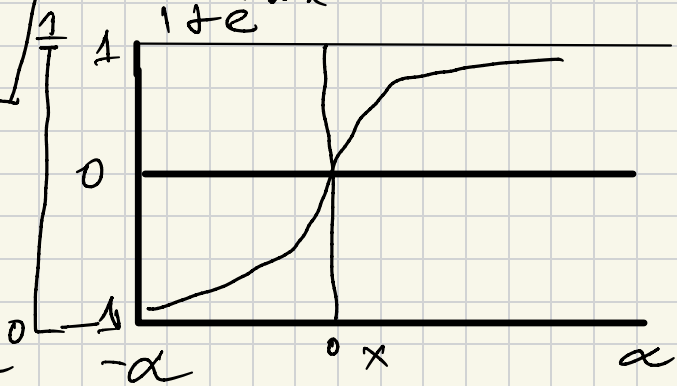
$$P(Y=1|X) = \sigma(w^T x) = \frac{1}{1+e^{-w^T x}}$$

$\sigma(-\infty) = 0$

$\sigma(0) = 0.5$

$\sigma(\infty) = 1$

so, sigmoid fun we can use to convert $w^T x$ to $\{0,1\}$

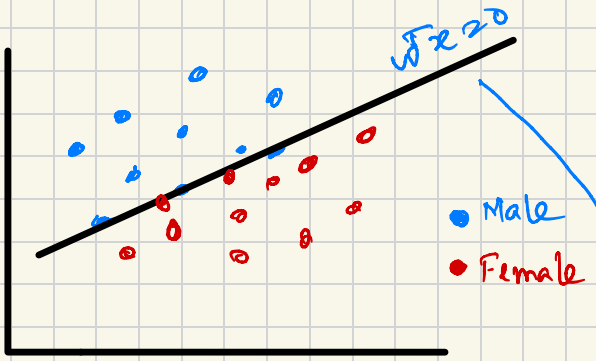


$$P(y=1|x) = \sigma(w^T x) = \frac{1}{1+e^{-w^T x}}$$

$$P(y=0|x) = 1 - \sigma(w^T x) = 1 - \frac{1}{1+e^{-w^T x}}$$

$$= \frac{e^{-w^T x}}{1+e^{-w^T x}}$$

$$= \frac{1}{1+e^{w^T x}}$$



→ this is decision boundary with equation $w^T x = 0$

$P(y=1|x) = P(y=0|x) = 0.5$

Points on lies on the line $w^T x = 0$ have same probability value = 0.5.

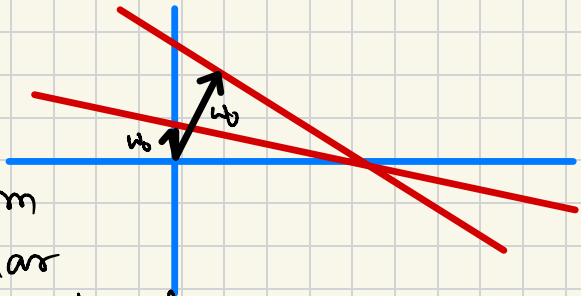
points above $w^T x = 0$ line have high probability towards class 1. or Male class & points lies below $w^T x = 0$ have high probability towards class 0 or Female class.

This probability distribution over the output is good for decision making. But sometime we don't need probability to find confidence towards a positive class. In such scenario we can go with approaches which just provide decision boundary. Ex {LDA, FLD, SVM.}

→ All they learn the decision boundary which is called Discriminant function.

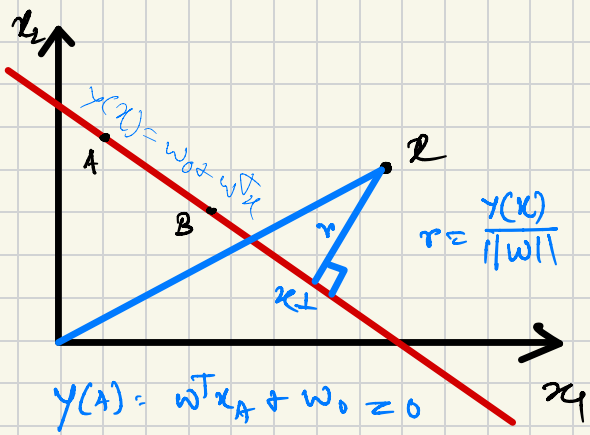
Discriminant Function:

$y(x) = w_0 + w^T x$



w_0 denotes how far your decision boundary from origin. It's a perpendicular direction towards decision boundary from origin.

w is orthogonal to every vector lying within the decision surface, so w determines the orientation of the decision surface.



$r = \frac{y(x)}{\|w\|}$

$y(A) = w^T x_A + w_0 = 0$

$y(B) = w^T x_B + w_0 = 0$

Both are on the line.

So,

$w^T (x_A - x_B) = 0$

Product of two vectors is 0 so they are perpendicular to each other. So, w is orthogonal to every vector lying within the decision surface.

Notes:

Simplistic discriminant function $y = w_0 + w^T x$ represent a hyperplane in $(m-1)$ dimensional space. where m is # of features.

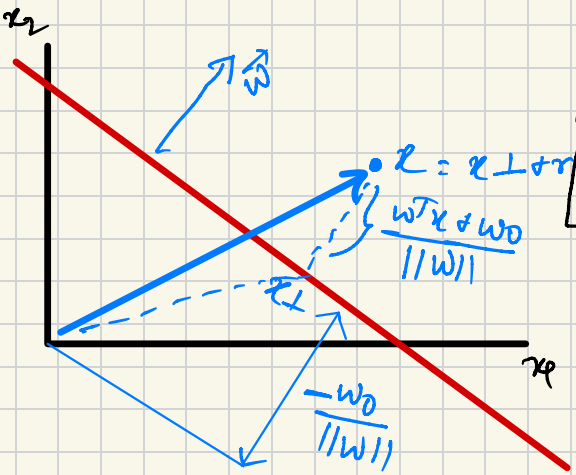
# features	Discriminant function
1	print
2	Line
3	plane
4	Hyperplane in 3D space
...	...
m	Hyperplane in $(m-1)$ D space

We have $w_0 + w^T x = 0$

$$w^T x = -w_0$$

Normalizing both side with the length of the vector $\|w\|$, we get normal distance from the origin to the decision surface.

$$d = \frac{w^T x}{\|w\|} = -\frac{w_0}{\|w\|}$$



So, w_0 shifts the boundary away from origin.

$\frac{w}{\|w\|}$ this is unit vector for direction.

$$y(x) = w^T x + w_0$$

replacing x with $x_\perp + r\frac{w}{\|w\|}$

$$y(x) = w^T x_\perp + r\frac{w^T w}{\|w\|} + w_0$$

both addition is 0 bcz they are on same line/plane.

$$y(x) = r\frac{w^T w}{\|w\|} = r\frac{\|w\|^2}{\|w\|}$$

$$y(x) = r\|w\|$$

So, $$r = \frac{y(x)}{\|w\|}$$

$y(x)$ determines the distance of a point to surface.

So, distance of x from origin =

$$\frac{w^T x + w_0}{\|w\|} + \frac{-w_0}{\|w\|}$$

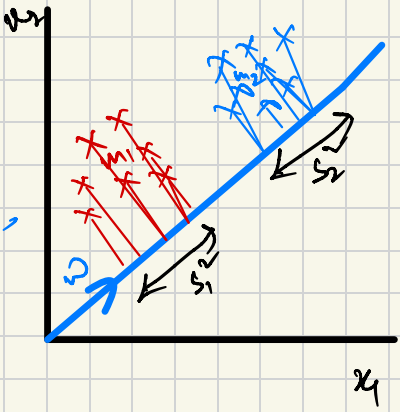
$$\Rightarrow \frac{w^T x}{\|w\|}$$

Supervised Linear Discriminant Analysis (LDA)

→ w as the direction to project x.

→ Find w such that when x is projected, classes are well separated.

→ Simplest measure of the separation of the classes, when projected onto w, is the separation of the projected class means.



$$m_1 = \frac{1}{N_1} \sum_{n \in C_1} x_n \qquad m_2 = \frac{1}{N_2} \sum_{n \in C_2} x_n$$

we need to maximize $m_2 - m_1 = w^T (m_2 - m_1)$

this expression can be made arbitrarily large simply by increasing the magnitude of w. To solve this problem we could constrain w to have unit length, so that $\sum_i w_i^2 = 1$

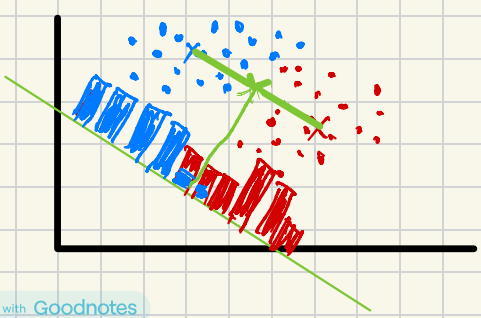
$$L = \underset{w}{\operatorname{argmax}}\; w^T (m_2 - m_1) - \lambda \|w_i\|^2$$

$$\nabla L = (m_2 - m_1) - 2\lambda w = 0$$

$$\Rightarrow \boxed{w = \frac{1}{2\lambda}(m_1 - m_2)}$$

So, $w \propto (m_1 - m_2)$

value of w is still depending on mean of two classes.



Therefore, we can see there are still considerable overlap when projected onto the line joining their means. To eliminate this, Fisher proposes a function which also consider variance of both class points.

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,

$$J(w) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

$$(m_2 - m_1)^2$$
$$= (w^T m_2 - w^T m_1)^2$$
$$= w^T (m_2 - m_1)(m_2 - m_1)^T w$$
$$= w^T S_B w$$

$$s_1^2 = \sum_{n \in C_k} (w^T x_n - m_1)^2$$
$$= \sum_{n \in C_k} w^T (x_n - m_1)(x_n - m_1)^T w$$
$$= w^T S_1 w$$

$$s_1^2 + s_2^2 = w^T S_w w \quad \text{where } S_w = S_1 + S_2$$

$$\text{so, } J(w) = \frac{w^T S_B w}{w^T S_w w}$$

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

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

$J(w)$ is a scalar, so we can drop this

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

$$\text{so, } \boxed{w \propto S_w^{-1} (m_1 - m_2)}$$

Multiclass classification:

given $\{(x_i, y_i)\}_{i=1}^{N}$ $\quad x_i \in \mathbb{R}^n$ $\quad y = \{0,1\}^K$

So, Probability that $y \in C_k$ given $x \& w$

$$P(y = C_k | x, w) = \sigma\left(\begin{bmatrix} w_1^T x \\ w_2^T x \\ \vdots \\ w_K^T x \end{bmatrix}\right)_k = \frac{e^{w_k^T x}}{\sum_i e^{w_i^T x}}$$

softmax function

$$\downarrow$$

$$\begin{bmatrix} - w_1 - \\ - w_2 - \\ \vdots \\ - w_K - \end{bmatrix} \begin{bmatrix} | \\ x \\ | \end{bmatrix}$$

Now we need to find $w_1, w_2 \cdots w_K$ given x, y

Likelihood

$$P(y | x, w) = \prod_{n=1}^{N} \prod_{k=1}^{K} \left(\frac{e^{w_k^T x_n}}{\sum_i e^{w_i^T x_n}}\right)^{y_{nk}}$$

$y_{nk} \leftarrow 1$ when $y_n \in C_k$

otherwise it will be 0

Now we need to minimize negative log likelihood

$$L(w) = -\sum_{n=1}^{N} \sum_{k=1}^{K} y_{nk} \left[\log e^{w_k^T x_n} - \log\left(\sum_{i=1}^{K} e^{w_i^T x_n}\right)\right]$$

$$L(w) = -\sum_{n=1}^{N} \sum_{k=1}^{K} y_{nk} w_k^T x_n + \sum_{n=1}^{N} \boxed{\sum_{k=1}^{K} y_{nk}} \log\left(\sum_{i=1}^{K} e^{w_i^T x_n}\right)$$

this will be 1. ✓

Now take derivative w.r.t w

$$\frac{\partial L(w)}{\partial w_j} = -\sum_{n=1}^{n} y_{nj} x_n + \sum_{n=1}^{n} \frac{1}{\sum_{i=1}^{k} e^{w_i^T x_n}} e^{w_j^T x_n} \cdot x_n$$

$$= \sum_{n=1}^{n} \left(\frac{e^{w_j^T x_n}}{\sum_{i=1}^{k} e^{w_i^T x_n}} - y_{nj} \right) x_n$$

$$\frac{e^{w_j^T x_n}}{\sum_{i=1}^{k} e^{w_i^T x_n}} \rightarrow P(y \in C_j \mid w, x)$$

$$= \sum_{n=1}^{n} \left(\hat{y}_{nj} - y_{nj} \right) x_n$$

diff between predicted value & true value.

Now we can apply gradient descent to get optimal value of w.

GD:

```
init {w}
for itr in iterations:
    for every class k:
        w^{k+1} = w^k - η ∇_{w_k} L(w)
```

After training

$$\text{Label}(x_{new}) = \text{argmax}_k \left[\sigma\left(w^{Final} x_{new} \right) \right]$$

$$= \text{argmax}_k \left(\begin{bmatrix} -w_1^T- \\ -w_2^T- \\ \vdots \\ -w_k^T- \end{bmatrix}_{k \times M} \begin{bmatrix} \quad \end{bmatrix}_{M \times 1} \right)$$

$$= \text{argmax}_k (k \times 1 \text{ matrix})$$

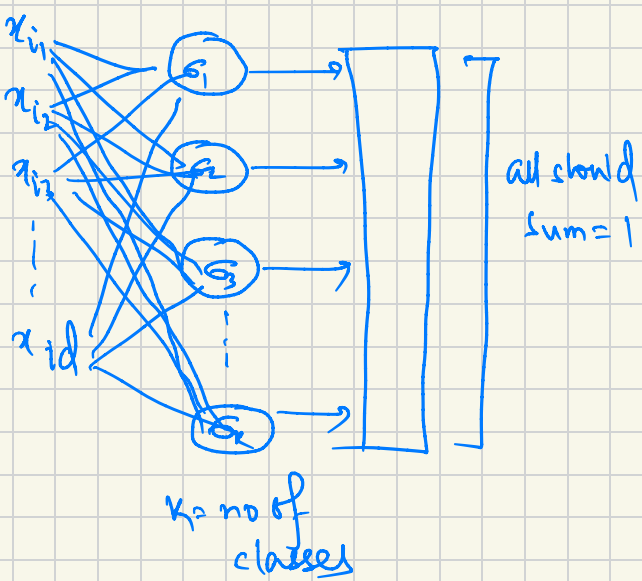
Multi Class Classification

$$P(y=c \mid x, w) = \frac{\exp(w_c^T x)}{\sum_{c'=1}^{C} \exp(w_{c'}^T x)}$$

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.



K = no of classes

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

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

Probabilistic Generative Models

$$P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)}$$

$$= \frac{1}{1+\exp(-a)} \quad \text{where } a = \ln \frac{P(x|C_1)P(C_1)}{P(x|C_2)P(C_2)}$$

$$= \sigma(a)$$

This is for binary classification.

for multiclass classification we have below form of the same equation.

$$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{\sum_{k=1} P(x|C_k)P(C_k)}$$

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

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

→ 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:

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

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

if $\sigma(a) = p(C_1|x)$

$$a = \ln\left(\frac{p(C_1|x)}{1-p(C_1|x)}\right)$$

$$a = \ln\left(\frac{p(C_1|x)}{p(C_2|x)}\right)$$

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**.

**

Let's say given logistic regression classifier

$$p = \sigma(x,y) = \frac{1}{1+\exp(-w_0 - 7.5x - 7.5y)}$$

then $-w_0 - 7.5x - 7.5y = \log\left(\frac{p}{1-p}\right)$

this is inverse of logit.

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

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

Σ = covariance matrix $(d \times d)$

$|\Sigma|$ = determinant

$$\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

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

$$p(C_1|x) = \sigma(w^T x + w_0)$$

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

$$w_0 = -\frac{1}{2}\mu_1^T \Sigma^{-1} \mu_1 + \frac{1}{2}\mu_2^T \Sigma^{-1} \mu_2 + \ln \frac{p(C_1)}{p(C_2)}$$

$$p(C_1|x) = \sigma\left(\ln \frac{p(x|C_1)p(C_1)}{p(x|C_2)p(C_2)}\right) =$$

$$\ln \frac{\frac{1}{\sqrt{(2\pi)^d}} \frac{1}{\sqrt{|\Sigma|}} \exp\left\{-\frac{1}{2}(x-\mu_1)^T \Sigma^{-1} (x-\mu_2)\right\}}{\frac{1}{\sqrt{(2\pi)^d}\sqrt{|\Sigma|}} \exp\left\{-\frac{1}{2}(x-\mu_2)^T \Sigma^{-1} (x-\mu_2)\right\}} + \ln \frac{p(C_1)}{p(C_2)}$$

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,and%20how%20they%20change%20together.

$$\ln \frac{\frac{1}{\sqrt{(2\pi)^d}\sqrt{|\Sigma|}} \cdot \frac{1}{\sqrt{(2\pi)^d}\sqrt{|\Sigma|}} \exp\left\{-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right\}}{\frac{1}{\sqrt{(2\pi)^d}\sqrt{|\Sigma|}} \exp\left\{-\frac{1}{2}(x-\mu_2)^T\Sigma^{-1}(x-\mu_2)\right\}} + \ln\frac{P(C_1)}{P(C_2)}$$

$A^T + A$
$= 2$

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

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

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

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

$$\Rightarrow \frac{1}{2}\Big(\underset{1\times d}{\Sigma^{-1}}\underset{d\times 1}{(\mu_1-\mu_2)^T}x + \underset{1\times d}{\Sigma^{-1}}\underset{d\times 1}{(\mu_1-\mu_2)}x^T - \mu_1^T\Sigma^{-1}\mu_1 + \mu_2^T\Sigma^{-1}\mu_2\Big)$$

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

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

For multiclass classification

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

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

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

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

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

If covariance is not shared

Quadratic terms of x will be present.

It is also called Quadratic discriminant analysis

$$P(y=c|x,w) = \frac{\pi_c |2\pi\Sigma_c|^{-1/2} \exp\left[-\frac{1}{2}(x-\mu_c)^T \Sigma_c^{-1} (x-\mu_c)\right]}{\sum_{c'} \pi_{c'} |2\pi\Sigma_{c'}|^{-1/2} \exp\left[-\frac{1}{2}(x-\mu_{c'})^T \Sigma_{c'}^{-1} (x-\mu_{c'})\right]}$$

parameter estimation technique

$$P(x_n, C_1) = P(C_1)\,P(x_n|C_1) = \pi\,\mathcal{N}(x_n|\mu_1, \Sigma)$$

$$P(x_n, C_2) = P(C_2)\,P(x_n|C_2) = (1-\pi)\,\mathcal{N}(x_n|\mu_2, \Sigma)$$

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

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

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

$$\Rightarrow \sum_{n=1}^{N} \gamma_n \ln p \left[\frac{1}{2}(x_n-\mu_1)^T\Sigma^{-1}(x_n-\mu_1)\right] + (1-\gamma_n)\ln(1-p)\left[\frac{1}{2}(x_n-\mu_2)^T\Sigma^{-1}(x_n-\mu_2)\right]$$

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

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

$$\gamma_n - \gamma_n p - p + \gamma_n p = 0$$

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

$$\frac{\partial L}{\partial \mu_1} = \sum_{n=1}^{N} \frac{\partial}{\partial \mu_1} \gamma_n \left[-\frac{1}{2} (x_n - \mu_1)^T \Sigma^{-1} (x_n - \mu_1) \right]$$

$$= \sum_{n=1}^{N} \gamma_n \left(-\frac{1}{2} \cdot 2 (x_n - \mu_1) \Sigma^{-1} \right)$$

$$= \sum_{n=1}^{N} \gamma_n \left[+ (x_n - \mu_1) \Sigma^{-1} \right]$$

$$\Rightarrow \sum_{n=1}^{N} \gamma_n (x_n - \mu_1) \Sigma^{-1} = 0$$

$$\sum_{n=1}^{N} \gamma_n (x_n - \mu_1) = 0$$

$$\boxed{\mu_1 = \frac{1}{N_1} \left(\sum_{n=1}^{N} x_n \gamma_n \right)}$$

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

$$= -\frac{1}{2} \left(-2 \Sigma^{-1} (y - \mu) \right)$$

$$= \Sigma^{-1} (y - \mu)$$

PUBLIC

Questions

Tags

Users

Unanswered

TEAMS

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

Free

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

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

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

As "extra credit", use equations (57) and (61) to find that the gradient with respect to $\boldsymbol{\Sigma}$ is

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

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 :)