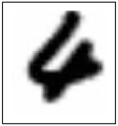
Linear Models for Classification

CMPT 419/726
Mo Chen
SFU Computing Science
Jan. 22, 2020

Bishop PRML Ch. 4

Classification: Hand-written Digit Recognition

$\boldsymbol{x}_i =$  $t_i = (0, 0, 0, 1, 0, 0, 0, 0, 0, 0)$

- Each input vector classified into one of K discrete classes
  - Denote classes by $\mathcal{C}_k$
- Represent input image as a vector $\boldsymbol{x}_i \in \mathbb{R}^{784}$.
- We have target vector $\boldsymbol{t}_i \in \{0, 1\}^{10}$
- Given a training set $\{(\boldsymbol{x}_1, \boldsymbol{t}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{t}_N)\}$, learning problem is to construct a “good” function $\boldsymbol{y}(\boldsymbol{x})$ from these.
  - $y: \mathbb{R}^{784} \rightarrow \mathbb{R}^{10}$

Generalized Linear Models

- Similar to previous chapter on linear models for regression, we will use a "linear" model for classification:

$$y(\boldsymbol{x}) = f(\boldsymbol{w}^\top \boldsymbol{x} + w_0)$$

- This is called a generalized linear model
- $f(\cdot)$ is a fixed non-linear function
    - e.g.

$$f(u) = \begin{cases} 1, & \text{if } u \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

- Decision boundary between classes will be linear function of $\boldsymbol{x}$
- Can also apply non-linearity to $\boldsymbol{x}$, as in $\phi_i(\boldsymbol{x})$ for regression

Outline

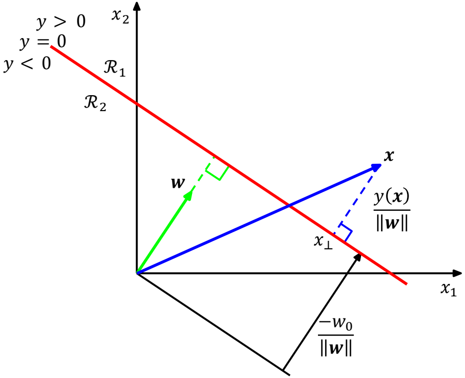
Outline

Discriminant Functions

Generative Models

Discriminative Models

Discriminant Functions with Two Classes

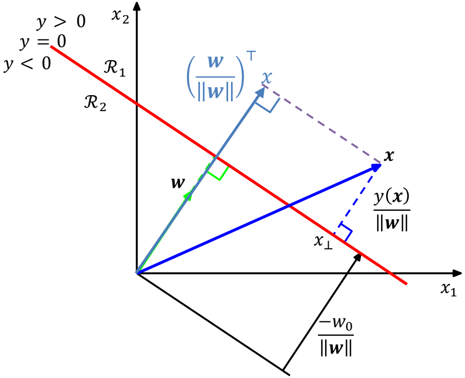


- Start with 2 class problem, $t_i \in \{0,1\}$
- Simple linear discriminant

$$y(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + w_0$$

apply threshold function to get classification

Discriminant Functions with Two Classes



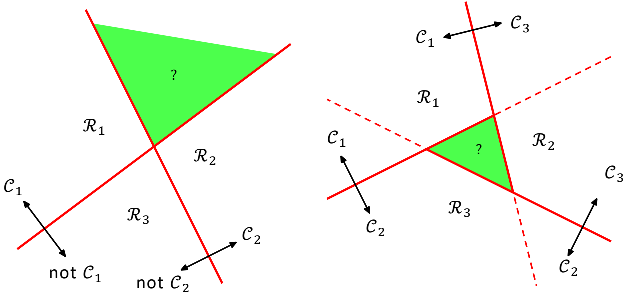
- $y(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + w_0$
  - Gradient of y is $\boldsymbol{w}$
  - Constant y values $\Rightarrow$ parallel lines
- If $y = 0$ (decision boundary),

$$\boldsymbol{w}^\top \boldsymbol{x} = -w_0 \Rightarrow \left(\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\right)^\top x = -\frac{w_0}{\|\boldsymbol{w}\|}$$

- In general, $\frac{y}{\|\boldsymbol{w}\|} = \frac{w^\top x}{\|w\|} + \frac{w_0}{\|w\|}$, or

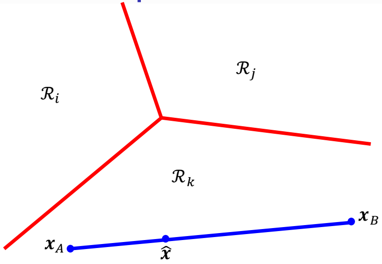
$$\left(\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\right)^\top x = \frac{y(\boldsymbol{x})}{\|\boldsymbol{w}\|} - \frac{w_0}{\|\boldsymbol{w}\|}$$

Multiple Classes



- A linear discriminant between two classes separates with a hyperplane
- How to use this for multiple classes?
- One-versus-the-rest method: build $K - 1$ classifiers, between $\mathcal{C}_k$ and all others
- One-versus-one method: build $K(K-1)/2$ classifiers, between all pairs

Multiple Classes



- A solution is to build K linear functions:

$$y_k(\boldsymbol{x}) = \boldsymbol{w}_k^\top \boldsymbol{x} + w_{k0}$$

  assign $\boldsymbol{x}$ to class $\arg\max_k y_k(\boldsymbol{x})$

- Gives connected, convex decision regions

$$\widehat{\boldsymbol{x}} = \lambda \boldsymbol{x}_A + (1-\lambda)\boldsymbol{x}_B$$

$$y_k(\widehat{x}) = \lambda y_k(\boldsymbol{x}_A) + (1-\lambda) y_k(\boldsymbol{x}_B)$$

$$\Rightarrow y_k(\widehat{x}) > y_j(\widehat{x}), \qquad \forall j \neq k$$

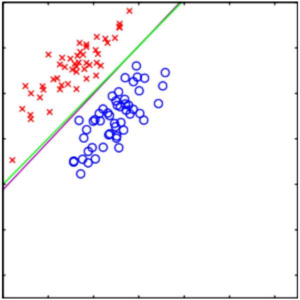
Least Squares for Classification

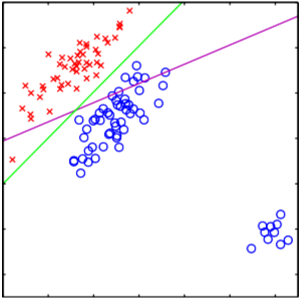
- How do we learn the decision boundaries $(\boldsymbol{w}_k, w_{k0})$?
- One approach is to use least squares, similar to regression
- Find $\boldsymbol{W}$ to minimize squared error over all examples and all components of the label vector:

$$E(\boldsymbol{W}) = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} (y_k(\boldsymbol{x}_n) - t_{nk})^2$$

- Some algebra, we get a solution using the pseudo-inverse as in regression

Problems with Least Squares

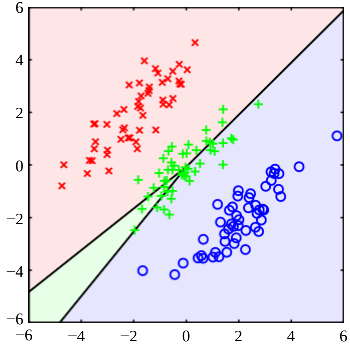




- Looks okay... least squares decision boundary
  - Similar to logistic regression decision boundary (more later)

- Gets worse by adding easy points?!
- Why?
  - If target value is 1, points far from boundary will have high value, say 10; this is a large error so the boundary is moved

More Least Squares Problems

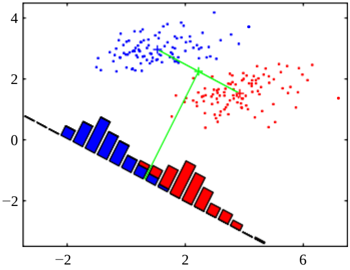


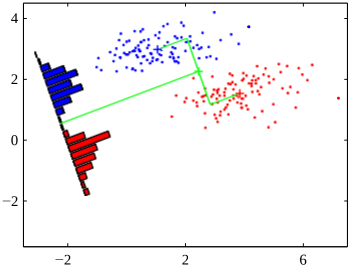
- Easily separated by hyperplanes, but not found using least squares!
- We'll address these problems later with better models
- First, a look at a different criterion for linear discriminant

Fisher's Linear Discriminant

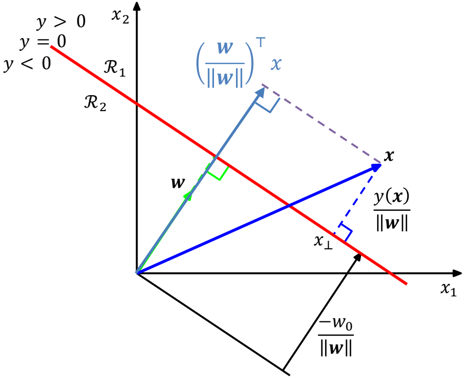
- The two-class linear discriminant acts as a projection:
$$y = \boldsymbol{w}^T \boldsymbol{x} \geq -w_0$$
followed by a threshold
- In which direction $\boldsymbol{w}$ should we project?
- One which separates classes "well"

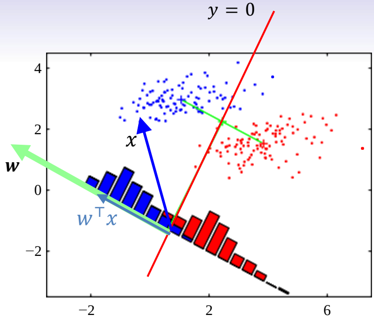
Fisher's Linear Discriminant

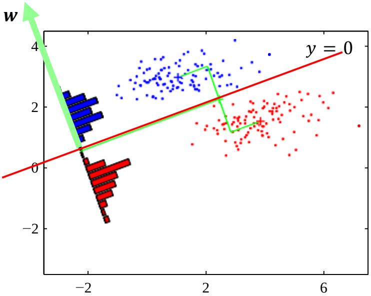




- A natural idea would be to project in the direction of the line connecting class means
- However, problematic if classes have variance in this direction
- Fisher criterion: maximize ratio of inter-class separation (between) to intra-class variance (inside)


$y > 0$
$y = 0$
$y < 0$
x_2
x_1
$\mathcal{R}_1$
$\mathcal{R}_2$
$\left(\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\right)^\top x$
$\boldsymbol{w}$
$\boldsymbol{x}$
$\frac{y(\boldsymbol{x})}{\|\boldsymbol{w}\|}$
$x_\perp$
$\frac{-w_0}{\|\boldsymbol{w}\|}$


$y = 0$
x
$\boldsymbol{w}$
$w^\top x$
4
2
0
−2
−2
2
6


$\boldsymbol{w}$
$y = 0$
4
2
0
−2
−2
2
6

Math time - FLD

- Projection $\boldsymbol{y}_n = w^\top \boldsymbol{x}_n$
- Inter-class separation is distance between class means (good):

$$m_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \boldsymbol{w}^\top \boldsymbol{x}_n$$

- Intra-class variance (bad):

$$s_k^2 = \sum_{n \in \mathcal{C}_k} (y_n - m_k)^2$$

- Fisher criterion:

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

maximize wrt $\boldsymbol{w}$

Math time - FLD

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

Between-class covariance:

$$S_B = (\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^\top$$

Within-class covariance:

$$S_W = \sum_{n \in \mathcal{C}_1} (\boldsymbol{x}_n - \boldsymbol{m}_1)(\boldsymbol{x}_n - \boldsymbol{m}_1)^\top + \sum_{n \in \mathcal{C}_2} (\boldsymbol{x}_n - \boldsymbol{m}_2)(\boldsymbol{x}_n - \boldsymbol{m}_2)^\top$$

Lots of math:

$$\boldsymbol{w} \propto S_W^{-1}(\boldsymbol{m}_2 - \boldsymbol{m}_1)$$

If covariance S_W is isotropic, reduces to class mean difference vector

FLD Summary

- FLD is a dimensionality reduction technique (more later in the course)
- Criterion for choosing projection based on class labels
  - Still suffers from outliers (e.g. earlier least squares example)

Perceptrons

- Perceptrons is used to refer to many neural network structures (more coming up)
- The classic type is a fixed non-linear transformation of input, one layer of adaptive weights, and a threshold:

$$y(\boldsymbol{x}) = f\left(\boldsymbol{w}^\top \phi(\boldsymbol{x})\right)$$

  - Developed by Rosenblatt in the 50s
- The main difference compared to the methods we've seen so far is the learning algorithm

Perceptron Learning

- Two class problem
- For ease of notation, we will use $t = 1$ for class $\mathcal{C}_1$ and $t = -1$ for class $\mathcal{C}_{2;}$ we choose f such that $f(a) = 1$ if $a \geq 0$ and $f(a) = -1$ otherwise
- We saw that squared error was problematic
- Instead, we'd like to minimize the number of misclassified examples
  - An example is mis-classified if $\boldsymbol{w}^\top \phi(\boldsymbol{x}_n) t_n < 0$
  - Perceptron criterion:

$$E_P(\boldsymbol{w}) = -\sum_{n \in \mathcal{M}} \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_n) t_n$$

sum over mis-classified examples only

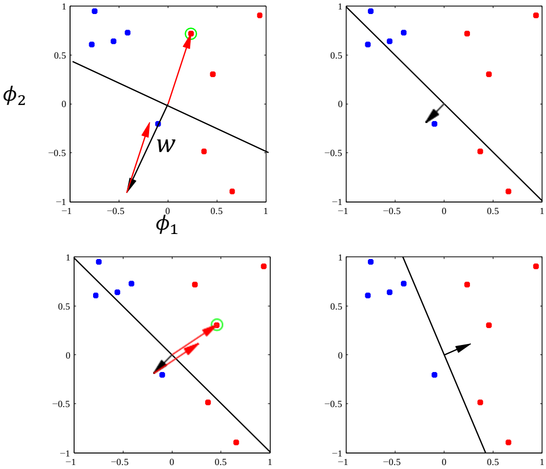
Perceptron Learning Algorithm

- Minimize the error function using stochastic gradient descent (gradient descent per example):

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} - \eta \nabla E_P(\boldsymbol{w}) = \boldsymbol{w}^{(\tau)} + \underbrace{\eta \phi(\boldsymbol{x}_n) t_n}_{\text{if incorrect}}$$

- Iterate over all training examples, only change $\boldsymbol{w}$ if the example is mis-classified
- Guaranteed to converge if data are linearly separable
- Will not converge if not
- May take many iterations
- Sensitive to initialization

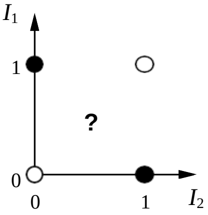
Perceptron Learning Illustration



- Note there are many hyperplanes with 0 error
  - Support vector machines have a nice way of choosing one

Limitations of Perceptrons

- Perceptrons can only solve linearly separable problems in feature space
  - Same as the other models in this chapter
- Canonical example of non-separable problem is X-OR
  - Real datasets can look like this too



Outline

Probabilistic Generative Models

- Up to now we've looked at learning classification by choosing parameters to minimize an error function
- We'll now develop a probabilistic approach
- With 2 classes, $\mathcal{C}_1$ and $\mathcal{C}_2$:

$$p(\mathcal{C}_1|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\boldsymbol{x})}$$ Bayes' Rule

$$p(\mathcal{C}_1|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\boldsymbol{x},\mathcal{C}_{\mathbf{1}}) + p(\boldsymbol{x},\mathcal{C}_2)}$$ Sum rule

$$p(\mathcal{C}_1|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\boldsymbol{x}|\mathcal{C}_{\mathbf{1}})p(\mathcal{C}_1) + p(\boldsymbol{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$ Product rule

- In generative models we specify the distribution $p(\boldsymbol{x}|\mathcal{C}_k)$ which generates the data for each class

Probabilistic Generative Models - Example

- Let's say we observe x which is the current temperature
- Determine if we are in Vancouver ($\mathcal{C}_1$) or Honolulu ($\mathcal{C}_2$)
- Generative model:

$$p(\mathcal{C}_1|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\boldsymbol{x}|\mathcal{C}_{\mathbf{1}})p(\mathcal{C}_1) + p(\boldsymbol{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

  - $p(\boldsymbol{x}|\mathcal{C}_1)$ is distribution over typical temperatures in Vancouver
    - e.g. $p(\boldsymbol{x}|\mathcal{C}_1) = \mathcal{N}(x; 10, 5)$
  - $p(\boldsymbol{x}|\mathcal{C}_2)$ is distribution over typical temperatures in Honolulu
    - e.g. $p(\boldsymbol{x}|\mathcal{C}_1) = \mathcal{N}(x; 25, 5)$
  - Class priors $p(\mathcal{C}_1) = 0.1, p(\mathcal{C}_2) = 0.9$
- $p(\mathcal{C}_1|x = 15) = \frac{0.0484 \times 0.1}{0.0484 \times 0.1 + 0.0108 \times 0.9} \approx 0.33$

Generalized Linear Models

- We can write the classifier in another form

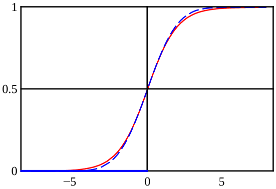
$$p(\mathcal{C}_1|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\boldsymbol{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

$$= \frac{1}{1 + \frac{p(\boldsymbol{x}|\mathcal{C}_2)p(\mathcal{C}_2)}{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1)}}$$

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

where $a = \ln \frac{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\boldsymbol{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$

Logistic Sigmoid



- The function $\sigma(a) = \dfrac{1}{1 + \exp(-a)}$ is known as the logistic sigmoid
- It squashes the real axis down to $[0, 1]$
- It is continuous and differentiable
- It avoids the problems encountered with the *too correct* least-squares error fitting

Multi-class Extension

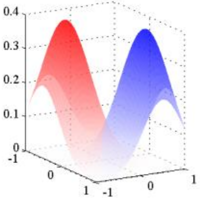
- There is a generalization of the logistic sigmoid to $K > 2$ classes:

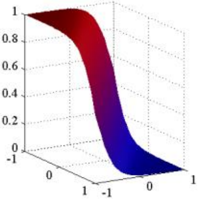
$$p(\mathcal{C}_k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\boldsymbol{x}|\mathcal{C}_j)p(\mathcal{C}_j)}$$
$$= \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

where $a_k = \ln p(\boldsymbol{x}|\mathcal{C}_k)p(\mathcal{C}_k)$

- a.k.a. softmax function
    - If some $a_k \gg a_j$, $p(\mathcal{C}_k|\boldsymbol{x})$ goes to 1

Gaussian Class-Conditional Densities





- Back to that a in the logistic sigmoid for 2 classes
- Let's assume the class-conditional densities $p(x|\mathcal{C}_k)$ are Gaussians, and have the same covariance matrix Σ:

$$p(\boldsymbol{x}|\mathcal{C}_k) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^\top \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\right\}$$

- a takes a simple form:

$$a = \ln \frac{p(\boldsymbol{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\boldsymbol{x}|\mathcal{C}_2)p(\mathcal{C}_2)} = \boldsymbol{w}^\top \boldsymbol{x} + w_0$$

  - Note that quadratic terms $\boldsymbol{x}^\top \Sigma^{-1} \boldsymbol{x}$ cancel

Maximum Likelihood Learning

- We can fit the parameters to this model using maximum likelihood
  - Parameters are $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma^{-1}, p(\mathcal{C}_1) \equiv \pi, p(\mathcal{C}_2) \equiv 1 - \pi$
  - Refer to as θ
- For a datapoint x_n from class $\mathcal{C}_1$ ($t_n = 1$):

$$p(\boldsymbol{x}_n, \mathcal{C}_1) = p(\mathcal{C}_1)p(\boldsymbol{x}_n|\mathcal{C}_1) = \pi\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_1, \Sigma)$$

- For a datapoint x_n from class $\mathcal{C}_2$ ($t_n = 0$):

$$p(\boldsymbol{x}_n, \mathcal{C}_2) = p(\mathcal{C}_2)p(\boldsymbol{x}_n|\mathcal{C}_2) = (1-\pi)\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_2, \Sigma)$$

Maximum Likelihood Learning

- The likelihood of the training data is:

$$p(\boldsymbol{t}|\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} [\pi \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_1, \Sigma)]^{t_n} [(1-\pi)\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_2, \Sigma)]^{1-t_n}$$

- As usual, $\ln(\cdot)$ is our friend:

$$l(\boldsymbol{t}|\theta) = \sum_{n=1}^{N} \underbrace{(t_n \ln \pi + (1-t_n)\ln(1-\pi)}_{\pi} + \underbrace{t_n \ln \mathcal{N}_1 + (1-t_n)\ln \mathcal{N}_2)}_{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma}$$

- Maximize for each separately

Maximum Likelihood Learning - Class Priors

- Maximization with respect to the class priors parameter π is straightforward:

$$\frac{\partial}{\partial \pi} l(t|\theta) = \sum_{n=1}^{N} \left(\frac{t_n}{\pi} - \frac{1 - t_n}{1 - \pi} \right)$$

$$\Rightarrow \pi = \frac{N_1}{N_1 + N_2}$$

- N_1 and N_2 are the number of training points in each class
- Prior is simply the fraction of points in each class

Maximum Likelihood Learning - Gaussian Parameters

- The other parameters can also be found in the same fashion
- Class means:

$$\mu_1 = \frac{1}{N_1} \sum_{n=1}^{N} t_n \boldsymbol{x}_n$$

$$\mu_2 = \frac{1}{N_2} \sum_{n=1}^{N} (1 - t_n) \boldsymbol{x}_n$$

  - Means of training examples from each class
- Shared covariance matrix:

$$\Sigma = \frac{N_1}{N} \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} (\boldsymbol{x}_n - \boldsymbol{\mu}_1)(\boldsymbol{x}_n - \boldsymbol{\mu}_1)^\top + \frac{N_2}{N} \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} (\boldsymbol{x}_n - \boldsymbol{\mu}_2)(\boldsymbol{x}_n - \boldsymbol{\mu}_2)^\top$$

  - Weighted average of class covariances

Probabilistic Generative Models Summary

- Fitting Gaussian using ML criterion is sensitive to outliers
- Simple linear form for a in logistic sigmoid occurs for more than just Gaussian distributions
  - Arises for any distribution in the exponential family, a large class of distributions

Outline

Probabilistic Discriminative Models

- Generative model made assumptions about form of class-conditional distributions (e.g. Gaussian)
  - Resulted in logistic sigmoid of linear function of $\boldsymbol{x}$
- Discriminative model - explicitly use functional form

$$p(\mathcal{C}_1|\boldsymbol{x}) = \frac{1}{1 + \exp(-\boldsymbol{w}^\top \boldsymbol{x} + w_0)}$$

  and find $\boldsymbol{w}$ directly
- For the generative model we had $\underbrace{2M + M(M+1)/2 + 1}_{\text{Means, variance, prior}}$ parameters
  - M is dimensionality of $\boldsymbol{x}$
- Discriminative model will have $M + 1$ parameters

Generative vs. Discriminative

- Generative models
  - Can generate synthetic example data
  - Perhaps accurate classification is equivalent to accurate synthesis
    - e.g. vision and graphics
  - Tend to have more parameters
  - Require good model of class distributions
- Discriminative models
  - Only usable for classification
  - Don't solve a harder problem than you need to
  - Tend to have fewer parameters
  - Require good model of decision boundary

Maximum Likelihood Learning - Discriminative Model

- As usual we can use the maximum likelihood criterion for learning

$$p(\boldsymbol{t}|\boldsymbol{w}) = \prod_{n=1}^{N} y_n^{t_n}\{1 - y_n\}^{1-t_n}, \quad \text{where } y_n = p(\mathcal{C}_1|\boldsymbol{x}_n)$$

- Taking ln and derivative gives:

$$\nabla l(\boldsymbol{w}) = \sum_{n=1}^{N} (t_n - y_n)\boldsymbol{x}_n$$

- This time no closed-form solution since $y_n = \sigma(\boldsymbol{w}^\top \boldsymbol{x})$
- Could use (stochastic) gradient descent
  - But there's a better iterative technique

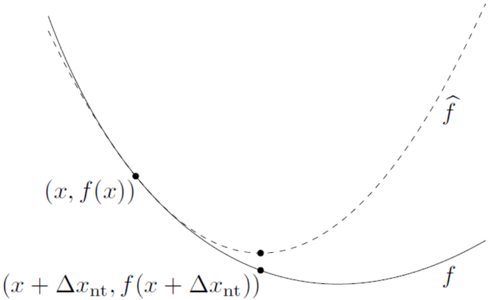
Iterative Reweighted Least Squares

- Iterative reweighted least squares (IRLS) is a descent method
  - As in gradient descent, start with an initial guess, improve it
  - Gradient descent - take a step (how large?) in the gradient direction
- IRLS is a special case of a Newton-Raphson method
  - Approximate function using second-order Taylor expansion:

$$\hat{f}(\boldsymbol{w}+\boldsymbol{v}) = f(\boldsymbol{w}) + \nabla f(\boldsymbol{w})^\top(\boldsymbol{v}-\boldsymbol{w}) + \frac{1}{2}(\boldsymbol{v}-\boldsymbol{w})^\top Hf(\boldsymbol{w})(\boldsymbol{v}-\boldsymbol{w})$$

  - Closed-form solution to minimize this is straight-forward: quadratic, derivatives linear
- In IRLS this second-order Taylor expansion ends up being a weighted least-squares problem, as in the regression case from last week
  - Hence the name IRLS

Newton-Raphson



- Figure from Boyd and Vandenberghe, *Convex Optimization*
    - Excellent reference, free for download online http://www.stanford.edu/~boyd/cvxbook/

Conclusion

- Readings: Ch. 4.1.1-4.1.4, 4.1.7, 4.2.1-4.2.2, 4.3.1-4.3.3
- Generalized linear models $y(\boldsymbol{x}) = f(\boldsymbol{w}^\top \boldsymbol{x} + w_0)$
- Threshold/max function for $f(\cdot)$
  - Minimize with least squares
  - Fisher criterion - class separation
  - Perceptron criterion - mis-classified examples
- Probabilistic models: logistic sigmoid / softmax for $f(\cdot)$
  - Generative model - assume class conditional densities in exponential family; obtain sigmoid
  - Discriminative model - directly model posterior using sigmoid (a. k. a. logistic regression, though classification)
  - Can learn either using maximum likelihood
- All of these models are limited to linear decision boundaries in feature space