In [2]:

```
# plotting
%matplotlib inline
from matplotlib import pyplot as plt;
import matplotlib as mpl;
from matplotlib import mlab;
if "bmh" in plt.style.available: plt.style.use("bmh");

# scientific
import numpy as np;
import scipy as scp;
import scipy.stats;

# python
import random;

# rise config
from notebook.services.config import ConfigManager
cm = ConfigManager()
cm.update('livereveal', {
              'theme': 'simple',
              'start_slideshow_at': 'selected',
              'transition':'fade',
              'scroll':False
})
```

Out[2]:

```
{'scroll': False,
 'start_slideshow_at': 'selected',
 'theme': 'simple',
 'transition': 'fade'}
```

$\LaTeX$ command declarations here.

EECS 545: Machine Learning

Lecture 07: More Linear Classifiers

Naive Bayes, GDA, LDA

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

Lecture Exposition Credit: Benjamin Bray, Valli Chockalingam

Logistics

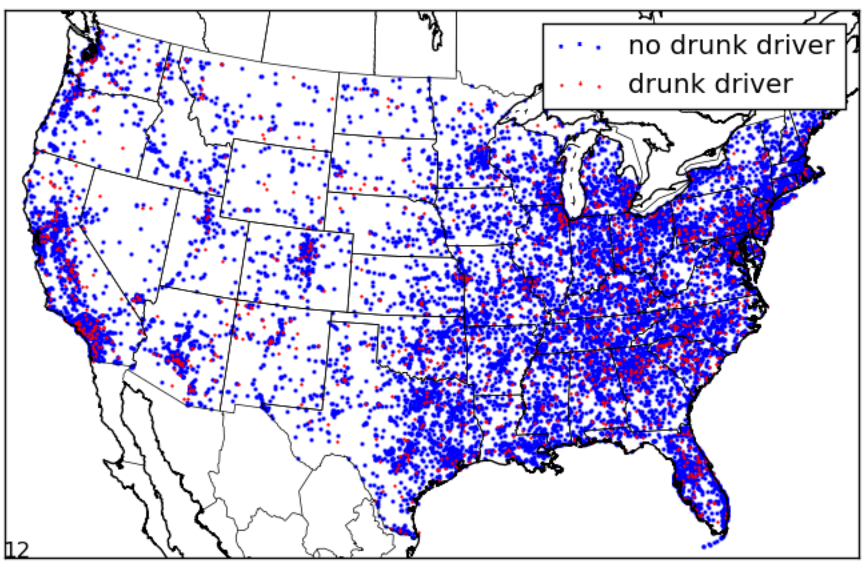
- **No class** on Wednesday 2/3! (I'm off to San Diego!)
- My office hours this week **cancelled**
- The GSIs/IA want your attention! Very few folks are attending office hours.

Note about Kaggle challenge grading

- Some parts of the Kaggle problems are open-ended. You can design your own features etc.
- 85% of the grade based on simply beating a basic benchmark (should not be too difficult)
- 15% of the grade based on performance on leaderboard

New MDST Challenge Released Soon!

- The Michigan Data Science Team (http://mdst.eecs.umich.edu) is announcing a new competition in a few days.
- Topic has to do with drunk driving accidents!



- Meeting: **Thursday Feb. 4, 5pm in 3150 DOW**

Outline

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

References

This lecture draws upon the following resources:

- **[MLAPP]** (https://mitpress.mit.edu/books/machine-learning-0) Murphy, Kevin. *Machine Learning: A Probabilistic Perspective*. 2012.
- **[PRML]** (http://www.springer.com/us/book/9780387310732) Bishop, Christopher. *Pattern Recognition and Machine Learning*. 2006.
- **[CS229]** Ng, Andrew. CS 229: Machine Learning (http://cs229.stanford.edu/). Autumn 2015.

Naive Bayes Classifiers

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

Review: Generative Classifiers

A **generative classifier** learns a joint model $P(y, x) = P(x|y)P(y)$.

- The *likelihood* specifies how to generate observed features x if labels y are known
- The *prior* encodes beliefs about popularity of each label

Classify using the **MAP Estimation**, via Bayes' Rule:

$$\hat{y} = \arg\max_y P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

Naive Bayes: Problem

We will use **Naive Bayes** to solve the following classification problem:

- Categorical features x
- Predict discrete class label y

Naive Bayes: Spam Classification

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_j, for $j \in \{1, 2, \ldots, D\}$, indentifies the j^{th} word

Naive Bayes: Independence Assumption

For simplicity, we assume all features are **conditionally independent** given the label,

$$P(x|y = c) = \prod_{k=1}^{D} P(x_j|y = c)$$

Features x_k modeled as categorical random variables, expressed differently between classes

$$x_j \mid y = c \;\sim\; \text{Categorical}(\theta_c)$$

- Same categorical distribution for each feature (not necessary)
- *each word has different probabilities across different contexts*!

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!

Naive Bayes: Is Independence Justified?

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

Naive Bayes: Is Independence Justified?

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"...

Naive Bayes: Full Model

The full generative model of **Naive Bayes** is:

$$y \sim \text{Categorical}(\pi)$$
$$x_j | y = c \sim \text{Categorical}(\theta_c) \quad \forall j = 1, \ldots, D$$

with parameters:

- Class-conditional probabilities $\theta = (\theta_1, \ldots, \theta_C)$

> $\theta_c \in \mathbb{R}^M$ parameterizes a dist. over vocab.

- Class priors $\pi = (\pi_1, \ldots, \pi_C) \in \mathbb{R}^C$

Naive Bayes: Parameter Estimation

Goal: Estimate class-conditional probabilities θ_c and class priors π_c.

- Given training data $\mathcal{D} = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N))$

We will discuss the **MLE** and **MAP** parameter estimates.

Naive Bayes: Maximum Likelihood

The probability for a single data case $(\mathbf{x}_i, y_i = c)$ is

$$P(\mathbf{x}_i, y_i = c) = P(y_i|\pi) \prod_{j=1}^{D} P(x_{ij}|\theta_c) = \pi_c \prod_{j=1}^{D} P(x_{ij}|\theta_c)$$

Therefore, the log-likelihood is

$$\log P(\mathbf{x}_i, y_i) = \log \pi_c + \sum_{j=1}^{D} \log P(x_{ij}|\theta_c)$$

Naive Bayes: Maximum Likelihood

The log-likelihood given all training data $\mathcal{D}$ is then

$$\log P(\mathcal{D}|\theta, \pi) = \sum_{c=1}^{C} N_c \log \pi_c + \sum_{j=1}^{D} \sum_{c=1}^{C} \sum_{i:y_i=c} \log P(x_{ij}|\theta_c)$$

where N_c is the occurrence count of class c in the training data $\mathcal{D}$

Naive Bayes: Maximum Likelihood

It is easy to check (**crucial exercise!**) that the maximum likelihood estimators are:

$$\hat{\pi}_c = \frac{N_c}{N} \quad \hat{\theta}_{cw} = \frac{N_{cw}}{N_c}$$

- N = number of examples in $\mathcal{D}$
- N_c = number of examples in class c in $\mathcal{D}$
- N_{cw} = number of examples in class c containing word w in $\mathcal{D}$

Naive Bayes: Sparse Features

Problem: When working with text, features are **sparse**:

1. In training, we only see a *small, small* fraction of words in the vocabulary
2. 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?
- What happens if that word never appears in testing? (*Black Swan Paradox*)

Naive Bayes: Priors

Solution: Place Dirichlet priors on π and θ_c to *smooth out* unknowns:

$$\begin{aligned}
\pi &\sim \text{Dirichlet}(\alpha_1, \ldots, \alpha_C) \\
\theta_c &\sim \text{Dirichlet}(\beta_1, \ldots, \beta_M) \quad \forall\, c = 1, \ldots, C \\
y &\sim \text{Categorical}(\pi) \\
x_j | y = c &\sim \text{Categorical}(\theta_c) \quad \forall\, j = 1, \ldots, D
\end{aligned}$$

> **Recall:** The Dirichlet defines a probability distribution over vectors with nonnegative entries summing to one, i.e. categorical distributions!

Naive Bayes: MAP Estimation

Exercise: Show that the MAP parameter estimates are

$$\hat{\pi}_c = \frac{N_c + \alpha_c}{N + \sum_{c'} \alpha_{c'}} \qquad \hat{\theta}_{cw} = \frac{N_{cw} + \beta_w}{N_c + \sum_{w'} \beta_{w'}}$$

The Dirichlet α and β parameters turn out to be **pseudocounts**!

- We assume we've seen α_c examples of each class
- and β_w examples of each word per class.

The choice $\alpha_c = \beta_w = 1$ is **Laplace Smoothing**

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?

Discriminant Functions

> Uses material from **[PRML]**

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**

Linear Discriminant Functions

A **linear discriminant function** $y(x) = w^T x + w_0$ divides two classes in feature space

- weight vector $w \in \mathbb{R}^D$
- bias $w_0 \in \mathbb{R}$

Assign x to C_1 if $y(x) \geq 0$ and to C_0 otherwise.

> See **[PRML]** section 4.1 for a discussion of multiclass problems.

Linear Discriminant Functions

We have a linear discriminant $y(x) = w^T x + w_0$

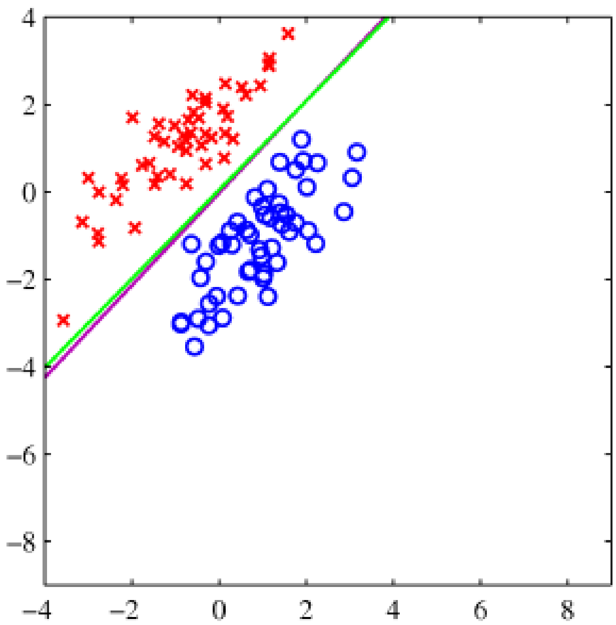
- In two dimensions, this is a line
- In three dimensions, a plane
- In general, a **separating hyperplane**!

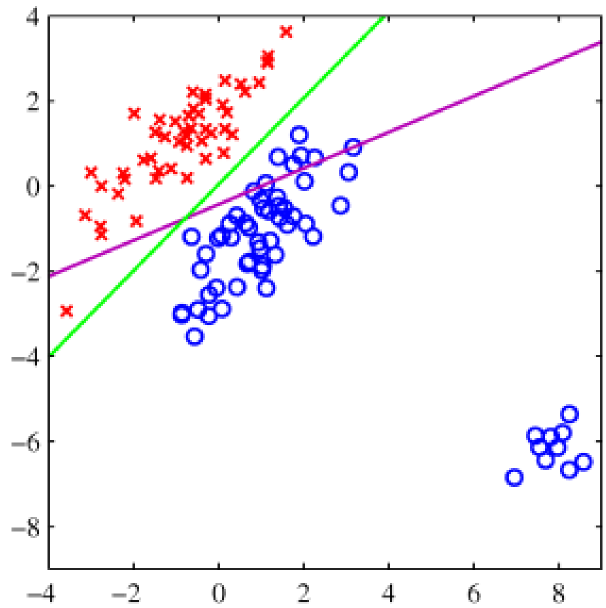
How to select weights w?

One approach: Least Squares for classification! Choose w that minimizes squared error

Bad Idea:

- Treat like a regression problem y vs. x
- Least squares is too sensitive to outliers. (*Why?*)





How to select weights w?

Better Idea: We'll cover the following models

- Gaussian Discriminant Analysis
  - Quadratic Discriminant Analysis
  - Linear Discriminant Analysis
- Fisher's Linear Discriminant
- Perceptron Learning Algorithm

Gaussian Discriminant Analysis

> Uses material from **[PRML]**, **[MLAPP]**, and **[CS229]**

Review: Multivariate Normal

A **normally distributed** random variable with mean $\mu \in \mathbb{R}^D$ and psd 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 $Z = (2\pi)^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}$. Compare to the one-dimensional case:

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

Review: Multivariate Normal
- when cov matrix is diagonal, we are sampling an independent normal for each coordinate - off-diagonal entries model *correlations* between variables

Gaussian Discriminant Analysis: Model

Generative probabilistic model

- Predict discrete label y from continuous features x
- Class-conditional densities are multivariate normals

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

with class priors π_c and per-class means μ_c and covariance matrices Σ_c

Gaussian Discriminant Analysis: Remark

- Unlike Naive Bayes, Gaussian Discriminant Analysis models **feature correlations**
- However, if all covariance matrices are diagonal, then GDA can be seen as the continuous analogue of Naive Bayes!

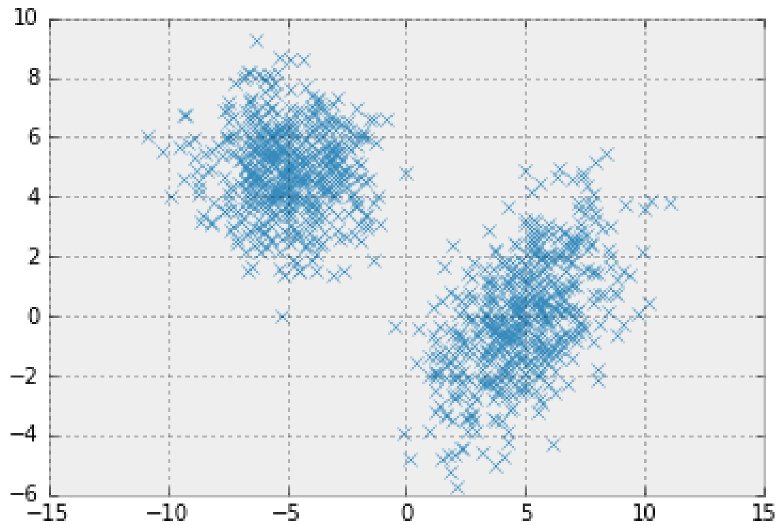
Gaussian Discriminant Analysis: Data

Because GDA is a *generative* model, we can *generate* fake data! (Assume uniform π)

In [11]:
```
def generate_gda(means, covs, num_samples):
    num_classes = len(means);
    num_samples //= num_classes;

    # cheat and draw equal number of samples from each gaussian
    samples = [
        np.random.multivariate_normal(means[c],covs[c],num_sample
s).T
        for c in range(num_classes)
    ];

    return np.concatenate(samples, axis=1);
```

Gaussian Discriminant Analysis: Data

In [12]:
```
# define gaussians
means = [ [-5,5], [5, 0] ];
covs = [ [[3, 0], [0, 2]], [[4, 2], [2, 4]] ];
# plot
x, y = generate_gda(means, covs, 1000);
plt.plot(x, y, 'x');
```



Gaussian Discriminant Analysis: Classification

Classify a feature vector x using the **posterior mode**:

$$\begin{aligned}\hat{y}(x) &= \arg\max_c \log P(y = c|x) = \arg\max_c \log P(x|y = c)P(y = c) \\ &= \arg\max_c \left[\log P(y = c|\pi) + \log P(x|\theta_c)\right]\end{aligned}$$

- The probability of 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**!

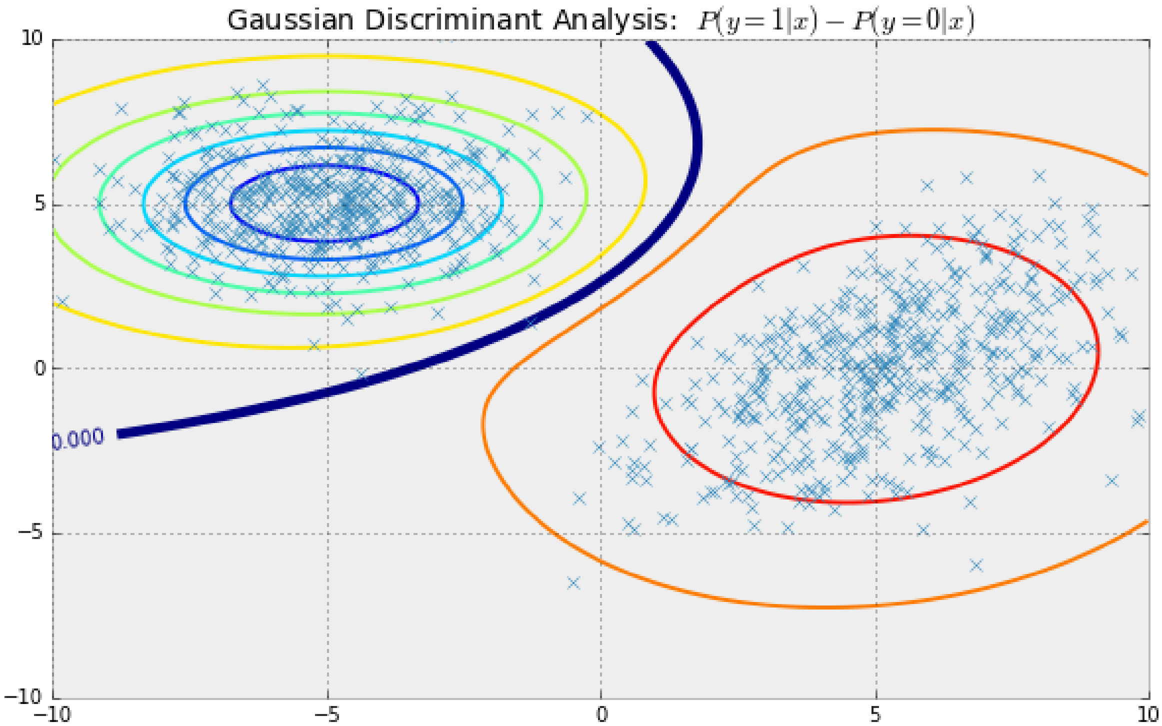
Gaussian Discriminant Analysis: Decision Boundary

In [18]:
```
# PLOTTING CODE!  (SKIP!)
def plot_decision_contours(means, covs):
    # plt
    fig = plt.figure(figsize=(10,6));
    ax = fig.gca();

    # generate samples
    data_x,data_y = generate_gda(means, covs, 1000);
    ax.plot(data_x, data_y, 'x');

    # dimensions
    min_x, max_x = -10,10;
    min_y, max_y = -10,10;

    # grid
    delta = 0.025
    x = np.arange(min_x, max_x, delta);
    y = np.arange(min_y, max_y, delta);
    X, Y = np.meshgrid(x, y);

    # bivariate difference of gaussians
    mu1,mu2 = means;
    sigma1, sigma2 = covs;
    Z1 = mlab.bivariate_normal(X, Y, sigmax=sigma1[0][0], sigmay=si
gma1[1][1], mux=mu1[0], muy=mu1[1], sigmaxy=sigma1[0][1]);
    Z2 = mlab.bivariate_normal(X, Y, sigmax=sigma2[0][0], sigmay=si
gma2[1][1], mux=mu2[0], muy=mu2[1], sigmaxy=sigma2[0][1]);
    Z = Z2 - Z1;

    # contour plot
    ax.contour(X, Y, Z, levels=np.linspace(np.min(Z),np.max(Z),1
0));
    cs = ax.contour(X, Y, Z, levels=[0], c="k", linewidths=5);
    plt.clabel(cs, fontsize=10, inline=1, fmt='%1.3f')

    # plot settings
    ax.set_xlim((min_x,max_x));
    ax.set_ylim((min_y,max_y));

    ax.set_title("Gaussian Discriminant Analysis:  $P(y=1 | x) -
P(y=0 | x)$")
```

Gaussian Discriminant Analysis: Decision Boundary

The decision boundary is **quadratic** in general, as seen from the thresholded class posterior.

In [19]:
```
# define gaussians
means = [ [-5,5], [5, 0] ];
covs = [ [[3, 0], [0, 2]], [[4, 2], [2, 4]] ];

plot_decision_contours(means, covs);
```



Linear Discriminant Analysis

Consider the case where covariances are **shared**, that is $\Sigma_c = \Sigma$. Recall that $P(y = c|x) \propto P(y = c)P(x|y = c)$, so we have

$$\begin{aligned} P(y = c|x) \propto \pi_c \exp\left[-\frac{1}{2}(x - \mu_c)^T \Sigma^{-1}(x - \mu_c)\right] \\ = \pi_c \exp\left[\mu_c^T \Sigma^{-1} x - \frac{1}{2} x^T \Sigma^{-1} x - \frac{1}{2}\mu_c^T \Sigma^{-1} \mu_c\right] \\ = \exp\left[\mu_c^T \Sigma^{-1} x - \frac{1}{2}\mu_c^T \Sigma^{-1} \mu_c + \log \pi_c\right] \\ \cdot \exp\left[-\frac{1}{2} x^T \Sigma^{-1} x\right] \end{aligned}$$

The rightmost term is independent of c, so will cancel out when we normalize.

Linear Discriminant Analysis

From the previous slide, we had

$$P(y=c|x) \propto \exp\left[\mu_c^T \Sigma^{-1} x - \frac{1}{2}\mu_c^T \Sigma^{-1} \mu_c + \log \pi_c\right]$$

Letting $\gamma_c = -\frac{1}{2}\mu_c^T \Sigma^{-1} \mu_c + \log \pi_c$ and $\beta_c = \Sigma^{-1} \mu_c$,

$$P(y=c|x) = \frac{\exp\left[\beta_c^T x + \gamma_c\right]}{\sum_c \exp\left[\beta_c^T x + \gamma_c\right]} = Softmax_c(\eta)$$

with $\eta = \left[\beta_c^T x + \gamma_c\right]_{c=1}^{C}$.

Linear Discriminant Analysis: Decision Boundary

For binary classification, $\mathrm{Softmax}(\eta)_\mathrm{c}$ becomes the sigmoid function.

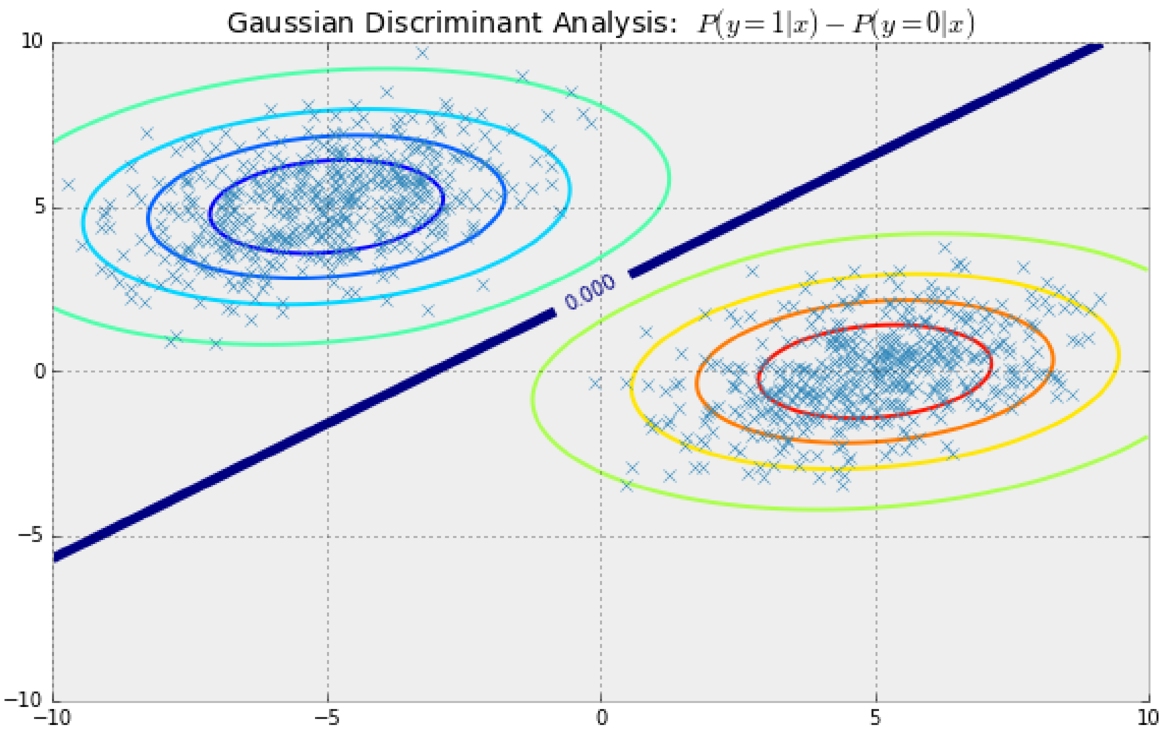
- As with logistic regression, taking logs yields a linear function of x.
- Thresholding gives a **linear decision boundary**

We conclude that Gaussian Discriminant Analysis with **shared covariances** yields a linear classifier.

- Different priors move the threshold up and down, just shifting the decision boundary.

Linear Discriminant Analysis: Decision Boundary

```
In [20]: # define gaussians
means = [ [-5,5], [5, 0] ];
covs = [ [[3, 1], [1, 2]],
         [[3, 1], [1, 2]] ];

plot_decision_contours(means, covs);
```



Gaussian Discriminant Analysis: MLE

Maximum Likelihood estimation would proceed similarly to Naive Bayes,

- Fit a per-class Gaussian by estimating the mean and covariance of examples within each class

$$\hat{\mu}_c = \frac{1}{N_c} \sum_{i:y_i=c} x_i \qquad \hat{\Sigma}_c = \frac{1}{N_c} \sum_{i:y_i=c} (x_i - \hat{\mu}_c)(x_i - \hat{\mu}_c)^T$$

> Left as an exercise!

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)

> Uses material from **[PRML]**

Fisher's Linear Discriminant

Use $\mathbf{w}$ to project x onto one dimension

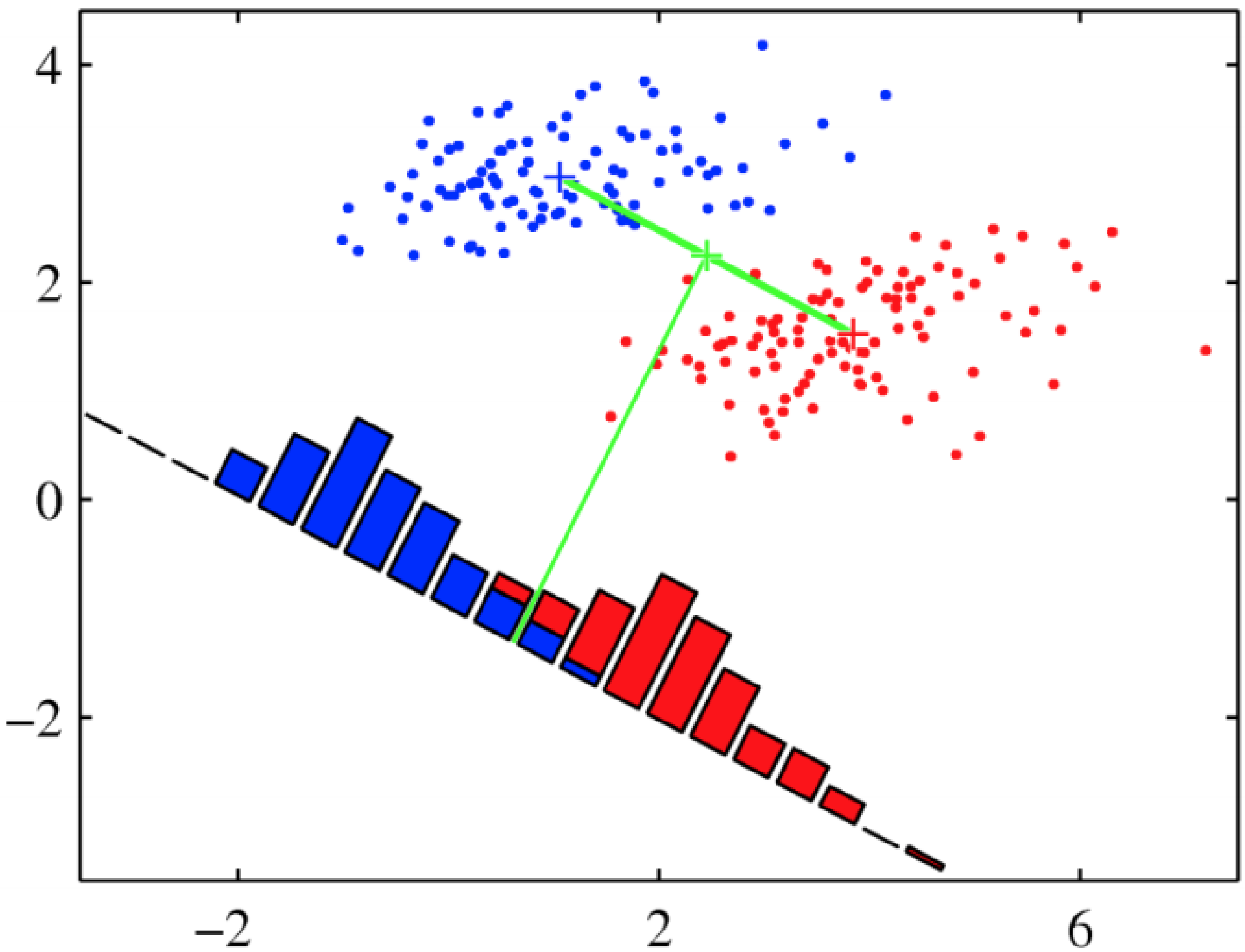
- If $\mathbf{w}^T x \geq -w_0$ then assign $\mathbf{x}$ to class 1, else to class 0.

Select a projection w that best "separates" the classes, i.e., both

- Maximizes class separation (distance between means)
- Minimizes class variance

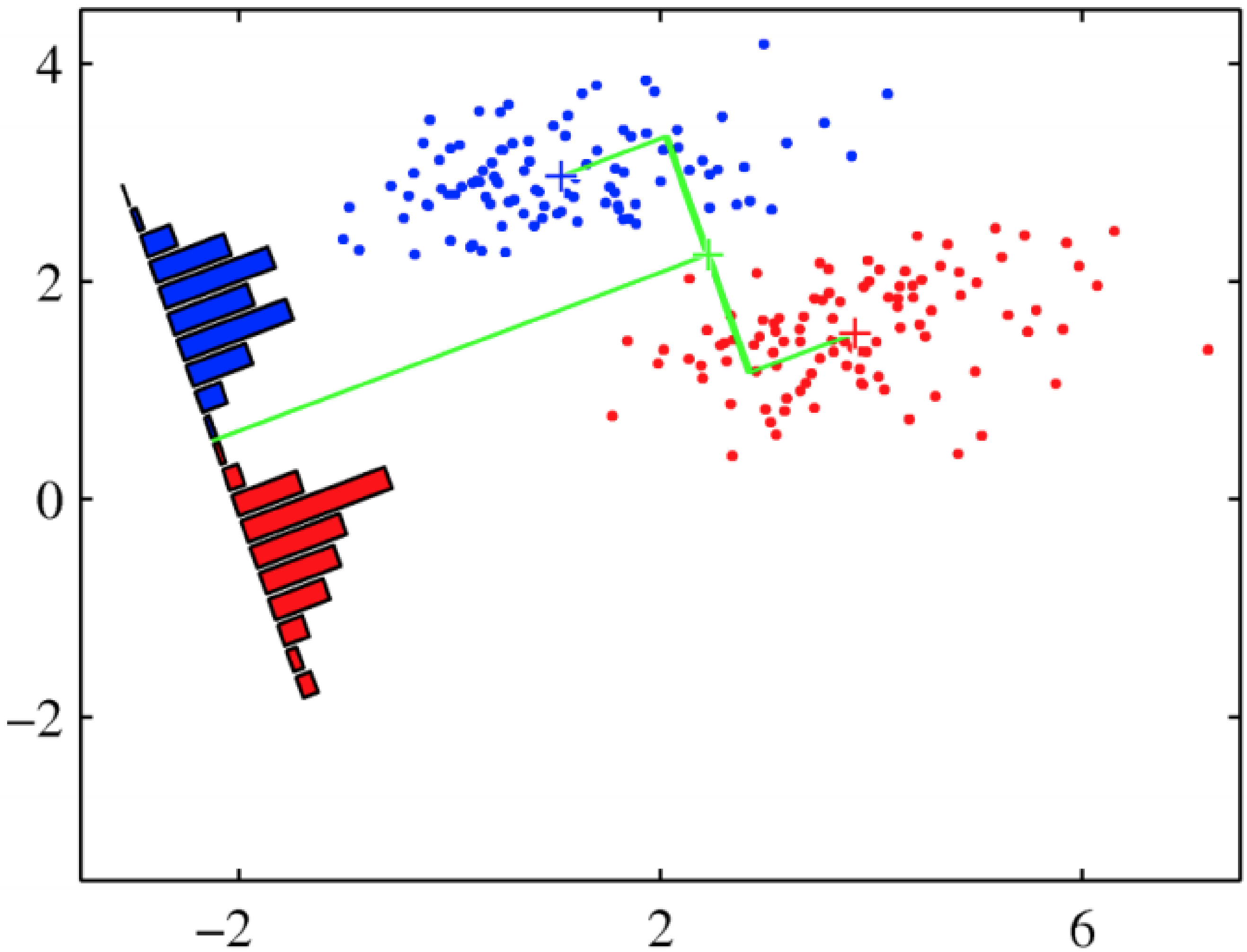
Fisher's Linear Discriminant

Maximizing separation alone:



Fisher's Linear Discriminant

Maximizing both inter-class separation and minimizing in-class variance:



Fisher's Linear Discriminant: Objective

Goal 1: Maximize the *distance between classes*

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

$$\mathbf{m}_k = \frac{1}{N_k} \sum_{i:y_i=c} x_i$$

In particular, maximize distance between projected means

Fisher's Linear Discriminant: Objective

Goal 2: Minimize the *variance within classes*

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

Fisher's Linear Discriminant: Objective

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

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

Fisher's Linear Discriminant: Objective

Using $(m_1 - m_0) = \mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_0)$ we can rewrite the numerator:

$$\begin{aligned} ||m_1 - m_0||^2 &= \mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_0)(\mathbf{m}_1 - \mathbf{m}_0)^T \mathbf{w} \\ &= \mathbf{w}^T S_B \mathbf{w} \end{aligned}$$

Where S_B is the **between-class scatter matrix**

- 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

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

Fisher's Linear Discriminant: Objective

We can know rewrite the objective explicitly in terms of w,

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

See **[PRML]** for a derivation of the solution $\mathbf{w} = S_W^{-1}(\mathbf{m}_1 - \mathbf{m}_0)$