$\LaTeX$ command declarations here.

EECS 545: Machine Learning
Lecture 08: Naive Bayes, GDA and LDA

- Instructor: **Jacob Abernethy**
- Date: Monday, February 1, 2016

Lecture Exposition Credit: Benjamin Bray, Valli Chockalingam

Outline

- Probabilistic Models
  - Generative Models
  - Discriminative Models
- Naive Bayes Classifiers
  - Independence Assumption
  - MLE and MAP Parameter Estimates
- Gaussian Discriminant Analysis
  - Quadratic Discriminant Analysis
  - Linear Discriminant Analysis
- Fisher's Linear Discriminant

Reading List

- Required:
  - **[PRML]**, §4.2: Probabilistic Generative Models
  - **[PRML]**, §4.3: Probabilistic Discriminative Models
  - **[PRML]**, §4.1: Discriminant Functions
  - **[MLAPP]**, §3.5: Naive Bayes Classifiers
  - **[MLAPP]**, §4.1: Gaussian Models
  - **[MLAPP]**, §4.2: Gaussian Discriminant Analysis

In this lecture, we will first talk about the some concepts of probabilistic models for classifiers, especially generative model and discriminant model. And we will introduce Naive Bayes classifier, which assumes independent features give label and is a classical classifier commonly used in spam email classification. The remaining two classifier we will talk about is Gaussian Discriminant Analysis and Fisher's Linear Discriminant. Gaussian Discriminant Analysis has quadratic boundary when different classes have different covariance matrix and has linear boundary when the covariance matrices are shared.

Probablistic Models

Probabilistic Models: Generative Models

- **Generative model** learns *class-conditional* $P(X|Y)$ and label densities $P(Y)$ from training data
- Perform prediction using the **posterior** via Bayes' Rule. For some new data $\mathbf{x}^{new}$

$$\begin{aligned} y &= \underset{k\in\{1,\ldots,K\}}{\arg\max}\, P(Y=k|X=\mathbf{x}^{new}) \\ &= \underset{k\in\{1,\ldots,K\}}{\arg\max}\, \frac{P(X=\mathbf{x}^{new}|Y=k)P(Y=k)}{P(X=\mathbf{x}^{new})} \\ &= \boxed{\underset{k\in\{1,\ldots,K\}}{\arg\max}\, P(X=\mathbf{x}^{new}|Y=k)P(Y=k)} \end{aligned}$$

  of which the last equality holds because the denominator $P(X=\mathbf{x}^{new})$ is independent of k.
- Basic idea of prediction is picking the label with largest posterior probability given its features $\mathbf{x}^{new}$.
- Why is this model called **generative**?
  - We learned *class-conditional probability* $P(X|Y)$ from training data.
  - $P(X|Y)$ is distribution of data X given label Y
  - So given some label Y, could **generate**/sample new data X from $P(X|Y)$.
- The *prior* $P(Y)$ encodes beliefs about popularity of each label
- By comparing the synthetic data and real data, we get a sense of how good our generative model is.

Probabilistic Models: Generative Models—Examples

- Simple examples:
  - Naive Bayes (Later)
  - Gaussian Discriminant Analysis (Later)
- More abstract examples:
  - Linear Regression
  - Most Bayesian models

Probabilistic Models: Discriminative Models

- Conversely, a **discriminative model** learns posterior $P(Y|X)$ directly from training data.
- Goal: select a hypothesis to *discriminates* between class labels.
- The prediction for some new data $\mathbf{x}^{new}$ is

$$y = \underset{k\in\{1,\ldots,K\}}{\arg\max}\, P(Y=k|X=\mathbf{x}^{new})$$

- Does not (necessarily) provide the ability to **generate** new random examples because unlike generative models, we have no idea what $P(X|Y)$ is.
- Allows us to focus purely on the classification task
- We will discuss the pros and cons of each model later.

Probabilistic Models: Discriminative Models—Property

- The discriminative approach will typically
  - have fewer parameters to estimate
  - make fewer assumptions about data distribution
    - Linear (logistic regression) vs quadratic (GDA) in the input dimension
  - make fewer generative assumptions about the data
    - However, reconstruction features from labels may require prior knowledge

Naive Bayes Classifiers

> Follows the approach taken by **[MLAPP]**

Naive Bayes: Problem

- We will use **Naive Bayes** to solve the following classification problem:
  - **Categorical** feature vector $\mathbf{x} = (x_1, x_2, \ldots, x_D)$ with length D
    - Each feature $x_d \in \{1, \ldots, M\}, \forall d = 1, \ldots, D$
  - Predict discrete class label $y \in \{1, 2, \ldots, C\}$
- For example, in **Spam Mail Classification**,
  - Predict whether an email is SPAM ($y = 1$) or HAM ($y = 0$)
  - Use words / metadata in the email as features
  - For simplicity, we can use **bag-of-words** features,
    - Assume fixed vocabulary V of size $|V| = D$
    - Feature x_d, for $d \in \{1, 2, \ldots, D\}$, indicates the existence of $d\text{th}$ word in the email
    - Eg. $x_d = 1$ if $d\text{th}$ word is in the email; $x_d = 0$ otherwise
    - In this case $M = 2$

Naive Bayes: Independence Assumption and Full model

- The essence of Naive Bayes is the **conditionally independence assumption**

$$P(\mathbf{x}|y = c) = \prod_{d=1}^{D} P(x_d|y = c)$$

  i.e., given the label, all features are independent.
- The **full generative** model of Naive Bayes is:

$$\begin{aligned} y &\sim \text{Categorical}(\pi) \\ x_d|y = c &\sim \text{Categorical}(\theta_{cd}) \quad \forall\, d = 1, \ldots, D \end{aligned}$$

  with parameters:
  - **Class priors** $\pi = (\pi_1, \ldots, \pi_C) \in \Delta^C$,
    - i.e. $P(y = c) = \pi_c, \forall c = 1, \ldots, C$
    - Δ^C is C-**simplex**. $\pi \in \Delta^C$ is saying that $\sum_{c=1}^{C} \pi_c = 1$ and $\pi_c \geq 0, \forall c = 1, \ldots, C$
  - **Class-conditional probabilities** $\theta_{cd} = (\theta_{cd1}, \ldots, \theta_{cdM}) \in \Delta^M$
    - i.e. $P(x_d = m|y = c) = \theta_{cdm}$ for every $d = 1, \ldots, D, m = 1, \ldots, M, c = 1, \ldots, C$
- Parameter π and θ are learned from training data.

> **Remark**
>
> - **NOTE** in definition and derivation of this lecture, we assume a more general case $x_d \in \{1, \ldots, M\}$ of which $M > 2$. But in spam email classification and the derivation in textbook, binary feature, i.e. $M = 2$, is used. So don't get confused!
> - When $M = 2$, $x_d|y = c$ is also Bernoulli distribution.

Naive Bayes: Prediction

- Given the independence assumption and full model, for some new data $\mathbf{x}^{\text{new}} = (x_1^{\text{new}}, \ldots, x_D^{\text{new}})$ we will classify based on

$$\begin{aligned} y &= \arg\max_{c \in \{1, \ldots, C\}} P(y = c|\mathbf{x} = \mathbf{x}^{\text{new}}) \\ &= \arg\max_{c \in \{1, \ldots, C\}} P(\mathbf{x} = \mathbf{x}^{\text{new}}|y = c) P(y = c) \\ &= \arg\max_{c \in \{1, \ldots, C\}} P(y = c) \prod_{d=1}^{D} P(x_d = x_d^{\text{new}}|y = c) \\ &= \boxed{\arg\max_{c \in \{1, \ldots, C\}} \pi_c \prod_{d=1}^{D} \theta_{cdx_d^{\text{new}}}} \end{aligned}$$

- If we assume $x_d^{\text{new}} \in \{1, \ldots, M\}, \forall d = 1, \ldots, D$, we could also express the above expression equivalently using **indicator function**

$$y = \arg\max_{c \in \{1, \ldots, C\}} \pi_c \prod_{d=1}^{D} \prod_{m=1}^{M} \theta_{cdm}^{\mathbb{I}(m = x_d^{\text{new}})}$$

- So as long as we learned parameter π and θ, we could classify.

Remark

- Indicator function
$$\mathbb{I}(m = x_d^{\text{new}}) = \begin{cases} 1 & \text{if } m = x_d^{\text{new}} \\ 0 & \text{otherwise} \end{cases}$$
- In inner product $\prod_{m=1}^{M} \theta_{cdm}^{\mathbb{I}(m=x_d^{\text{new}})}$, only $\theta_{cdx_d^{\text{new}}}$ is multiplied and all the other multipliers are 1 due to the power of indicator function.
- One thing to note is that the above classification criterion is the product of a series numbers smaller than 1 which will generate a rather small number. A better way is to take **logarithm** to transform product into summation and then compare.

Naive Bayes: Parameter Estimation

- **Goal:** Given training data $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, estimate **class-conditional probabilities** θ and **class priors** π.
- We will discuss the **MLE** and **MAP** parameter estimates.

Naive Bayes: Maximum Likelihood

- The **likelihood** for a single data case $(\mathbf{x}_n, y_n = c)$ is
$$\begin{aligned} & P((\mathbf{x}_n, y_n)|\pi, \theta) \\ &= P(y_n) \prod_{d=1}^{D} P(x_{nd}|y_n) \\ &= \prod_{c=1}^{C} P(y_n = c)^{\mathbb{I}(y_n=c)} \cdot \prod_{c=1}^{C} \prod_{d=1}^{D} \prod_{m=1}^{M} P(x_{nd} = m|y_n = c)^{\mathbb{I}(x_{nd}=m)\mathbb{I}(y_n=c)} \\ &= \prod_{c=1}^{C} \pi_c^{\mathbb{I}(y_n=c)} \cdot \prod_{c=1}^{C} \prod_{d=1}^{D} \prod_{m=1}^{M} \theta_{cdm}^{\mathbb{I}(x_{nd}=m)\mathbb{I}(y_n=c)} \end{aligned}$$
- Therefore, the **log-likelihood** is
$$\begin{aligned} & \log P((\mathbf{x}_n, y_n)|\pi, \theta) \\ &= \sum_{c=1}^{C} \mathbb{I}(y_n = c) \log \pi_c + \sum_{c=1}^{C} \sum_{d=1}^{D} \sum_{m=1}^{M} \mathbb{I}(x_{nd} = m)\mathbb{I}(y_n = c) \log \theta_{cdm} \end{aligned}$$
- The **log-likelihood** for all training data $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$ is
$$\begin{aligned} & \log P(\mathcal{D}|\pi, \theta) \\ &= \log \prod_{n=1}^{N} P((\mathbf{x}_n, y_n)|\pi, \theta) = \sum_{n=1}^{N} \log P((\mathbf{x}_n, y_n)|\pi, \theta) \\ &= \boxed{\sum_{n=1}^{N} \sum_{c=1}^{C} \mathbb{I}(y_n = c) \log \pi_c + \sum_{n=1}^{N} \sum_{c=1}^{C} \sum_{d=1}^{D} \sum_{m=1}^{M} \mathbb{I}(x_{nd} = m)\mathbb{I}(y_n = c) \log \theta_{cdm}} \end{aligned}$$

Naive Bayes: Maximum Likelihood

- With the constraints $\sum_{c=1}^{C} \pi_c = 1$ and $\sum_{m=1}^{M} \theta_{cdm} = 1$, we could maximize log-likelihood function $\log P(\mathcal{D}|\pi, \theta)$ using *Lagrange multiplier*. (Derivation is in the notes!)
- By maximizing log-likelihood function, we could have maximum likelihood estimators:
$$\hat{\pi}_c = \frac{N_c}{N} \quad \hat{\theta}_{cdm} = \frac{N_{cdm}}{N_c}$$
and
$$\hat{\pi} = (\hat{\pi}_1, \ldots, \hat{\pi}_c, \ldots, \hat{\pi}_C); \hat{\theta}_{cd} = (\hat{\theta}_{cd1}, \ldots, \hat{\theta}_{cdm}, \ldots, \hat{\theta}_{cdM})$$
  - $N =$ Number of examples in $\mathcal{D}$
  - $N_c =$ Number of examples in class c in $\mathcal{D}$
  - $N_{cdm} =$ Number of examples in class c with $x_d = m$ in $\mathcal{D}$
- Intuitive Interpretation
  - The class prior π is obtained from the density of each class $\{1, \ldots, C\}$ in $\mathcal{D}$
  - The class-conditional probability θ_{cd} is obtained from the density of $x_d \in \{1, \ldots, M\}$ among all examples in class c

Remark

- Derivation of **maximum likelihood estimator** $\hat{\pi}_c$
  - We have the following problem

$$\begin{cases} \max & \log P(\mathcal{D}|\pi,\theta) \\ \text{s.t.} & \sum_{c=1}^{C}\pi_c = 1 \end{cases} \overset{\text{equivalent to}}{\Longrightarrow} \begin{cases} \max & \sum_{n=1}^{N}\sum_{c=1}^{C}\mathbb{I}(y_n=c)\log\pi_c \\ \text{s.t.} & \sum_{c=1}^{C}\pi_c = 1 \end{cases}$$

    We drop the second term in $\log P(\mathcal{D}|\pi,\theta)$ because it doesn't depend on π_c
  - The lagrangian is

$$L(\pi,\lambda) = \sum_{n=1}^{N}\sum_{c=1}^{C}\mathbb{I}(y_n=c)\log\pi_c - \lambda\sum_{c=1}^{C}\pi_c - \lambda$$

-
  - Setting partial derivative with respect to π_c to 0, we have

$$\frac{\partial L(\pi,\lambda)}{\partial\pi_c} = 0 \quad \Rightarrow \quad \sum_{n=1}^{N}\mathbb{I}(y_n=c)\frac{1}{\pi_c} - \lambda = 0 \quad \Rightarrow \quad \pi_c = \frac{1}{\lambda}\sum_{n=1}^{N}\mathbb{I}(y_n=c)$$

  - Plug π_c back into the constraint $\sum_{c=1}^{C}\pi_c = 1$, we have

$$\frac{1}{\lambda}\sum_{c=1}^{C}\sum_{n=1}^{N}\mathbb{I}(y_n=c) = 1 \quad \Rightarrow \quad \lambda = \sum_{c=1}^{C}\sum_{n=1}^{N}\mathbb{I}(y_n=c)$$

  - Plug λ into $\pi_c = \frac{1}{\lambda}\sum_{n=1}^{N}\mathbb{I}(y_n=c)$, we have

$$\hat{\pi}_c = \frac{\sum_{n=1}^{N}\mathbb{I}(y_n=c)}{\sum_{n=1}^{N}\sum_{c=1}^{C}\mathbb{I}(y_n=c)} = \frac{N_c}{N}$$

- Derivation of maximum likelihood estimator $\hat{\theta}_{cdm}$
  - With the constraint $\sum_{m=1}^{M}\theta_{cdm} = 1$, using similar approach, we could have

$$\hat{\theta}_{cdm} = \frac{\sum_{n=1}^{N}\mathbb{I}(x_{nd}=m)\mathbb{I}(y_n=c)}{\sum_{n=1}^{N}\sum_{m=1}^{M}\mathbb{I}(x_{nd}=m)\mathbb{I}(y_n=c)} = \frac{N_{cdm}}{N_c}$$

    - Details are left as an exercise XD

Naive Bayes: Sparse Features

- **Problem:** When working with text, features are **sparse**:
  - In training, we only see a *small, small* fraction of words in the vocabulary
  - Moreover, we won't see all words exhibited across all classes
- This causes overfitting!
  - What if a word (e.g. "`subject:`") occurs in every training example of both classes?
  - Then if we encounter a new email without this word, our algorithm will crash.
  - What happens if that word never appears in testing? (*Black Swan Paradox*)

Naive Bayes: Priors

- **Solution:** Place Dirichlet priors on π and θ_{cd} to *smooth out* unknowns:

$$\begin{aligned} \pi &\sim \text{Dirichlet}(\alpha_1,\ldots,\alpha_C) \\ \theta_{cd} &\sim \text{Dirichlet}(\beta_{cd1},\ldots,\beta_{cdM}) && \forall\, c=1,\ldots,C; d=1,\ldots D \\ y &\sim \text{Categorical}(\pi) \\ x_d|y=c &\sim \text{Categorical}(\theta_{cd}) && \forall\, d=1,\ldots,D \end{aligned}$$

- **Dirichlet distribution** $\pi \sim \text{Dirichlet}(\alpha_1,\ldots,\alpha_C)$ defines the distribution of C-simplex $\pi = (\pi_1,\ldots,\pi_C)$ such that
  - $\pi_1,\ldots,\pi_C \geq 0$
  - $\pi_1 + \cdots + \pi_C = 1$
  - PDF $f(\pi_1,\ldots,\pi_C) = \frac{1}{B(\alpha)}\prod_{c=1}^{C}\pi_c^{\alpha_c-1}$, where the quantity $B(\alpha)$ is to normalize the distribution
- When $M=2$, θ_{cd} reduces to **Beta** distribution

Naive Bayes: MAP Estimate

- The **MAP parameter** estimates with priors
$$\pi \sim \text{Dirichlet}(\alpha_1, \ldots, \alpha_C) \qquad \theta_{cd} \sim \text{Dirichlet}(\beta_{cd1}, \ldots, \beta_{cdM})$$
are
$$\hat{\pi}_c = \frac{N_c + \alpha_c - 1}{N + \sum_{c'=1}^{C}(\alpha_{c'} - 1)} \quad \hat{\theta}_{cdm} = \frac{N_{cdm} + \beta_{cdm} - 1}{N_c + \sum_{m'=1}^{M}(\beta_{cdm'} - 1)}$$
- Proof is in the notes!
- The Dirichlet α and β_{cd} parameters turn out to be **pseudocounts**!
  - We assume we've seen α_c examples of class c beforehand
  - and β_{cdm} examples with $x_d = m$ in class c.
- The choice $\alpha_c = \beta_{cdm} = 1$ is referred to as **Laplace Smoothing**

Naive Bayes: Mean Estimate

- Note that the posterior of parameters still have **Dirichlet** distributions!
$$\pi|\mathcal{D} \sim \text{Dirichlet}(N_1 + \alpha_1, \ldots, N_C + \alpha_C) \qquad \theta_{cd}|\mathcal{D} \sim \text{Dirichlet}(N_{cd1} + \beta_{cd1}, \ldots, N_{cdM} + \beta_{cdM})$$
Proof for $\pi|\mathcal{D}$ is in the notes! Proof for $\theta_{cd}|\mathcal{D}$ is left as an exercise.
- If we pick the **mean** of posteriors as parameter estimate, we could get a slightly different result:
$$\bar{\pi}_c = \frac{N_c + \alpha_c}{N + \sum_{c'=1}^{C}\alpha_{c'}} \quad \bar{\theta}_{cdm} = \frac{N_{cdm} + \beta_{cdm}}{N_c + \sum_{m'=1}^{M}\beta_{cdm'}}$$
-1 terms no longer exist!
- You might see this version of estimate in **[MLAPP]** and some online materials
- Advantage of using mean estimate
  - Since posteriors still have Dirichlet distribution, we could use posterior as the prior for the next learning phase!
  - This can be helpful in sequential learning!

Remark

- Posterior also has Dirichlet distribution!
  - **Prior:**
$$\pi \sim \text{Dirichlet}(\alpha_1, \ldots, \alpha_C) \qquad f(\pi) = \frac{1}{B(\alpha)}\prod_{c=1}^{C}\pi_c^{\alpha_c - 1}$$
  - **Likelihood:**
$$P(\mathcal{D}|\pi) = \prod_{n=1}^{N}\prod_{c=1}^{C}\pi_c^{\mathbb{I}(y_n = c)} = \prod_{c=1}^{C}\pi_c^{\sum_{n=1}^{N}\mathbb{I}(y_n = c)} = \prod_{c=1}^{C}\pi_c^{N_c}$$
  - **Posterior**
$$\begin{aligned} f(\pi|\mathcal{D}) &= \frac{P(\mathcal{D}|\pi)f(\pi)}{P(\mathcal{D})} = \frac{1}{P(\mathcal{D})}\prod_{c=1}^{C}\pi_c^{N_c} \cdot \frac{1}{B(\alpha)}\prod_{c=1}^{C}\pi_c^{\alpha_c - 1} \\ &= \frac{1}{P(\mathcal{D})B(\alpha)}\prod_{c=1}^{C}\pi_c^{N_c + \alpha_c - 1} \\ &= \frac{1}{B'(\alpha)}\prod_{c=1}^{C}\pi_c^{N_c + \alpha_c - 1} \end{aligned}$$
    of which $B'(\alpha) = P(\mathcal{D})B(\alpha)$.
  - From the expression of $f(\pi|\mathcal{D})$, we could see posterior also has **Dirichlet** distribution
$$\pi|\mathcal{D} \sim \text{Dirichlet}(N_1 + \alpha_1, \ldots, N_C + \alpha_C)$$

- Conjugate Prior
  - If posterior and prior are in the same distribution family with respect to some likelihood, we could call this distribution **conjugate prior** for that likelihood.
  - This is useful because we could take the posterior as the prior for next learning phase, which enables us to do sequential Bayesian learning.
  - In our case, we have shown Dirichlet distribution is the conjugate prior of the multinomial distribution!

- Derivation of **MAP estimate** $\hat{\pi}_{cMAP}$
  - MAP estimate is obtained by maximizing the posterior $f(\pi|\mathcal{D})$

$$\begin{cases} \max & \prod_{c=1}^{C} \pi_c^{N_c+\alpha_c-1} \\ \text{s.t.} & \sum_{c=1}^{C} \pi_c = 1 \end{cases} \overset{\text{equivalent to}}{\Longrightarrow} \begin{cases} \max & \sum_{c=1}^{C} (N_c+\alpha_c-1)\log \pi_c \\ \text{s.t.} & \sum_{c=1}^{C} \pi_c = 1 \end{cases}$$

  - Recall when deriving maximum likelihood estimator, we solved the following problem

$$\begin{cases} \max & \sum_{c=1}^{C}\sum_{n=1}^{N} \mathbb{I}(y_n=c)\log \pi_c \\ \text{s.t.} & \sum_{c=1}^{C} \pi_c = 1 \end{cases} \longrightarrow \hat{\pi}_{cMLE} = \frac{\sum_{n=1}^{N}\mathbb{I}(y_n=c)}{\sum_{n=1}^{N}\sum_{c=1}^{C}\mathbb{I}(y_n=c)} = \frac{N_c}{N}$$

  - So the solution of our current problem can be easily read off

$$\hat{\pi}_{cMAP} = \frac{N_c+\alpha_c-1}{\sum_{c'=1}^{C}(N_c+\alpha'_c-1)} = \frac{N_c+\alpha_c-1}{N+\sum_{c'=1}^{C}(\alpha'_c-1)}$$

- Derivation of **MAP estimator** $\hat{\theta}_{cdm}$ is left as an exercise! Approach is exactly the same as deriving $\hat{\pi}_c$.

Naive Bayes: Challenge

How should we deal with *out-of-vocabulary* words, i.e. words in the test set that we didn't include in the vocabulary during training?

Naive Bayes: Is Independence Justified?

- Naive Bayes assumes features contribute *independently* to the class label.
  - This is the *simplest possible* generative model... and an **extreme** assumption...
- This model is *naive* because we would never expect features to be independent!
  - We are completely ignoring correlations between variables!
- It seems not to matter that independence is often false...
  - Naive Bayes performs surprisingly well on real-world data
  - Naive Bayes is often used as a baseline
- One reason is that the model is quite simple
  - Only $O(CD)$ parameters, for C classes and D features
  - Hence relatively immune to overfitting
- There are some interesting theoretical justifications, too!
  - Zhang, 2004, "The optimality of naive Bayes. (http://www.cs.unb.ca/profs/hzhang/publications/FLAIRS04ZhangH.pdf)"
  - Domingos & Pazzani, 1997, "On the optimality of the simple Bayesian classifier under zero-one loss. (http://web.cs.ucdavis.edu/~vemuri/classes/ecs271/Bayesian.pdf)".
- Apparently, dependencies between variables can "cancel out"...

Discriminant Functions

Discriminant Functions

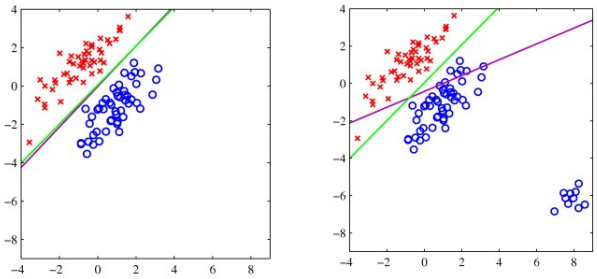
- A **discriminant function** maps an input vector to one of C classes.
  - Characterized by a **decision boundary**
  - We will mainly focus on **linear discriminants**
- A **linear discriminant function** $y(x) = \mathbf{w}^T x + w_0$ divides two classes in feature space
  - weight vector $\mathbf{w} \in \mathbb{R}^D$, bias $w_0 \in \mathbb{R}$
  - In two dimensions, this discriminant function is a line
  - In three dimensions, a plane
  - In general, a **separating hyperplane**!
- Assign x to C_1 if $y(x) \geq 0$ and to C_0 otherwise.
- Recall in Lec07, we used discriminant functions in perceptron and logistic regression.
  - In perceptron, $y = \text{sign}(\mathbf{w}^T\phi(\mathbf{x}))$
  - In logistic regression, $y = \mathbb{I}(\mathbf{w}^T\phi(\mathbf{x}) \geq 0)$
- See **[PRML]** section 4.1 for a discussion of multiclass problems.

Discriminant Functions: How to select weights $\mathbf{w}$?

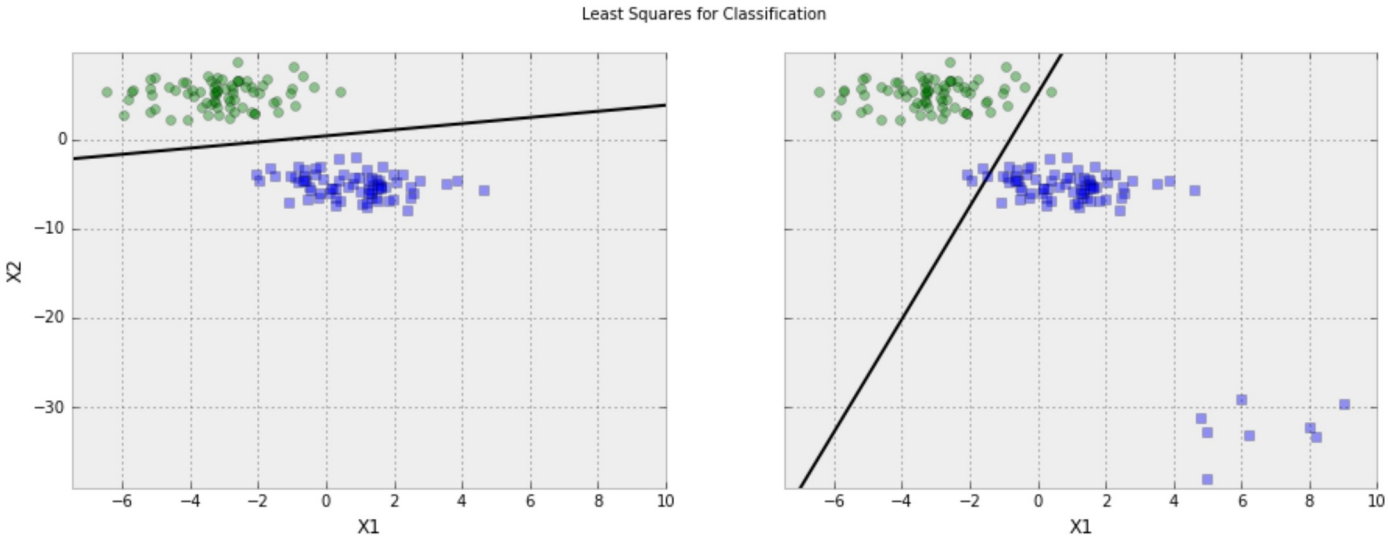
One approach: Use **least Squares** for classification by setting the target values to be 1 and 0. Choose $\mathbf{w}$ that minimizes squared error

Bad Idea:

- Least squares is too sensitive to outliers. (*Why?*)



Discriminant Functions: Least Squares is Sensitive to Noise



Discriminant Functions: How to select weights $\mathbf{w}$?

- **Better Idea:** We'll cover the following models
  - Gaussian Discriminant Analysis
    - Quadratic Discriminant Analysis
    - Linear Discriminant Analysis
  - Fisher's Linear Discriminant

Gaussian Discriminant Analysis

Review: Multivariate Gaussian

- A **normally distributed** random vector $\mathbf{x}$ with mean $\mu \in \mathbb{R}^D$ and positive semi-definite covariance matrix $\Sigma \in \mathbb{R}^{D \times D}$ has PDF:

$$\mathcal{N}(x|\mu, \Sigma) = \frac{1}{Z}\exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\}$$

  with normalization constant $Z = (2\pi)^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}$.
- One-dimensional case:

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}$$

- When covariance matrix Σ is diagonal, elements in $\mathbf{x}$ are *independent* of each other. Otherwise, elements are *correlated*.

Gaussian Discriminant Analysis: Model

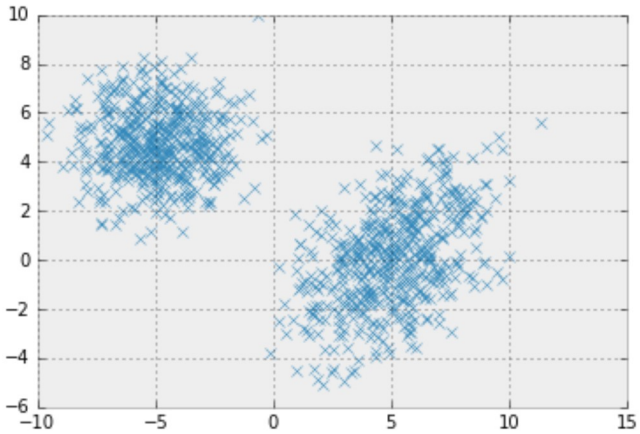
- Generative probabilistic model of Gaussian Discriminant Analysis (**GDA**)
  - Predict discrete label $y \in \{1, \ldots, C\}$ from *continuous* features $\mathbf{x}$
    - Recall in Naive Bayes, the features are *categorical* and *discrete*.
  - Class-conditional densities are multivariate Gaussian

$$y \sim \text{Categorical}(\pi)$$
$$\mathbf{x} \mid y = c \sim \mathcal{N}(\mu_c, \Sigma_c)$$

    with per-class means μ_c and covariance matrices Σ_c
- GDA models **feature correlations**
  - Recall in Naive Bayes, features are **independent** given the label
- However, if all covariance matrices are diagonal, then GDA can be seen as the continuous analogue of Naive Bayes!

Gaussian Discriminant Analysis: Data

- Because GDA is a *generative* model, when the parameters π, μ_c and Σ_c are known or learned, we can *generate* fake data!



Gaussian Discriminant Analysis: Classification

- Classify a feature vector $\mathbf{x}$ using the **posterior mode**:

$$\begin{aligned}
y &= \arg\max_c \log P(y = c|\mathbf{x}) = \arg\max_c \log P(\mathbf{x}|y = c)P(y = c) \\
&= \arg\max_c \left[\log P(y = c|\pi) + \log P(\mathbf{x}|\mu_c, \Sigma_c)\right] \\
&= \arg\max_c \log \pi_c + \log \left[(2\pi)^{-\frac{D}{2}} |\Sigma_c|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \mu_c)^T \Sigma_c^{-1}(\mathbf{x} - \mu_c)\right\}\right] \\
&= \arg\max_c \boxed{\log \pi_c - \frac{1}{2}\log |\Sigma_c| - \frac{1}{2}(\mathbf{x} - \mu_c)^T \Sigma_c^{-1}(\mathbf{x} - \mu_c)} \\
&= \arg\min_c \boxed{\frac{1}{2}\log |\Sigma_c| + \frac{1}{2}(\mathbf{x} - \mu_c)^T \Sigma_c^{-1}(\mathbf{x} - \mu_c) - \log \pi_c}
\end{aligned}$$

- The probability of $\mathbf{x}$ under each class-conditional density is the distance from x to the center μ_c of each class, using **Mahalanobis distance** (https://en.wikipedia.org/wiki/Mahalanobis_distance)!
- GDA can be thought of as a **nearest-centroid classifier**!

Gaussian Discriminant Analysis: Quadratic Discriminant Analysis

- The decision boundary is **quadratic** *in general*. We will show the binary case..
  - Consider two class $c = 1$ and $c = 0$, we could simply classify using

$$\log P(y = 1|\mathbf{x}) \underset{c=1}{\overset{c=0}{\lessgtr}} \log P(y = 0|\mathbf{x})$$

By plugging in derivation above, we have

$$\frac{1}{2}\log|\Sigma_1| + \frac{1}{2}(\mathbf{x} - \mu_1)^T\Sigma_1^{-1}(\mathbf{x} - \mu_1) - \log\pi_1$$
$$\underset{c=0}{\overset{c=1}{\lessgtr}} \frac{1}{2}\log|\Sigma_0| + \frac{1}{2}(\mathbf{x} - \mu_0)^T\Sigma_0^{-1}(\mathbf{x} - \mu_0) - \log\pi_0$$

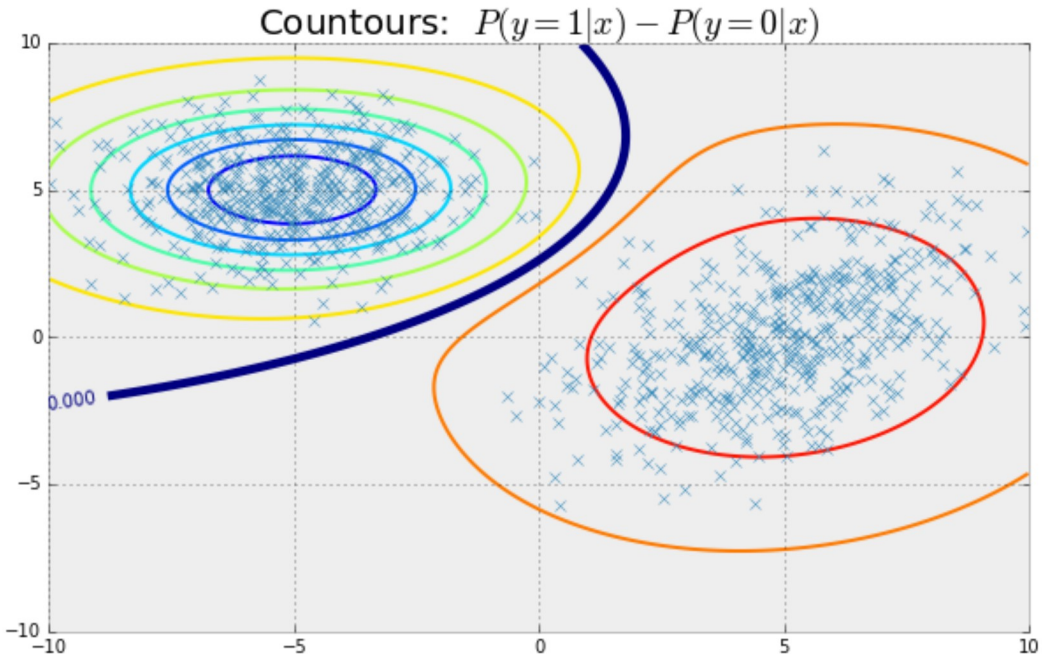
which can be transformed into

$$\frac{1}{2}\mathbf{x}^T\left(\Sigma_1^{-1} - \Sigma_0^{-1}\right)\mathbf{x} - 2\left(\mu_0^T\Sigma_0^{-1} + \mu_1^T\Sigma_1^{-1}\right)\mathbf{x}$$
$$+ \mu_1^T\Sigma_1^{-1}\mu_1 + \mu_0^T\Sigma_0^{-1}\mu_0 - \log\frac{\pi_1}{\pi_0} + \frac{1}{2}\log\frac{|\Sigma_1|}{|\Sigma_0|} \underset{c=0}{\overset{c=1}{\lessgtr}} 0$$

- A **quadratic** boundary!

Gaussian Discriminant Analysis: Quadratic Discriminant Analysis

- The bold blue line is the decision boundary



Gaussian Discriminant Analysis: Linear Discriminant Analysis

- We have just showed

$$\log P(\mathbf{x}|y = c)P(y = c) = -0.5\log|\Sigma_c| - 0.5(\mathbf{x} - \mu_c)^T\Sigma_c^{-1}(\mathbf{x} - \mu_c) + \log\pi_c$$

so

$$P(y = c|\mathbf{x}) \propto P(\mathbf{x}|y = c)P(y = c)$$
$$= \pi_c|\Sigma_c|^{-0.5}\exp\left[-0.5\mathbf{x}^T\Sigma_c^{-1}\mathbf{x} + \mu_c^T\Sigma_c^{-1}\mathbf{x} - 0.5\mu_c^T\Sigma_c^{-1}\mu_c\right]$$

- When **covariance matrices are shared**, i.e. $\Sigma_c = \Sigma, \forall c = 1, \ldots, C$

$$P(y = c|\mathbf{x}) \propto \boxed{\exp\left[\mu_c^T\Sigma^{-1}\mathbf{x} - 0.5\mu_c^T\Sigma^{-1}\mu_c + \log\pi_c\right]}$$

We dropped some terms because they don't depend on c

- Particularly, **softmax function** will show up here

$$P(y = c|\mathbf{x}) = \frac{\exp\left[\mu_c^T\Sigma^{-1}\mathbf{x} - 0.5\mu_c^T\Sigma^{-1}\mu_c + \log\pi_c\right]}{\sum_{c'=1}^{C}\exp\left[\mu_{c'}^T\Sigma^{-1}\mathbf{x} - 0.5\mu_{c'}^T\Sigma^{-1}\mu_{c'} + \log\pi_{c'}\right]} = \frac{\exp\left[\beta_c^T\mathbf{x} + \gamma_c\right]}{\exp\left[\sum_{c'=1}^{C}\beta_{c'}^T\mathbf{x} + \gamma_{c'}\right]}$$

of which $\beta_c = \mu_c^T\Sigma^{-1}$ and $\gamma_{c'} = -0.5\mu_c^T\Sigma^{-1}\mu_c + \log\pi_c$

Gaussian Discriminant Analysis: Linear Discriminant Analysis

- For binary classification with shared covariance matrix, we will get a **linear classifier**
  - Consider two class $c = 1$ and $c = 0$, we could classify using log-odds

$$\log \frac{P(y=1|\mathbf{x})}{P(y=0|\mathbf{x})} \underset{c=1}{\overset{c=0}{\lessgtr}} 0$$

since $P(y=c|\mathbf{x}) \propto \exp\left[\mu_c^T \Sigma^{-1} \mathbf{x} - 0.5 \mu_c^T \Sigma^{-1} \mu_c + \log \pi_c\right]$, the log-odds criterion could be transformed into

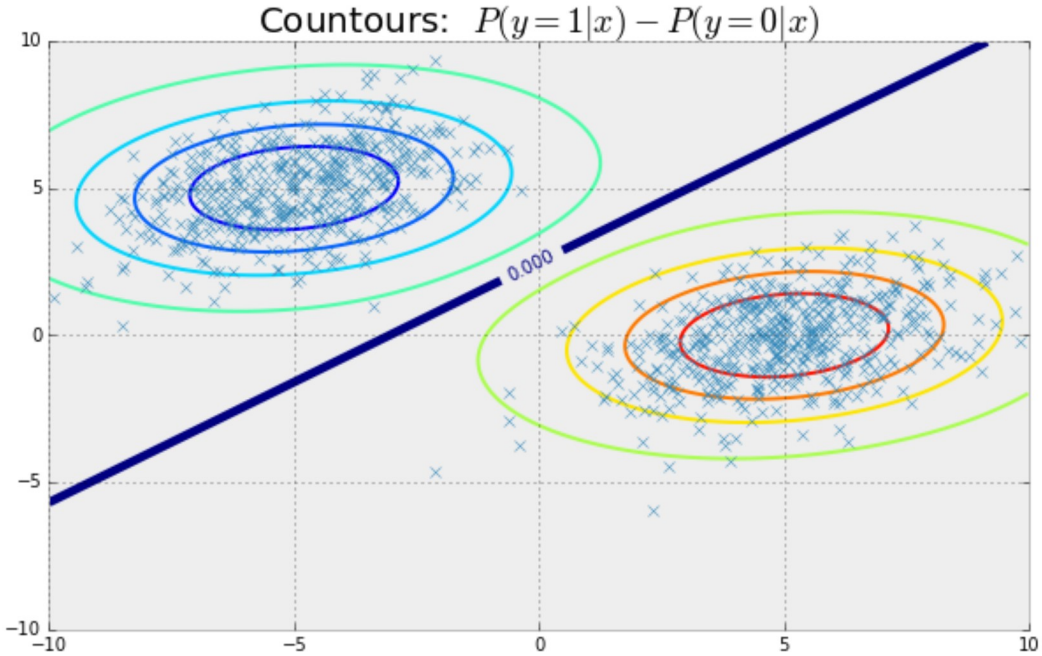
$$(\mu_1 - \mu_0)^T \Sigma^{-1} \mathbf{x} + \left[\log \frac{\pi_1}{\pi_0} - 0.5 \mu_1^T \Sigma^{-1} \mu_1 + 0.5 \mu_0^T \Sigma^{-1} \mu_0\right] \underset{c=1}{\overset{c=0}{\lessgtr}} 0$$

  - Log-odds criterion is finally transformed into determining the sign of liear function, which is quite similar to **Logistic Regression**
- When covariance Σ is *identity matrix*, the decision boundary becomes

$$(\mu_1 - \mu_0)^T \mathbf{x} + \left[\log \frac{\pi_1}{\pi_0} - 0.5 \|\mu_1\|_2 + 0.5 \|\mu_0\|_2\right]$$

which is *perpendicular* to line connecting two centroids μ_1 and μ_0 of class $c = 1$ and $c = 0$.

Gaussian Discriminant Analysis: Linear Discriminant Analysis



Gaussian Discriminant Analysis: Parameter Estimation

- One thing we haven't mention is **estimation of parameters π, μ and Σ
- Using **maximum likelihood estimation** similar to derivation in Naive Bayes, for training data $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$ we could get

$$\hat{\pi}_c = \frac{N_c}{N} \qquad \hat{\mu}_c = \frac{1}{N_c} \sum_{n=1}^{N} \mathbf{x}_n \mathbb{I}(y_n = c) \qquad \hat{\Sigma}_c = \frac{1}{N_c} \sum_{n=1}^{N} (\mathbf{x}_n - \hat{\mu}_c)(\mathbf{x}_n - \hat{\mu}_c)^T \mathbb{I}(y_n = c)$$

  - N = Number of examples in $\mathcal{D}$
  - N_c = Number of examples in class c in $\mathcal{D}$
- Left as an exercise!

Remark

- Derivation of maximum likelihood estimator
  - The likelihood is

$$\begin{aligned} P(\mathcal{D}|\pi,\mu,\Sigma) &= \prod_{n=1}^N P((\mathbf{x}_n,y_n)|\pi,\mu,\Sigma) = \prod_{n=1}^N P(y_n)P(\mathbf{x}_n|y_n) \\ &= \prod_{n=1}^N \pi_{y_n}(2\pi)^{-D/2}|\Sigma_{y_n}|^{-1/2}\exp\left[-0.5(\mathbf{x}_n-\mu_{y_n})^T\Sigma_{y_n}^{-1}(\mathbf{x}_n-\mu_{y_n})\right] \\ &= \prod_{n=1}^N \prod_{c=1}^C \left\{\pi_c(2\pi)^{-D/2}|\Sigma_c|^{-1/2}\exp\left[-0.5(\mathbf{x}_n-\mu_c)^T\Sigma_c^{-1}(\mathbf{x}_n-\mu_c)\right]\right\}^{\mathbb{I}(y_n=c)} \end{aligned}$$

  - Log-likelihood is

$$\begin{aligned} \log P(\mathcal{D}|\pi,\mu,\Sigma) = &\sum_{n=1}^N\sum_{c=1}^C \mathbb{I}(y_n=c)\log\pi_c - 0.5\sum_{n=1}^N\sum_{c=1}^C \mathbb{I}(y_n=c)\log|\Sigma_c| \\ &- 0.5\sum_{n=1}^N\sum_{c=1}^C \mathbb{I}(y_n=c)(\mathbf{x}_n-\mu_c)^T\Sigma_c^{-1}(\mathbf{x}_n-\mu_c) \end{aligned}$$

- To derive $\hat{\pi}_c$, we have the following problem

$$\begin{cases} \max & \log P(\mathcal{D}|\pi,\mu,\Sigma) \\ \text{s.t.} & \sum_{c=1}^C \pi_c = 1 \end{cases} \xrightarrow{\text{equivalent to}} \begin{cases} \max & \sum_{n=1}^N\sum_{c=1}^C \mathbb{I}(y_n=c)\log\pi_c \\ \text{s.t.} & \sum_{c=1}^C \pi_c = 1 \end{cases}$$

  Following the derivation in Naive Bayes, we could have

$$\hat{\pi}_c = \frac{\sum_{n=1}^N \mathbb{I}(y_n=c)}{\sum_{n=1}^N\sum_{c=1}^C \mathbb{I}(y_n=c)} = \frac{N_c}{N}$$

- Derivation of $\hat{\mu}_c$

$$\begin{aligned} \hat{\mu}_c &= \arg\min_{\mu_c} \sum_{n=1}^N\sum_{c=1}^C \mathbb{I}(y_n=c)(\mathbf{x}_n-\mu_c)^T\Sigma_c^{-1}(\mathbf{x}_n-\mu_c) \\ &= \arg\min_{\mu_c} \underbrace{\sum_{n=1}^N \mathbb{I}(y_n=c)(\mathbf{x}_n-\mu_c)^T\Sigma_c^{-1}(\mathbf{x}_n-\mu_c)}_{L_1} \end{aligned}$$

  It's derivative is

$$\begin{aligned} \mathrm{d}L_1/\,\mathrm{d}\mu_c &= \sum_{n=1}^N \mathbb{I}(y_n=c)\left[2\Sigma^{-1}(\mathbf{x}_n-\mu_c)\right] \\ &= 2\Sigma^{-1}\sum_{n=1}^N \mathbb{I}(y_n=c)(\mathbf{x}_n-\mu_c) \end{aligned}$$

  Setting $\partial L_1/\partial\mu_c = 0$, we could have

$$\hat{\mu}_c = \frac{\sum_{n=1}^N \mathbb{I}(y_n=c)\mathbf{x}_n}{\sum_{n=1}^N \mathbb{I}(y_n=c)} = \frac{1}{N_c}\sum_{n=1}^N \mathbb{I}(y_n=c)\mathbf{x}_n$$

- Derivation of $\hat{\Sigma}_c$

$$\begin{aligned} \hat{\Sigma}_c &= \arg\min_{\Sigma_c} \sum_{n=1}^N\sum_{c=1}^C \mathbb{I}(y_n=c)\log|\Sigma_c| + \sum_{n=1}^N\sum_{c=1}^C \mathbb{I}(y_n=c)(\mathbf{x}_n-\mu_c)^T\Sigma_c^{-1}(\mathbf{x}_n-\mu_c) \\ &= \arg\min_{\Sigma_c} \sum_{n=1}^N \mathbb{I}(y_n=c)\log|\Sigma_c| + \sum_{n=1}^N \mathbb{I}(y_n=c)(\mathbf{x}_n-\mu_c)^T\Sigma_c^{-1}(\mathbf{x}_n-\mu_c) \\ &= \arg\min_{\Sigma_c} \log|\Sigma_c|\sum_{n=1}^N \mathbb{I}(y_n=c) + \sum_{n=1}^N \mathbb{I}(y_n=c)\mathrm{trace}\left[(\mathbf{x}_n-\mu_c)^T\Sigma_c^{-1}(\mathbf{x}_n-\mu_c)\right] \\ &= \arg\min_{\Sigma_c} \log|\Sigma_c|\sum_{n=1}^N \mathbb{I}(y_n=c) + \sum_{n=1}^N \mathbb{I}(y_n=c)\mathrm{trace}\left[(\Sigma_c^{-1}\mathbf{x}_n-\mu_c)(\mathbf{x}_n-\mu_c)^T\right] \\ &= \arg\min_{\Sigma_c} \underbrace{-\log|\Sigma_c^{-1}|\sum_{n=1}^N \mathbb{I}(y_n=c) + \mathrm{trace}\left[\Sigma_c^{-1}\sum_{n=1}^N \mathbb{I}(y_n=c)(\mathbf{x}_n-\mu_c)(\mathbf{x}_n-\mu_c)^T\right]}_{L_2} \end{aligned}$$

- The derivative with respect to Σ_c^{-1} is

$$\mathrm{d}L_2/\,\mathrm{d}\Sigma_c^{-1} = -\Sigma_c \sum_{n=1}^{N} \mathbb{I}(y_n = c) + \sum_{n=1}^{N} \mathbb{I}(y_n = c)(\mathbf{x}_n - \mu_c)(\mathbf{x}_n - \mu_c)^T$$

Setting $\mathrm{d}L_2/\,\mathrm{d}\Sigma_c^{-1} = 0$, we could have

$$\hat{\Sigma} = \frac{\sum_{n=1}^{N} \mathbb{I}(y_n = c)(\mathbf{x}_n - \hat{\mu}_c)(\mathbf{x}_n - \hat{\mu}_c)^T}{\sum_{n=1}^{N} \mathbb{I}(y_n = c)} = \frac{1}{N_c} \sum_{n=1}^{N} \mathbb{I}(y_n = c)(\mathbf{x}_n - \hat{\mu}_c)(\mathbf{x}_n - \hat{\mu}_c)^T$$

- We used the fact

$$\mathrm{d}\,\mathrm{trace}(AX)/\,\mathrm{d}X = A^T \qquad \mathrm{d}\log|X|/\,\mathrm{d}X = (X^{-1})^T$$

GDA vs. Logistic Regression

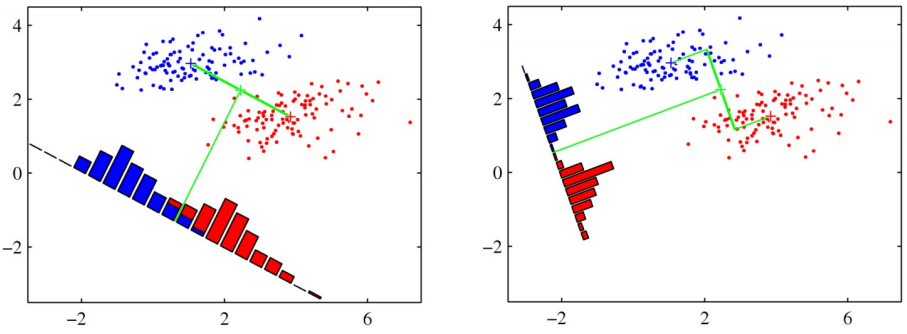
- For a D-dimensional feature space,
  - Logistic Regression must fit D parameters.
  - Gaussian Discriminant Analysis has to fit
    - $C \cdot D$ parameters for each class mean μ_c
    - $D(D+1)/2$ params for the shared covariance mtx
  - Logistic regression has fewer parameters and is more flexible about data distribution!
  - GDA makes stronger modeling assumptions, and works better (only) when assumptions hold (**approximately**)

Fisher's Linear Discriminant

- For **binary** classification

Fisher's Linear Discriminant

- Use $\mathbf{w}$ to project x onto one dimension
  - If projection $\mathbf{w}^T\mathbf{x} \geq -w_0$ then assign $\mathbf{x}$ to class 1, else to class 0.
- Select a projection $\mathbf{w}$ that best "separates" the classes, i.e., both
  - Maximizes inter-class separation (distance between means)
  - Minimizes in-class variance
- **Left**: Maximizing inter-class separation alone
- **Right**: Maximizing both inter-class separation and minimizing in-class variance



Fisher's Linear Discriminant: Objective

- **Goal 1:** Maximize the *distance* between *projected* means m_1 and m_0 of two classes

$$\text{maximize} \quad m_1 - m_0 \equiv w^T(\mathbf{m}_1 - \mathbf{m}_0)$$

$$\mathbf{m}_1 = \frac{1}{N_1}\sum_{i:y_i=1} x_i \qquad \mathbf{m}_0 = \frac{1}{N_0}\sum_{i:y_i=0} x_i$$

- **Goal 2:** Minimize the *variance within classes*

$$\text{minimize} \quad s_1^2 + s_0^2 \equiv \sum_{i:y_i=1}(\mathbf{w}^T\mathbf{x}_i - m_1)^2 + \sum_{i:y_i=0}(\mathbf{w}^T\mathbf{x}_i - m_0)^2$$

- **Objective Function:** Encodes both *Goal 1* and *Goal 2*:

$$\text{maximize} \quad J(\mathbf{w}) = \frac{(m_1 - m_0)^2}{s_1^2 + s_0^2} = \frac{\mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_0)(\mathbf{m}_1 - \mathbf{m}_0)^T\mathbf{w}}{s_1^2 + s_0^2} = \frac{\mathbf{w}^T S_B \mathbf{w}}{s_1^2 + s_0^2}$$

  of which $S_B = (\mathbf{m}_1 - \mathbf{m}_0)(\mathbf{m}_1 - \mathbf{m}_0)^T$ is the **between-class scatter matrix**
- S_B tells us
  - tells us how much the means of different features covary, or
  - how correlated they are (without regard for scaling)

Fisher's Linear Discriminant: Objective

- Define the **within-class scatter matrix**

$$S_W = \sum_{i:y_i=1}(x_i - \mathbf{m}_1)(x_i - \mathbf{m}_1)^T + \sum_{i:y_i=0}(x_i - \mathbf{m}_0)(x_i - \mathbf{m}_0)^T$$

- As an exercise, check that

$$s_1^2 + s_0^2 = \mathbf{w}^T S_W \mathbf{w}$$

- We can know rewrite the objective explicitly in terms of $\mathbf{w}$,

$$\text{maximize} \quad J(\mathbf{w}) = \frac{(m_1 - m_0)^2}{s_1^2 + s_0^2} = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}$$

- The solution $\mathbf{w} = S_W^{-1}(\mathbf{m}_1 - \mathbf{m}_0)$. Derivation is in the notes.

Remark

- Derivation of $\mathbf{w}$
  - The derivative of $J(\mathbf{w})$ is

$$\frac{\mathrm{d}J(\mathbf{w})}{\mathrm{d}\mathbf{w}} = \frac{\mathrm{d}}{\mathrm{d}\mathbf{w}}\left(\mathbf{w}^T S_B \mathbf{w} \cdot \frac{1}{\mathbf{w}^T S_W \mathbf{w}}\right) = -\frac{\mathbf{w}^T S_B \mathbf{w}}{(\mathbf{w}^T S_W \mathbf{w})^2} S_W \mathbf{w} + \frac{S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}$$

  - Setting $\frac{\mathrm{d}J(\mathbf{w})}{\mathrm{d}\mathbf{w}} = 0$, we could have

$$(\mathbf{w}^T S_B \mathbf{w}) S_W \mathbf{w} = (\mathbf{w}^T S_W \mathbf{w}) S_B \mathbf{w}$$

    Since we only care the direction of $\mathbf{w}$ and don't care its magnitude, so this equation can be reduced to

$$S_W \mathbf{w} = S_B \mathbf{w}$$

    Since $S_B \mathbf{w} = (\mathbf{m}_1 - \mathbf{m}_0)(\mathbf{m}_1 - \mathbf{m}_0)^T \mathbf{w}$ and $(\mathbf{m}_1 - \mathbf{m}_0)^T \mathbf{w}$ is a scalar, we could see $S_B \mathbf{w}$ has direction $\mathbf{m}_1 - \mathbf{m}_0$. Recall we only care the direction of $\mathbf{w}$, so we could further write the equation as

$$S_W \mathbf{w} = \mathbf{m}_1 - \mathbf{m}_0$$

    and the result is

$$\mathbf{w} = (S_W)^{-1}(\mathbf{m}_1 - \mathbf{m}_0)$$

Summary of Classifiers

- **Logistic Regression**
  - Provides model for $P(y|\mathbf{x})$ using sigmoid
  - No explicit model for $P(\mathbf{x}|y)$
- **Naive Bayes**
  - Provides a full model for $P(\mathbf{x}|y)$ and $P(y)$
  - Assumes independence between features *conditioned on* target y
  - Typically requires discrete data (can generalize to continuous spaces)
  - ML estimates are pretty straightforward
- **Gaussian Discriminant Analysis**
  - Gives full model $P(\mathbf{x}|y)$ and $P(y)$ via *multivariate Gaussian*
  - Requires estimating μ, Σ for each class
  - When we include additional assumption that all class covariances matrices are identical, we get *Linear Discriminant Analysis*, and decision threshold becomes linear.
- **Perceptron**
  - No probability model on $\mathbf{x}, y$
  - Can prove algorithmic benefits (i.e. guaranteed convergence) under *margin assumption*