Today

→ Classification

- Binary least squares classifier (separating plane)
- Extensions to multiple classes
  - → One vs rest / One vs one / K class discriminant
  - → Probabilistic classifiers (in particular logistic regression)

General idea: Assign to our feature vectors one of K discrete classes $C_1, \dots C_K$

→ the result of this is a division of the input space into decision regions whose boundaries are called decision boundaries.

We will start with simple models for which the decision boundaries are linear functions of the input vector

→ Datasets where classes can be separated exactly by linear decision surfaces are called linearly separable.

→ One of the most common representation for class labels is to take $t^{(i)} = \{0, 1\}$

For multi class problems, we will use a one-hot encoding scheme. E.g if we have K classes, then a feature vector from class C_2 will be represented as

$$\vec{t} = (0, 1, 0, \ldots, 0) \in \mathbb{R}^K$$

Discriminant function takes a feature vector as input and assign it to one of the K classes

We will start by considering linear discriminant of the

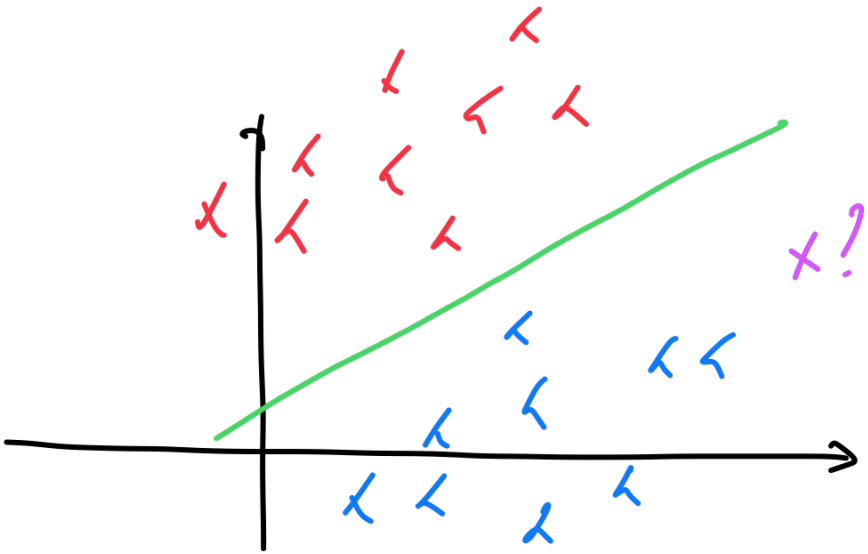
form $y(x) = \beta^T x + \beta_0$

Decision boundary given by $y(x) = 0$

Assuming $t^{(i)} = +1$ for $x^{(i)} \in C_1$ & $t^{(i)} = -1$ for $x^{(i)} \in C_2$

a new point will be assigned to C_1 if $y(x) > 0$

C_2 $y(x) < 0$



→ General objective:
learn the discriminant
function $y(x)$ that returns
a positive value for the red points
and a negative value for the blue prototypes

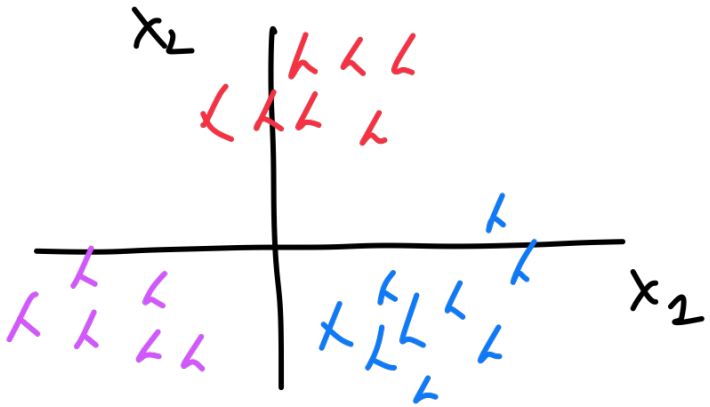
→ One of the simplest approach at learning our classifier is through least squares

$$\beta^* = \underset{\beta}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \left(t^{(i)} - \left(\beta_0 + \beta_1 x_1^{(i)} + \dots + \beta_D x_D^{(i)} \right) \right)^2$$

$t^{(i)}$: $+1$, -1

$\beta_0 + \beta_1 x_1^{(i)} + \dots + \beta_D x_D^{(i)}$: > 0, < 0

How about the Multiple class setting ?

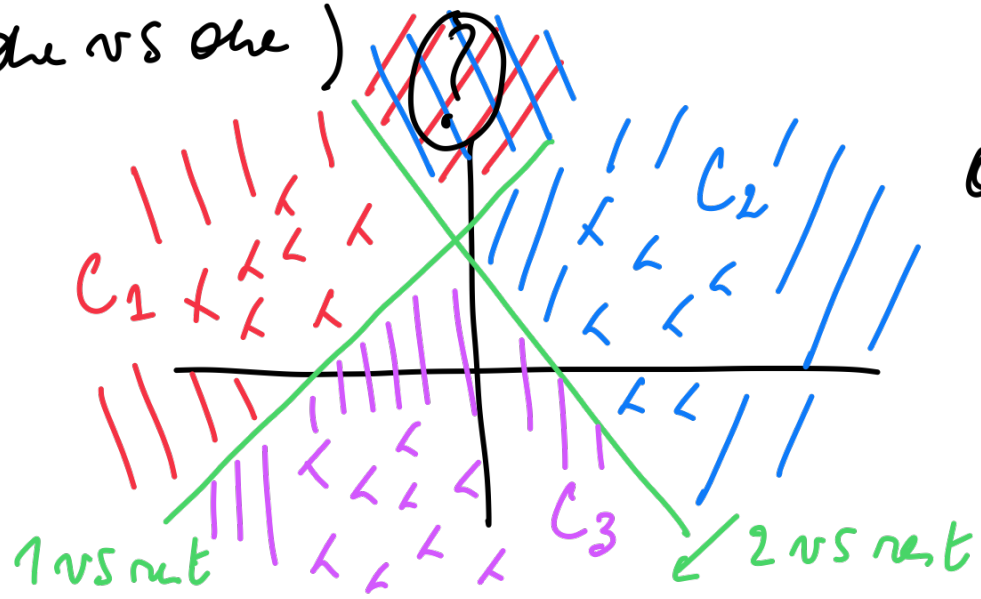


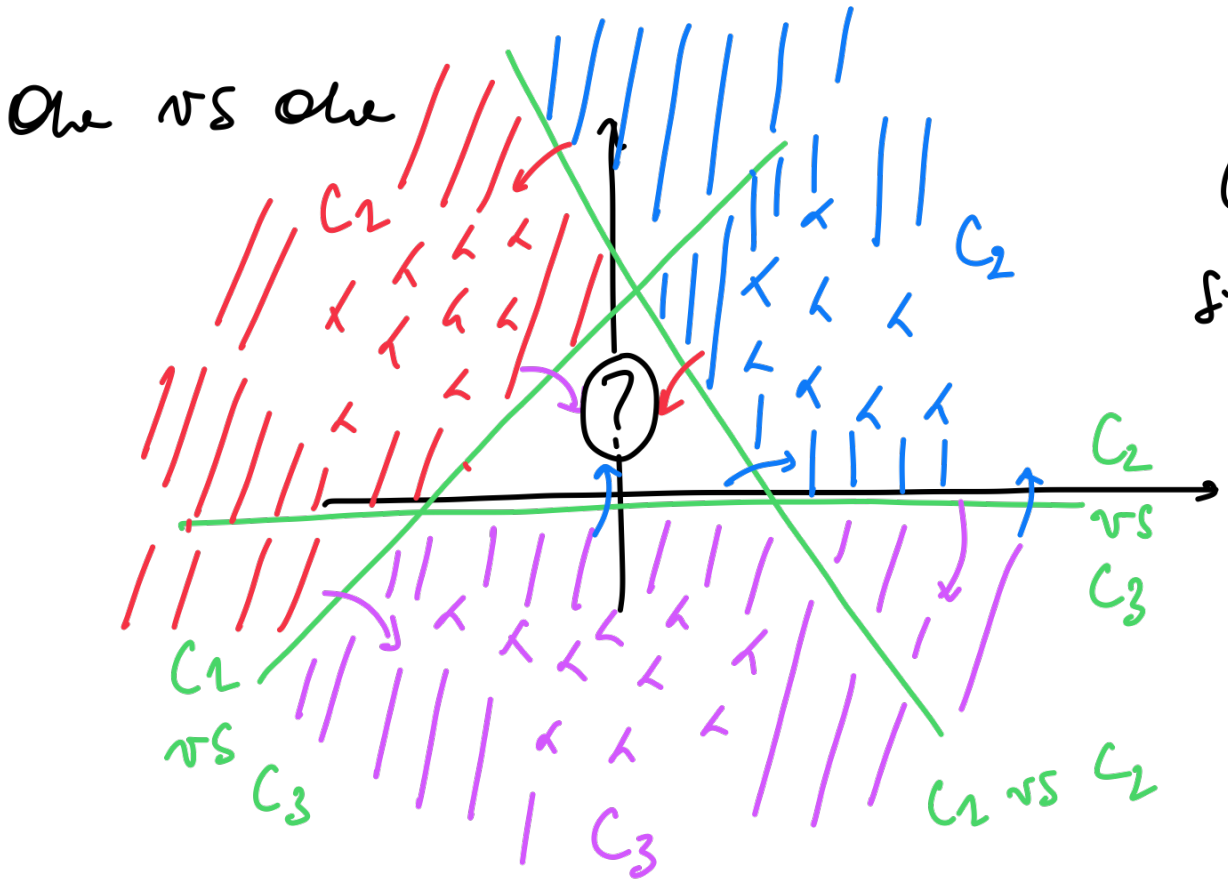
→ Solution 1 : Learn $K-1$ binary classifiers which separate each of the classes $C_1, \dots, C_{K-1}$ from the rest of the dataset.

(One vs all / One vs the rest)

→ Solution 2 learn a set of $\frac{K(K-1)}{2}$ classifiers separating class C_i from class C_j → class of a new point is determined based on the outcomes of the $\frac{K(K-1)}{2}$ classifiers following a majority vote

(One vs One)





Central region is simultaneously classified as C_1, C_2, and C_3.

→ In order to resolve those ambiguities, we can rely on a single K class discriminant function (learning K linear discriminant functions at once.)

→ encode the targets using a 1 of K coding scheme

$$t^{(i)} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

→ from which we can define the target matrix

$$T = \begin{bmatrix} | & | & & | \\ t^{(1)} & t^{(2)} & \cdots & t^{(N)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{K \times N}$$

We then consider a set of K separating planes β_k which we store as the rows of B

$$B = \begin{bmatrix} — & \vec{\beta}_1^T & — \\ & \vdots & \\ — & \vec{\beta}_K^T & — \end{bmatrix}$$

where each $\vec{\beta}_k$ is a vector

$$\vec{\beta}_k = (\beta_{k0}, \beta_{k1}, \cdots, \beta_{kD}) \in \mathbb{R}^{D+1}$$

We can then write the deviations between the targets and the predictions from the discriminant functions as

$$\underline{\underline{T}} - \underline{\underline{B}} \cdot \underline{\underline{\tilde{X}}}^T = \begin{bmatrix} | & | & & | \\ t^{(1)} & t^{(2)} & \dots & t^{(N)} \\ | & | & & | \end{bmatrix} - \begin{bmatrix} - & \vec{\beta}_1^T & - \\ & \vdots & \\ - & \vec{\beta}_K^T & - \end{bmatrix} \underbrace{\begin{bmatrix} | & & | \\ \tilde{x}^{(1)} & \dots & \tilde{x}^{(N)} \\ | & & | \end{bmatrix}}_{\underline{\underline{\tilde{X}}}^T}$$

and we can then minimize the residual sum of squares loss as before

$$\frac{1}{NK} \sum_{i=1}^{N} \sum_{k=1}^{K} \left(t_k^{(i)} - \vec{\beta}_k^T \tilde{x}^{(i)} \right)^2 \qquad (*)$$

Minimizing (*) is equivalent to looking for a set of planes where the plane associated to the k^{th} class outputs a value as close as possible to 1 for points that are in C_k and 0 otherwise.

(*) = Sum of squared entries of the matrix $T - B\tilde{X}$

$= Tr((T - B\tilde{X}^T)^T(T - B\tilde{X}^T))$

→ Computing the derivatives of (**) with respect to B and setting it to 0, we get a matrix extension of the normal equations which we can solve for B

$$\underline{\underline{B}}^T = \left(\underline{\underline{\tilde{X}}}^T \underline{\underline{\tilde{X}}}\right)^{-2} \underline{\underline{\tilde{X}}}^T T^T$$

So far: Supervised learning

→ linear regression (w/ OLS loss)

$\{x^{(i)}, t^{(i)}\}_{i=1}^{N}$

→ gradient descent (including batch, stochastic)

→ Normal equations

→ regularization (Best subset selection, l_p regularization including Ridge & LASSO)

→ Statistical intuition (including MLE / MAP)

→ Evolution of the MSE and Bias / Variance Decomposition as a function of Model Complexity

Linear Classification ($t^{(i)} \in \{1, 2, \dots, K\}$)

→ Least squares binary classifier

→ Extensions to multiple classes through one vs rest or one vs one and limitations

→ Single K class discriminant classifier (least squares)
(Relying on one hot encoding of the $t^{(i)}$)

$$\vec{t}^{(i)} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \text{ if } x^{(i)} \in C^1 \qquad T = \begin{bmatrix} | & | & & | \\ \vec{t}^{(1)} & t^{(2)} & \dots & t^{(N)} \\ | & | & & | \end{bmatrix}$$

$$\text{Classification Matrix} = \begin{bmatrix} — & \vec{\beta}^{(2)} & — \\ & \vdots & \\ — & \vec{\beta}^{(K)} & — \end{bmatrix} \qquad \underset{=}{\tilde{X}} = \begin{bmatrix} — & \tilde{x}^{(1)} & — \\ & \vdots & \\ — & \tilde{x}^{(N)} & — \end{bmatrix}$$

→ K classes discriminant: Solve for B in

$$\min_{B} \sum_{ij} \left(T - B\tilde{\underline{\underline{X}}}^T\right)^2_{ij}$$

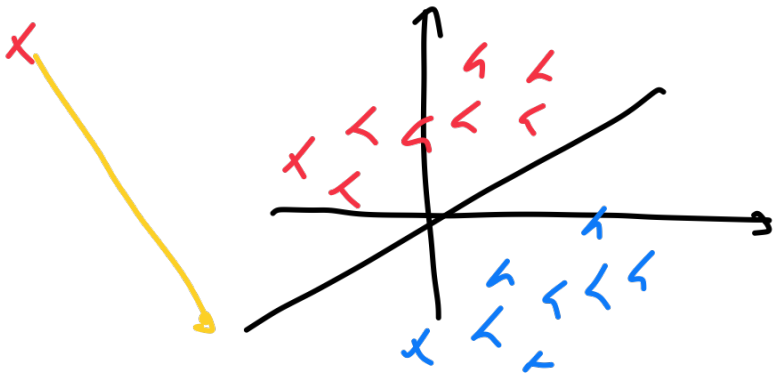
Today: – Probabilistic classifiers

Discriminative → logistic regression
→ definition
→ learning through MLE and binary cross entropy

∨
∧

Generative → Gaussian Discriminant Analysis
linear Discriminant Analysis

Logistic Regression



Recall the least squares loss

$$l(\beta) = \frac{1}{N} \sum_{i=1}^{N} (t^{(i)} - \beta^T \tilde{x}^{(i)})^2$$

→ 2 issues: Output is real (unbounded) while we would prefer a binary or confidence measure

→ risk of misclassification in the presence of outliers

→ One fix is to introduce an activation function.

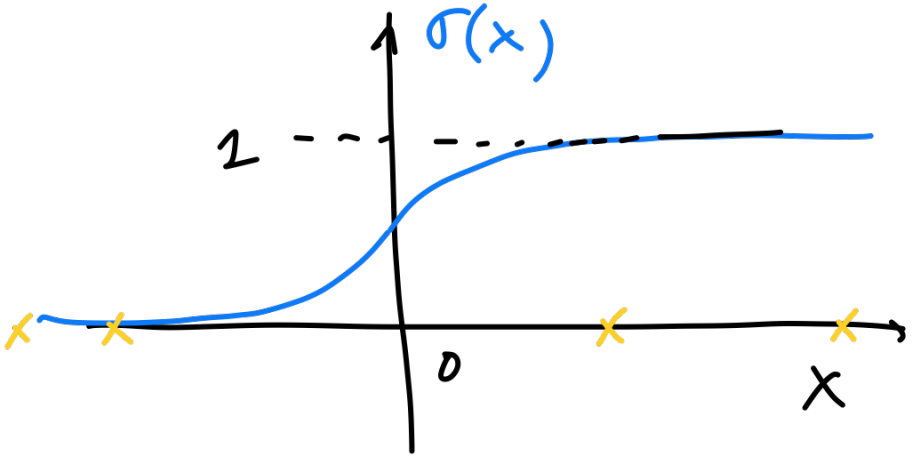
→ One possibility to improve interpretability is to chose a function that outputs a value in $[0,1]$ (can then be considered as measure of how "confident" we are of our classification. $y(x) = 1$ → means $x \in C^1$ with 100% certainty

$y(x) = 0$ " $x \in C^0$ "

$y(x) \in (0,1)$ → non zero risk of misclassification

An example of such function is the Sigmoid function $\sigma(x)$

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



→ Wherever we are with respect to the decision boundary, the decision remains bounded.

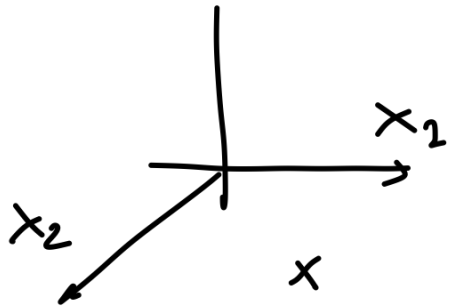
From this, we can define our logistic regression classifier as

$$h_\beta(x) = \sigma(\beta^T \tilde{x}) = \frac{1}{1 + e^{-\beta^T \tilde{x}}}$$

→ Since our classifier outputs a value in $[0, 1]$ we can interpret this output as a probability.

letting $p(t^{(i)} \in C_1 \mid x^{(i)}) = h_\beta(x^{(i)})$

$p(t^{(i)} \in C_0 \mid x^{(i)}) = 1 - h_\beta(x^{(i)})$



We can (1) consider the probability of observing our complete set of training pairs $\{x^{(i)}, t^{(i)}\}_{i=1}^N$ and then look for the weight vector β which maximizes this probability

provided that $t^{(i)}$'s are independent

$$p(\{t^{(i)}\}_{i=1}^N \mid \{x^{(i)}\}_{i=1}^N ; \beta) = \prod_{i=1}^N p(t(x^{(i)}) = t^{(i)} \mid x^{(i)}; \beta)$$

$$= \prod_{i=1}^N \begin{cases} p(t(x^{(i)}) = 1 \mid x^{(i)}; \beta) & \text{if } t^{(i)} = 1 \\ p(t(x^{(i)}) = 0 \mid x^{(i)}, \beta) & \text{if } t^{(i)} = 0 \end{cases} \quad (*)$$

$$(*) = \prod_{i=1}^{N} p\left(t(x^{(i)}) = 1 \mid x^{(i)}; \beta\right)^{t^{(i)}} p\left(t(x^{(i)}) = 0 \mid x^{(i)}; \beta\right)^{(1-t^{(i)})}$$

→ taking the log and maximizing, we get

$$\beta_{MLE} = \underset{\beta}{\operatorname{argmax}} \log \left(\prod_{i=1}^{N} h_\beta(x^{(i)})^{t^{(i)}} \left(1 - h_\beta(x^{(i)})\right)^{(1-t^{(i)})} \right)$$

$$= \underset{\beta}{\operatorname{argmax}} \sum_{i=1}^{N} t^{(i)} \log\left(h_\beta(x^{(i)})\right) + (1-t^{(i)}) \log\left(1 - h_\beta(x^{(i)})\right)$$

$$= \underset{\beta}{\operatorname{argmin}} \left(- \sum_{i=1}^{N} t^{(i)} \log\left(h_\beta(x^{(i)})\right) + (1-t^{(i)}) \log\left(1 - h_\beta(x^{(i)})\right) \right) \quad (*)$$

LOG LOSS = BINARY CROSS ENTROPY

→ the minimizer to (*) can then be found through gradient descent. Taking the gradient of the binary cross entropy loss with respect to β,

$$\frac{\partial}{\partial \beta}\left\{ -t^{(i)} \log\left(h_\beta(x^{(i)})\right) - (1-t^{(i)}) \log\left(1-h_\beta(x^{(i)})\right)\right\}$$

$$= -t^{(i)} \frac{\frac{\partial}{\partial \beta} h_\beta(x^{(i)})}{h_\beta(x^{(i)})} - (1-t^{(i)}) \cdot \frac{\frac{\partial}{\partial \beta} h_\beta(x^{(i)})}{1-h_\beta(x^{(i)})} \quad (**)$$

$$\frac{\partial}{\partial \beta} h_\beta(x^{(i)}) = \frac{\partial}{\partial \beta} \sigma(\beta^T \tilde{x}^{(i)}) = \frac{\partial}{\partial \beta}\left(\frac{1}{1+e^{-\beta^T \tilde{x}^{(i)}}}\right) \quad (*)$$

$$(*) = \frac{1}{(1+e^{-\beta^T \tilde{x}^{(i)}})^2} \cdot \left(+ \tilde{x}^{(i)} \cdot e^{-\beta^T \tilde{x}^{(i)}} \right)$$

$$= \frac{1}{(1+e^{-\beta^T \tilde{x}^{(i)}})} \left(1 - \frac{1}{1+e^{-\beta^T \tilde{x}^{(i)}}} \right) \cdot \tilde{x}^{(i)}$$

$$\sigma'(x) = \sigma(x)(1-\sigma(x)) \Rightarrow \frac{\partial}{\partial \beta}(\sigma(\beta^T \tilde{x}^{(i)})) = \sigma(\beta^T \tilde{x}^{(i)})(1-\sigma(\beta^T \tilde{x}^{(i)}) \ldots \tilde{x}^{(i)}$$

Substituting this in (**) we get

$$\frac{\partial l}{\partial \beta} = -\frac{t^{(i)}}{\sigma(\beta^T \tilde{x}^{(i)})} \cdot \sigma(\beta^T \tilde{x}^{(i)})(1-\sigma(\beta^T \tilde{x}^{(i)}) \tilde{x}^{(i)}$$

$$+ \frac{(1-t^{(i)})}{1-\sigma(\beta^T \tilde{x}^{(i)})} \sigma(\beta^T \tilde{x}^{(i)})(1-\sigma(\beta^T \tilde{x}^{(i)})) \tilde{x}^{(i)}$$

$$\frac{\partial \ell}{\partial \beta} = -t^{(i)} \left(1 - \sigma(\beta^T \tilde{x}^{(i)})\right) \tilde{x}^{(i)} + (1 - t^{(i)}) \sigma(\beta^T \tilde{x}^{(i)}) \tilde{x}^{(i)}$$

$$= \left(t^{(i)} - \sigma(\beta^T \tilde{x}^{(i)})\right) \tilde{x}^{(i)}$$

From which we can find β_{MLE} as

$$\beta^{(k+1)} = \beta^{(k)} - \eta \left(t^{(i)} - \sigma(\beta^T \tilde{x}^{(i)})\right) \tilde{x}^{(i)}$$

gradient step for binary cross entropy on the logistic regression classifier.

Probabilistic classifiers

→ Discriminative classifier

(given a feature vector $x^{(i)}$, a discriminative model outputs a decision)

→ learns a model for $p(t|x)$

→ Generative classifier

Idea: learn a representation for the objects from the two classes then classify a new example based on how it compares to our model of what typical instances should look like

→ learn a model for $p(x|t)$

Today

Probabilistic classifiers

→ Discriminative classifiers $p(t|x)$

→ Generative classifiers $p(x|t)$

→ Gaussian Discriminant Analysis (GDA)

→ Quadratic Discriminant Analysis (QDA)

→ Linear Discriminant Analysis (LDA)

→ Comparison between logistic regression & LDA

→ Perceptron, perceptron learning rule, convergence

→ Neural Networks

Generative Classifier

$X \in \mathbb{R}^D$

$\{x^{(i)}, t^{(i)}\}_{i=1}^{N}$

1) learn a model for $p(x|t)$

2) to classify a new prototype, use Bayes rule

$$p(t|x) = \frac{p(x|t)\,p(t)}{p(x)} = \frac{p(x|t)\,p(t)}{\sum_{\forall c} p(x|t=c)\,p(t=c)}$$

in order to determine the class of a new point $x^{(i)}$ we maximize $p(t=t_c|x^{(i)})$ over t_c

$$t(x^{(i)}) = \underset{t_c}{\operatorname{argmax}}\; p(t=t_c|x^{(i)}) = \underset{t_c}{\operatorname{argmax}}\; \frac{p(x^{(i)}|t=t_c)\,p(t=t_c)}{\sum_{\forall c'} p(x^{(i)}|t=t_{c'})\,p(t=t_{c'})}$$

the denominator is constant across the t_c's we can just focus on the numerator and define our class as

$$t(x^{(i)}) = \arg\max_{t_c} \; p(x^{(i)} | t = t_c)\, p(t = t_c)$$

→ To determine the class of a new prototype $x^{(i)}$ based on a generative classifier, we therefore need a model for $p(x|t)$ and for $p(t)$

→ One of the simplest model for $p(x|t)$ is the Multivariate Normal distribution

For $p(t)$ a reasonable approach is to use a Bernoulli

$$p(t=0) = \pi_0 \quad \text{and} \quad p(t=1) = \pi_1 = 1 - \pi_0$$

$$p(x|t=t_c) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\tfrac{1}{2}(x-\mu_c)^T \Sigma_c^{-1}(x-\mu_c)\right)$$

Together these models describe the Gaussian Discriminant Analysis generative classifier

In logistic regression we trained our model following a maximum likelihood approach on $p(t|x)$

When training generative classifiers we will again use a Max likelihood approach but this time on $p([x, t])$

For the GDA model, recall that we have

$$p(x \mid t = t_c) = \frac{1}{(2\pi)^{D/2} |\Sigma|^{1/2}} \exp\left(-(x-\mu_c)^T \Sigma^{-2} (x-\mu_c)\right)$$

$$p(t = t_c) = \pi_c$$

$$p(\{x, t\}) = p(x \mid t)\, p(t)$$

As before, if we assume that our examples are independent we can write $p(\{x^{(i)}, t^{(i)}\}_{i=1}^N)$ as

$$p(\{x^{(i)}, t^{(i)}\}_{i=1}^N) = \prod_{i=1}^N p(\{x^{(i)}, t(x^{(i)}) = t^{(i)}\}) \quad (*)$$

$$p(\{x^{(i)}, t(x^{(i)}) = c\}) = \begin{cases} p(x^{(i)} \mid t(x^{(i)}) = c)\, p(t(x^{(i)}) = c) & \text{if } c = t^{(i)} \\ 0 & \text{otherwise} \end{cases}$$

$$p(\{x^{(i)}, t^{(i)}\}_{i=1}^{N}) = \prod_{i=1}^{N} \left(p(x^{(i)} \mid C_0)\, p(C_0)\right)^{1-t^{(i)}} \times \left(p(x^{(i)} \mid C_1)\, p(C_1)\right)^{t^{(i)}}$$

QUADRATIC DISCRIMINANT ANALYSIS

$$p(\{x^{(i)}, t^{(i)}\}_{i=1}^{N}) = \prod_{i=1}^{N} \left(\frac{1}{(2\pi)^{D/2}|\Sigma_0|^{1/2}} \exp\left(-(x-\mu_0)^T \Sigma_0^{-2} (x-\mu_0)\right) \pi_0\right)^{1-t^{(i)}} \times \left(\frac{1}{(2\pi)^{D/2}|\Sigma_1|^{1/2}} \exp\left(-(x-\mu_1)^T \Sigma_1^{-2} (x-\mu_1)\right) (1-\pi_0)\right)^{t^{(i)}}$$

When taking $\Sigma_0 = \Sigma_1$ we recover the simpler LINEAR DISCRIMINANT ANALYSIS MODEL.

$$p(\{x^{(i)}, t^{(i)}\}_{i=1}^{N}) = \prod_{i=1}^{N} \left(\frac{1}{(2\pi)^{D/2} |\Sigma|^{1/2}} \exp\left(-(x^{(i)} - \mu_0)^T \Sigma^{-1} (x^{(i)} - \mu_0)\right) \pi_0 \right)^{1-t^{(i)}}$$

$$\times \left(\frac{1}{(2\pi)^{D/2} |\Sigma|^{1/2}} \exp\left(-(x^{(i)} - \mu_1)^T \Sigma^{-1} (x^{(i)} - \mu_1)\right) (1-\pi_0) \right)^{t^{(i)}}$$

Taking the log and maximizing we get

$$LL = \log\left(p(\{x^{(i)}, t^{(i)}\}_{i=1}^{N}) \right)$$

$$\Sigma^*, \mu_0^*, \mu_1^*, \pi_0^* = \underset{\Sigma, \mu_0, \mu_1, \pi_0}{\arg\max} \sum_{i=1}^{N} -\log\left((2\pi)^{D/2}|\Sigma|^{1/2}\right)$$

$$(*) + \sum_{i=2}^{N} -\frac{1}{2}(x^{(i)} - \mu_0)^T \overbrace{\Sigma^{-2}}(x^{(i)} - \mu_0)(1 - t^{(i)})$$

$$(**) + \sum_{i=2}^{N} -\frac{1}{2}(x^{(i)} - \mu_1)^T \overbrace{\Sigma^{-2}}(x^{(i)} - \mu_1)\, t^{(i)}$$

$$+ \sum_{i=1}^{N} (1 - t^{(i)}) \log(\pi_0) + \sum_{i=1}^{N} t^{(i)} \log(1 - \pi_0)$$

In the LDA setting, we get closed form expressions for each of the $\mu_0, \mu_1, \Sigma, \pi_0$ by computing the derivatives and setting them to 0.

Starting with π_0, taking the derivative of the log likelihood function, we get

$$\frac{\partial LL}{\partial \pi_0} = \sum_{i=1}^{N} (1 - t^{(i)}) \frac{1}{\pi_0} + \sum_{i=1}^{N} t^{(i)} \frac{-1}{1-\pi_0} = 0$$

$$N_0 \frac{1}{\pi_0} - N_1 \cdot \frac{1}{1-\pi_0} = 0$$

N_0 = number of examples in class C_0

N_1 = number of examples in class C_1

$$\Leftrightarrow \quad N_0(1-\pi_0) = N_1 \pi_0 \qquad \pi_0 = \frac{N_0}{N_0 + N_1}$$

For μ_0, μ_1 (the two means) we follow a similar reasoning on (*) and (**)

$$\frac{\partial}{\partial \mu_0} -\sum_{i=1}^{N} (x^{(i)} - \mu_0)^T \Sigma^{-1} (x^{(i)} - \mu_0) (1 - t^{(i)})$$

$$\frac{\partial}{\partial x} x^T A x = 2Ax$$

$$= \sum_{i=2}^{N} 2 (1 - t^{(i)}) \Sigma^{-1} (x^{(i)} - \mu_0) = 0$$

$$\frac{\partial}{\partial \mu_2} - \sum_{i=2}^{N} (x^{(i)} - \mu_2)^T \Sigma^{-1} (x^{(i)} - \mu_1) t^{(i)}$$

$$= \sum_{i=1}^{N} 2 t^{(i)} \Sigma^{-1} (x^{(i)} - \mu_2) = 0$$

} Setting the derivatives to zero

$$\sum_{i=1}^{N} (1 - t^{(i)})\, x^{(i)} = \sum_{i=1}^{N} (1 - t^{(i)})\, \mu_0$$

$$\sum_{i \in C_0} x^{(i)} \qquad N_0$$

$$\sum_{i=1}^{N} t^{(i)}\, x^{(i)} = \sum_{i=1}^{N} t^{(i)}\, \mu_1$$

$$\sum_{i \in C_1} x^{(i)} \qquad N_1$$

$$N_0 \mu_0 = \sum_{i \in C_0} x^{(i)} \quad \Rightarrow \quad \mu_0 = \frac{1}{N_0} \sum_{i \in C_0} x^{(i)}$$

$$N_1 \mu_1 = \sum_{i \in C_1} x^{(i)} \qquad \mu_1 = \frac{1}{N_1} \sum_{i \in C_1} x^{(i)}$$

Finally For the covariance Σ, using the fact that

$$\frac{\partial}{\partial A} \log(|A|) = (A^{-1})^T \quad (*)$$

Computing the derivative of the First 3 terms in the log likelihood w.r.t Σ^{-2}, we get

$$\frac{\partial}{\partial \Sigma^{-2}} \left\{ -\frac{N}{2} \underbrace{\log(|\Sigma|)}_{} + \sum_{i=2}^{N} (x^{(i)} - \mu_0)^T \Sigma^{-2} (x^{(i)} - \mu_0)(1 - t^{(i)}) \right.$$

$$\left. + \sum_{i=1}^{N} (x^{(i)} - \mu_2)^T \Sigma^{-2} (x^{(i)} - \mu_2)\, t^{(i)} \right\}$$

$$=$$

using

$$\log(|\Sigma|) = \log(|\Sigma|^{-2}) = -\log(|\Sigma^{-2}|) \quad (*)$$

$$(*) = \frac{\partial}{\partial \Sigma^{-1}} \Big(N \log(|\Sigma^{-1}|) - \sum_{i=1}^{N} (x^{(i)} - \mu_0)^T \Sigma^{-1} (x^{(i)} - \mu_0)(1 - t^{(i)}) - \sum_{i=1}^{N} (x^{(i)} - \mu_1)^T \Sigma^{-1} (x^{(i)} - \mu_1)\, t^{(i)} \Big)$$

using $(*)$

as well as $\frac{\partial}{\partial A}(x^T A x) = x x^T$ we get

$$N \Sigma^T - \underbrace{\sum_{i=1}^{N} (1 - t^{(i)})(x^{(i)} - \mu_0)(x^{(i)} - \mu_0)^T}_{\sum_{i \in C_0} (x^{(i)} - \mu_0)(x^{(i)} - \mu_0)^T} - \underbrace{\sum_{i=1}^{N} t^{(i)} (x^{(i)} - \mu_1)(x^{(i)} - \mu_1)^T}_{\sum_{i \in C_1} (x^{(i)} - \mu_1)(x^{(i)} - \mu_1)^T} = 0$$

Final Solution for the Covariance

$$\Sigma = \frac{1}{N} \left\{ \sum_{i \in C_0} (x^{(i)} - \mu_0)(x^{(i)} - \mu_0)^T + \sum_{i \in C_1} (x^{(i)} - \mu_1)(x^{(i)} - \mu_1)^T \right\}$$

Today

Generative classifiers

→ Linear Discriminant Analysis

→ Connection between LDA & logistic regression

→ Naive Bayes Classifier

- Perceptron algorithm (including training, convergence

→ Neural Networks (including training through back propagation)

I) Connection between LDA & logistic regression

in LDA

$$p(t=c \mid x, \theta) \propto \frac{\exp\left(-\frac{1}{2}(x-\mu_c)^T \Sigma^{-2} (x-\mu_c)\right) \pi_c}{\sum_{\forall c'} \exp\left(-\frac{1}{2}(x-\mu_{c'})^T \Sigma^{-2} (x-\mu_{c'})\right) \pi_{c'}}$$

$\theta = (\pi_0, \mu_0, \mu_1, \Sigma)$

(vector of parameters)

c = class, in binary setting $c \in \{0, 1\}$

$$p(t=c \mid x, \theta) \propto \frac{\exp\left(\mu_c^T \Sigma^{-2} x - \frac{1}{2} x^T \Sigma^{-2} x - \frac{1}{2} \mu_c^T \Sigma^{-2} \mu_c\right) \pi_c}{\sum_{\forall c'} \exp\left(\mu_{c'}^T \Sigma^{-2} x - \frac{1}{2} x^T \Sigma^{-2} x - \frac{1}{2} \mu_{c'}^T \Sigma^{-2} \mu_{c'}\right) \pi_{c'}}$$

$$p(t=c \mid x, \theta) \propto \frac{\exp\left(\mu_c^T \Sigma^{-2} x - \frac{1}{2}\mu_c^T \Sigma^{-2} \mu_c + \log \pi_c\right) \exp\left(-\frac{1}{2} x^T \Sigma^{-2} x\right)}{\sum_{\forall c'} \exp\left(\mu_{c'}^T \Sigma^{-2} x - \frac{1}{2}\mu_{c'}^T \Sigma^{-2} \mu_{c'} + \log \pi_{c'}\right) \exp\left(-\frac{1}{2} x^T \Sigma^{-2} x\right)}$$

(*)

Assuming 2 classes $c \in \{0, 1\} \rightarrow \theta = (\pi_0, \pi_1, \mu_0, \mu_1, \Sigma)$

and letting $\beta_0 = \Sigma^{-2} \mu_0 \quad \gamma_0 = -\frac{1}{2}\mu_0^T \Sigma^{-2} \mu_0 + \log \pi_0$

$\beta_1 = \Sigma^{-2} \mu_1 \quad \gamma_1 = -\frac{1}{2}\mu_1^T \Sigma^{-2} \mu_1 + \log \pi_1$

Substituting $\beta_0, \beta_1, \gamma_0, \gamma_1$ in the class posterior (*) we get

$$p(t=c \mid x, \theta) = \frac{\exp\left(\beta_0^T x + \gamma_0\right)}{\exp\left(\beta_0^T x + \gamma_0\right) + \exp\left(\beta_1^T x + \gamma_1\right)}$$

$$p(t=0 \mid x, \theta) = \frac{1}{1 + \exp\left((\beta_1 - \beta_0)^T x + \gamma_1 - \gamma_0\right)}$$

$$= \sigma\left((\beta_0 - \beta_1)^T x + (\gamma_0 - \gamma_2)\right)$$

→ LDA can read as a particular instance of a logistic classifier

(II) Naive Bayes Classifier

→ Generative classifier

$$p(x \mid y=c, \theta) = \prod_{j=2}^{D} p(x_j \mid y=c, \theta_{jc})$$

→ Naive Bayes assumes that the features are conditionally independent given the class label

Depending on the nature of the features we should cite

Gaussian Naive Bayes (Real valued features)

$$p(x \mid y=c, \theta) = \prod_{j=1}^{D} \mathcal{N}(x_j \mid \mu_{jc}, \sigma_{jc}^2)$$

(LDA with diagonal covariance)

Bernoulli Naive Bayes (binary features)

$$p(x \mid y=c, \theta) = \prod_{j=1}^{D} \text{Ber}(x_j \mid \mu_{jc})$$

Categorical Naive Bayes (Categorial features $x_j \in \{1, \ldots, K\}$)

$$p(x \mid y=c, \theta) = \prod_{j=1}^{D} \text{Cat}(x_j \mid p_{jc})$$

p_{jck} probabilities

with $\sum_k p_{jck} = 1$

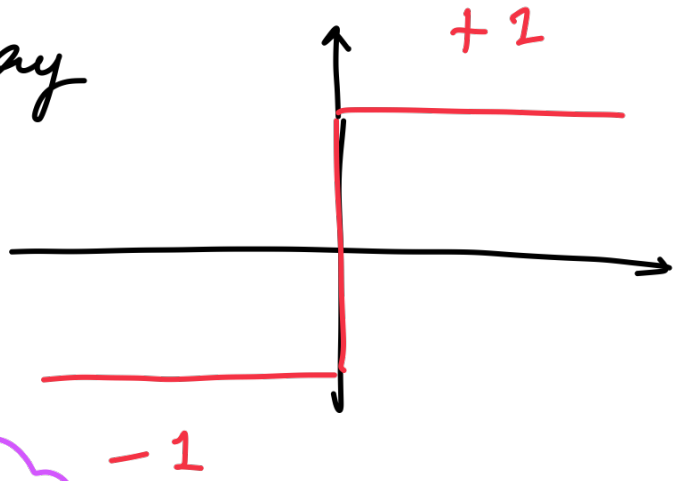
$p_{jck} \geq 0$

(III) Perceptron

→ combination of linear model with Heaviside activation

$$y(x) = \sigma(\beta^T \tilde{x}) \qquad \text{with} \quad \sigma(x) = \begin{cases} 1 & x > 0 \\ -1 & x \leq 0 \end{cases}$$

→ cannot be trained in the usual way as the gradient vanishes almost everywhere



$$\ell(\beta) = -\sum_{i \in \text{Misclassified}} t^{(i)} \left(\beta_0 + \beta_1 x_1^{(i)} + \dots + \beta_D x_D^{(i)} \right) \quad (*)$$

$$\{ x^{(i)}, t^{(i)} \}_{i=1}^{N}$$

$$t^{(i)} \in \{ \pm 1 \}$$

We then minimize $L(\beta)$ with respect to β and then recompute the new set of misclassified examples.

For a point that is strongly misclassified, we have

$$\text{sign}(t^{(i)}) \neq \text{sign}(\beta_0 + \beta_1 x_1^{(i)} + \dots + \beta_D x_D^{(i)})$$

hence a large positive contribution to (*)

Minimizing (*) will thus "correct" the plane β based on these misclassied points

→ the loss (*) can be minimized using an SGD approach over the misclassified points.

→ In this setting the SGD iterations are given by

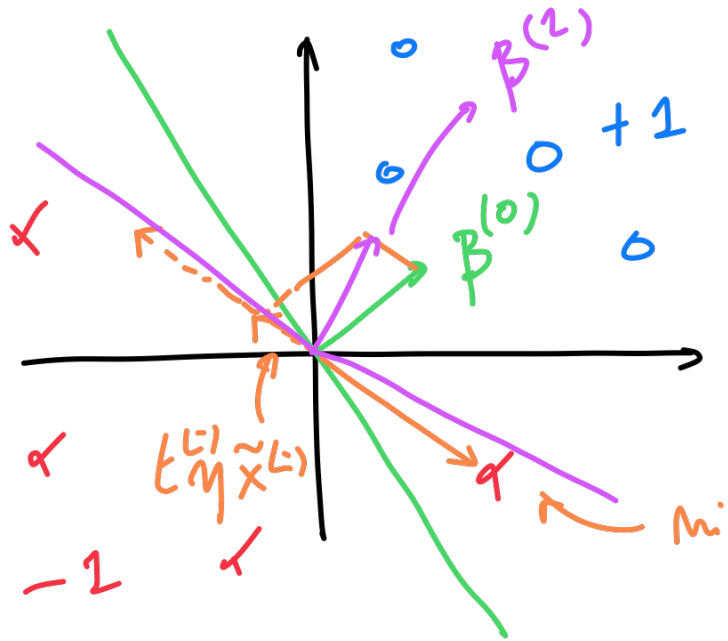
$$\beta \leftarrow \beta - \eta \operatorname*{grad}_{\beta} \ell(\beta)$$

$$\beta \leftarrow \beta + \eta \, \tilde{x}^{(i)} t^{(i)} \quad \text{(perceptron learning rule)}$$

→ Resulting Perceptron algorithm applies the rule iteratively on the misclassified points

for every misclassified $\tilde{x}^{(i)}$ do

$$\beta \leftarrow \beta + \eta \, \tilde{x}^{(i)} t^{(i)}$$



$$\beta^{(2)} = \beta^{(0)} + \eta\, t^{(i)} \tilde{x}^{(i)}$$

misclassified point $\tilde{x}^{(i)}$

$t^{(i)} = -1$

Step 1