Final Review CS195F
Part 1

Dae Il Kim

Topics Covered

Generative vs. Discriminative Models

Bayesian Decision Theory

Linear vs. Logistic Regression

Kernels & Gaussian Processes

Clustering & K-means Algorithms

Some figures & slides taken from Sudderth's Lecture Notes & Murphy's ML Book

Generative vs. Discriminative Models

- What's the difference?

Model Parameters

Data

Generative Models

- *Training:* Learn prior and likelihood: $p(y \mid \theta), p(x \mid y, \theta)$
- *Test:* Posterior from Bayes' rule:

$$p(y \mid x) \propto p(y \mid \theta) p(x \mid y, \theta)$$

Input Features

Discriminative or Conditional Models

- *Training:* Learn posterior: $p(y \mid x, \theta)$
- *Test:* Apply posterior:
- *Con:* Easier to incorporate domain knowledge generatively
- *Con:* Cannot handle missing features, no model of $p(x)$
- *Pro:* No need to design an accurate model of $p(x)$

Types of Generative/Discriminative Models

Examples of Generative Models

- Naïve Bayes
- Gaussian Discriminant Analysis
- Linear Discriminant Analysis

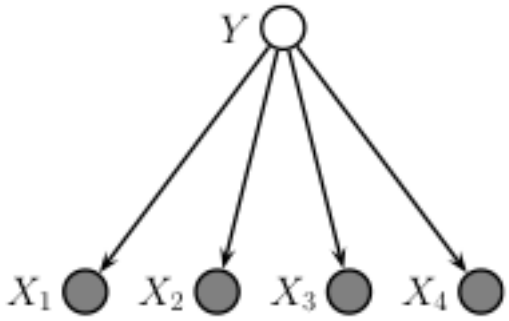
Examples of Discriminative Models

- Linear Regression
- Bayesian Linear Regression
- Logistic Regression
- Gaussian Processes

Naïve Bayes

- The simplest of our generative classification models where input features (X) are assumed to be independent for a given class (Y = c).

Naïve Bayes
Graphical Model



(i.e. Spam, Not Spam)

Figure from Murphy (2012)

Input Features (i.e. word counts of Viagra)

$$p(y = c|\mathbf{x}, \boldsymbol{\theta}) = \frac{p(y = c|\boldsymbol{\theta})p(\mathbf{x}|y = c, \boldsymbol{\theta})}{\sum_{c'} p(y = c'|\boldsymbol{\theta})p(\mathbf{x}|y = c', \boldsymbol{\theta})}$$

Generative since we are modeling the generative process for our features X

Placing a non-uniform prior on y allows us to do MAP estimation (e.g. Dirichlet / Multinomial)

Gaussian Discriminant Analysis

Our features are continuous and we place a Gaussian prior on our class conditional densities

Prior Term

$$p(y = c \mid \pi) = Cat(y = c \mid \pi)$$

Likelihood Term

$$p(x \mid y = c, \theta) = N(x \mid \mu_c, \Sigma_c)$$

Deriving the posterior using Bayes Rule

$$p(y = c \mid \pi, x, \theta) = \frac{p(y{=}c|\pi)p(x|y{=}c,\theta)}{\sum_{c\prime=1}^{C} p(y{=}c\prime|\pi)p(x|y{=}c\prime,\theta)}$$

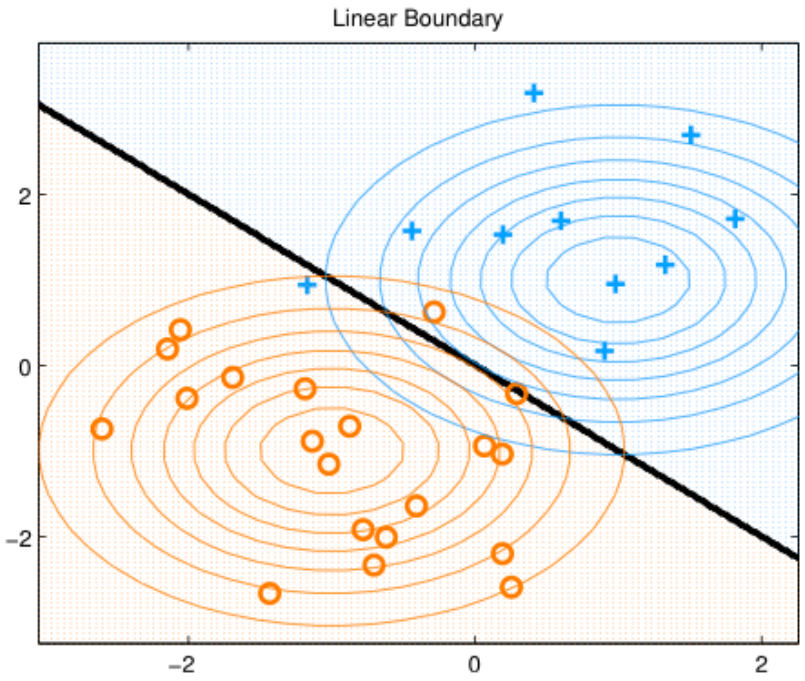
Note: This model becomes Naïve Bayes when the class covariances are diagonal.

$$p(y = c|\mathbf{x}, \boldsymbol{\theta}) = \frac{\pi_c |2\pi\boldsymbol{\Sigma}_c|^{-\frac{1}{2}} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_c)^T \boldsymbol{\Sigma}_c^{-1}(\mathbf{x} - \boldsymbol{\mu}_c)\right]}{\sum_{c'} \pi_{c'} |2\pi\boldsymbol{\Sigma}_{c'}|^{-\frac{1}{2}} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_{c'})^T \boldsymbol{\Sigma}_{c'}^{-1}(\mathbf{x} - \boldsymbol{\mu}_{c'})\right]}$$

Linear Discriminant Analysis

When our class conditional covariance parameters are all equal so that: $\Sigma_c = \Sigma$

The boundaries all become linear functions, thus the term ***Linear Discriminant Analysis***



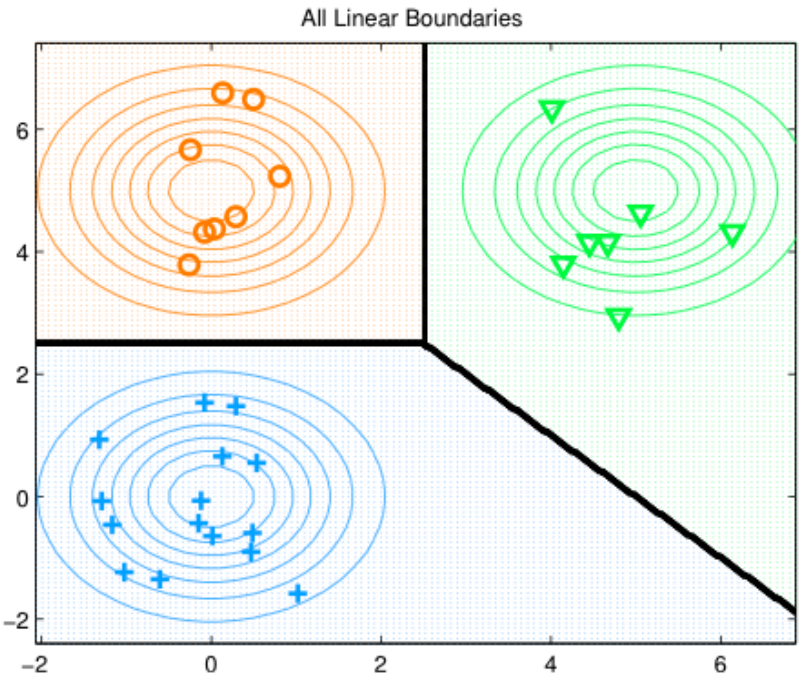


Figure from Murphy (2012)

Note: This was the key in solving the midterm exam question 3.

Linear Regression / Bayes Linear Regression

Offset/Bias Term

Feature Weights (Learned)

Input/Covariates

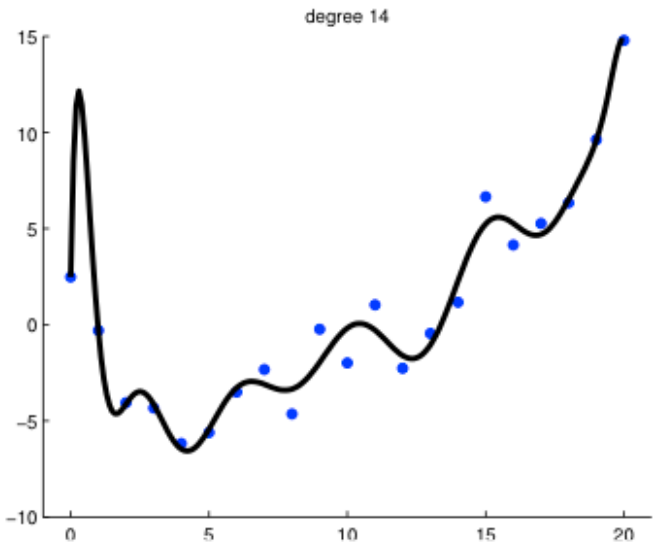
Directly maximize Y assuming it is Gaussian

$$p(y|\mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}(y|w_0 + \mathbf{w}^T\mathbf{x}, \sigma^2)$$

Basis Function

Incorporate a basis function to better model Y. Model is still linear since w is linear.

$$p(y|\mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}(y|\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}), \sigma^2)$$



$$\phi(x) = [1, x, x^2, \ldots, x^d]$$

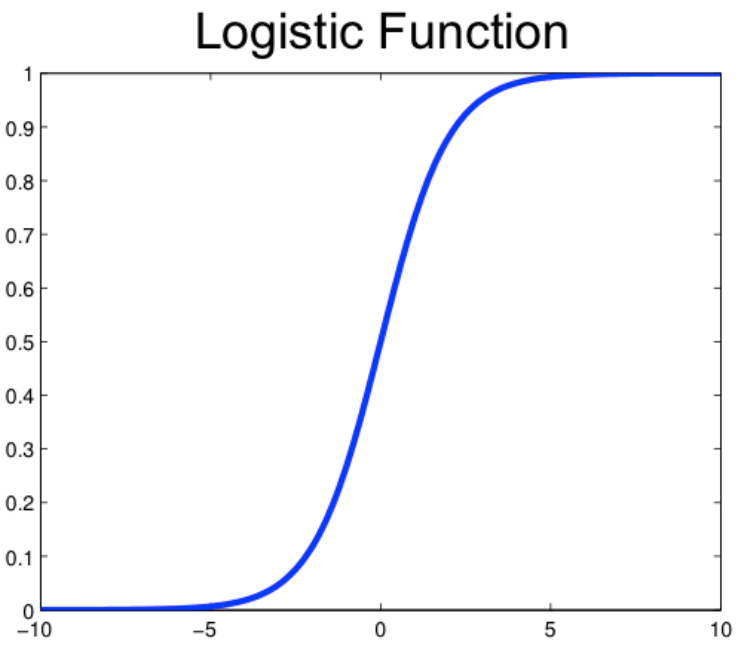
Basis functions need to be carefully chosen to avoid overfitting. Here we have a basis function of polynomial degree 14.

Placing a Gaussian prior on w results in a model called ridge regression and leads to MAP estimation.

Logistic Regression

Recall the Logistic Function

$$\sigma(w^T x) = \frac{1}{1+exp(-w^T x)}$$



Returns a value between 0 and 1

Binary Logistic Regression

$$p(y \mid x, w) = \text{Bern}(y \mid \sigma(w^T x))$$

Multi-Class Logistic Regression

$$p(y \mid x, W) = \text{Cat}(y \mid S(W^T x))$$

Where S is the softmax function

$$S_c(W^T x) = \frac{\exp(w_c^T x)}{\sum_k \exp(w_k^T x)}$$

Learning our weights requires us to solve a complex convex optimization problem.

Bayes Decision Theory

The optimal action in Bayes Decision theory is minimizing the posterior expected loss

$$\rho(a|\mathbf{x}) \triangleq \mathbb{E}_{p(y|\mathbf{x})}\left[L(y,a)\right] = \sum_y L(y,a)p(y|\mathbf{x})$$

The sum becomes an integral when y is continuous

What kind of a decision minimizes this loss function? MAP Estimate

$$L(y,a) = \mathbb{I}(y \neq a) = \begin{cases} 0 & \text{if } a = y \\ 1 & \text{if } a \neq y \end{cases}$$

Predicted class labels

	$\hat{y}=1$	$\hat{y}=0$
$y=1$	0	1
$y=0$	1	0

Loss (FN)

Loss (FP)

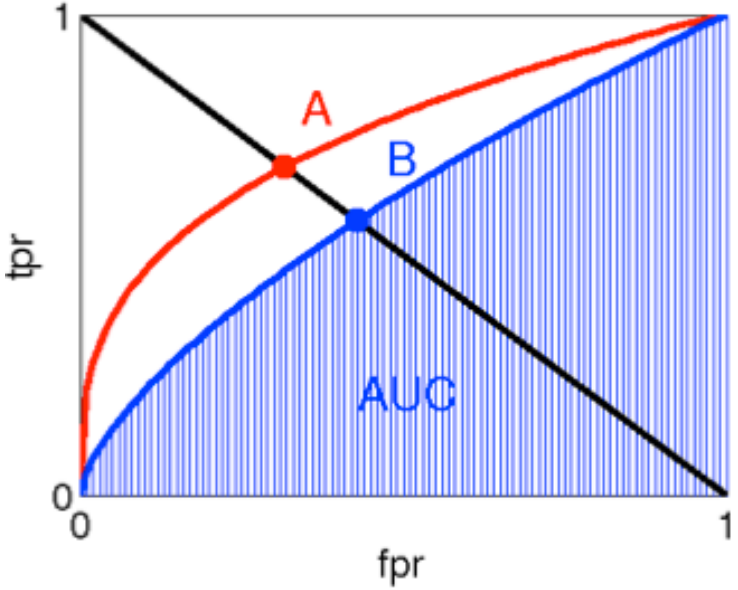
True class labels

Consider that not all losses are equal (e.g. cancer tests)

The loss function can also be written in a matrix format if y can take K discrete classes.

Bayes Decision Theory

ROC Curves (Binary Classification)

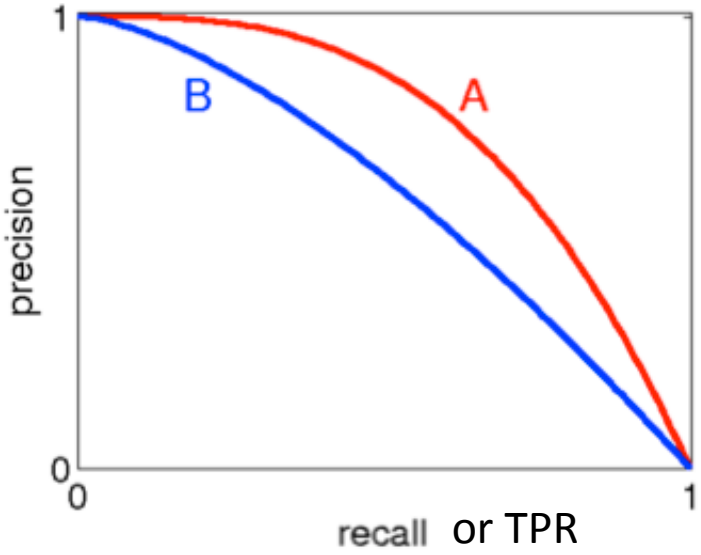


Useful when there are roughly a similar number of positives and negatives (e.g. link prediction in a binary graph)

TPR = TP / P (total positives)

FPR = FP / N (total negatives)

Precision Recall Curves



or TPR

Useful when trying to detect a rare event or there are a high number of negatives (e.g. object recognition)

Recall:

$$\frac{TP}{N_+} \approx p(\hat{y} = 1 \mid y = 1)$$

TP / (TP + FN)

Precision:

$$\frac{TP}{\hat{N}_+} \approx p(y = 1 \mid \hat{y} = 1)$$

TP / (TP + FP)

Surrogate Loss Functions

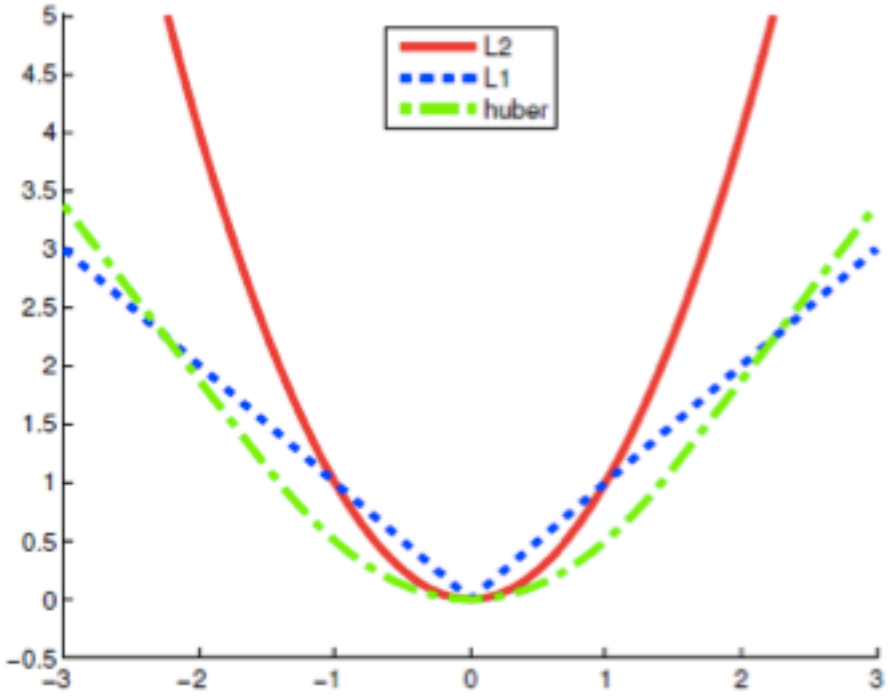
What kind of loss is L2 loss?

Differentiable? (YES)
Provides Sparisty? (NO)

What kind of loss is L1 loss?

Differentiable? (NO)
Provides Sparisty? (YES)

Note, previous lecture assumed L1 and L2 loss with logistic and linear regression respectively. This is incorrect, either can be performed for both.



What kind of loss is Huber loss?

A compromise between the two. Provides L2 regularization for errors smaller than delta and provides L1 regularization for errors larger than delta.

Advantages? Robust to outliers and is differentiable everywhere

$$p(y|\mathbf{x}, \mathbf{w}, b) \quad = \quad \text{Lap}(y|\mathbf{w}^T\mathbf{x}, b) \propto \exp(-\frac{1}{b}|y - \mathbf{w}^T\mathbf{x}|)$$

Kernels & Mercer's Theorem

A Kernel function takes two inputs (x1, x2) and maps this to some real value that denotes the similarity between these inputs.

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R} \qquad k(x_i, x_j) = k(x_j, x_i)$$

Intuition: Larger values indicate inputs are "more similar"

Often symmetric and non-negative, but not necessary constraints

Mercer's Theorem: ANY positive semidefinite kernel can be written as:

$$k(x_i, x_j) = \sum_{\ell=1}^{d} \phi_\ell(x_i)\phi_\ell(x_j)$$

for some feature mapping ϕ
(but may need $d \to \infty$)

Kernelizing a Learning Algorithm

Start with any learning algorithm based on features $\phi(x)$

(Don't worry that computing features might be expensive or impossible.)

Manipulate steps in algorithm so that it depends not directly on features, but only their inner products: $k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$

Gaussian Processes

We can rewrite our parametric form for linear regression via kernels:

$$f(x) = w^T \phi(x) \qquad \phi(x) \in \mathbb{R}^{m \times 1}$$

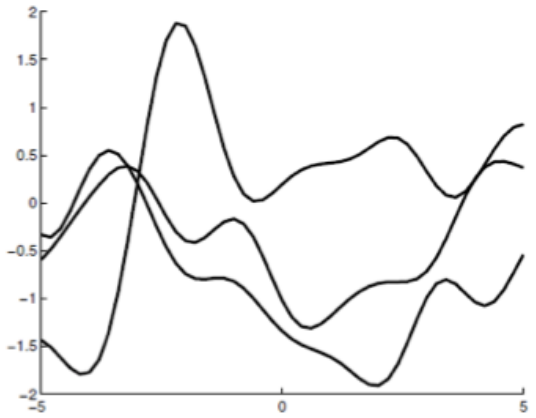
Assume a Gaussian prior on our weights

$$p(w) = \mathcal{N}(w \mid 0, \alpha^{-1} I_m) \qquad w \in \mathbb{R}^{m \times 1}$$

The joint predictive distribution results in a Gaussian defined via a Kernel function

$$p(f) = \mathcal{N}(f \mid 0, \alpha^{-1} \Phi \Phi^T) = \mathcal{N}(f \mid 0, K)$$

This results in a Gaussian process, a distribution over functions where any finite subset of these functions are jointly Gaussian distributed. (no need to learn weights w)



Gaussian Process Regression

To perform prediction, we understand that by definition of the GP, the joint distribution has the following form:

$$\begin{pmatrix} \mathbf{f} \\ \mathbf{f}_* \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu}_* \end{pmatrix}, \begin{pmatrix} \mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_*^T & \mathbf{K}_{**} \end{pmatrix} \right)$$

Pick your favorite Kernel!

a) Squared Exponential
b) Radial Basis Kernel
c) Polynomial Kernel

To make a prediction, we need to apply standard Gaussian formulas for the posterior mean and covariance (assuming noise-free observations y = f(x)):

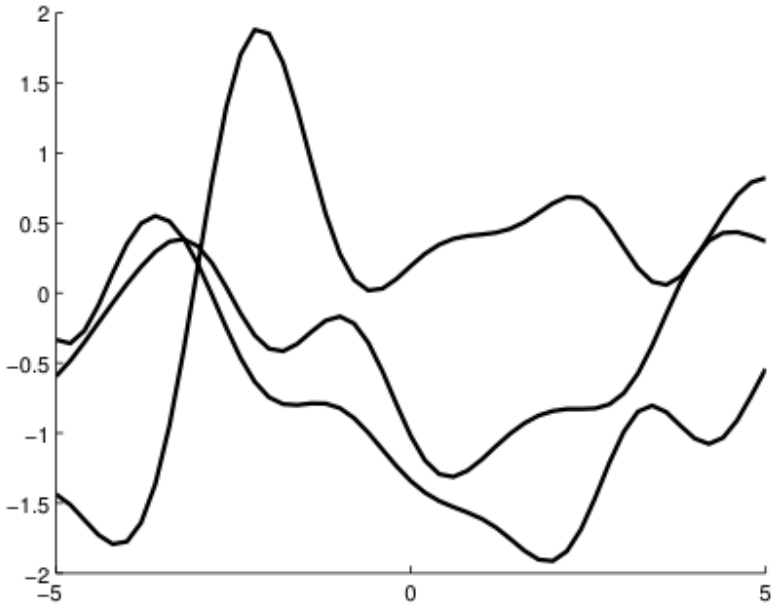
$$\begin{aligned} p(\mathbf{f}_*|\mathbf{X}_*, \mathbf{X}, \mathbf{f}) &= \mathcal{N}(\mathbf{f}_*|\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*) \\ \boldsymbol{\mu}_* &= \boldsymbol{\mu}(\mathbf{X}_*) + \mathbf{K}_*^T \mathbf{K}^{-1}(\mathbf{f} - \boldsymbol{\mu}(\mathbf{X})) \\ \boldsymbol{\Sigma}_* &= \mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{K}_* \end{aligned}$$

$$\mathbf{K}_* = \kappa(\mathbf{X}, \mathbf{X}_*)$$

N x D Test Data

$$\mathbf{K}_{**} = \kappa(\mathbf{X}_*, \mathbf{X}_*)$$

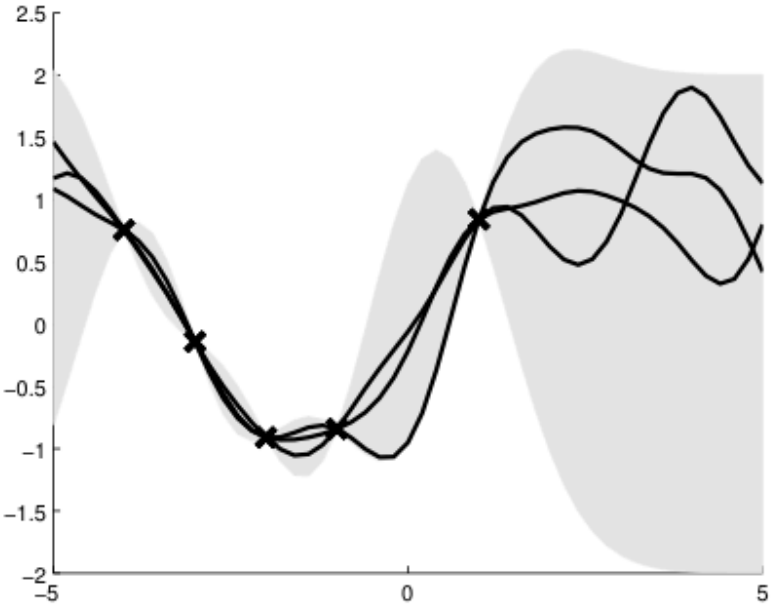
Gaussian Process Regression



Samples from Prior

$$\kappa(x, x') = \sigma_f^2 \exp(-\frac{1}{2\ell^2}(x - x')^2)$$

Squared exponential kernel



Posterior Given 5 Noise-Free Observations

Note: Learning hyperparameters within the Kernel require fancier learning techniques (i.e. discrete grid search)

Gaussian Processes

GP Classification is similar to Logistic Regression, but uses kernels rather than features

$$p(f) = \mathcal{N}(f \mid 0, K)$$

$$p(y_i \mid x_i, f_i) = \mathrm{Ber}(y_i \mid \mathrm{sigm}(f_i))$$

Again, no closed form form the posterior, have to use convex optimization tools

Final note on GPs: Kernels or Features? Really depends on your dataset…

$N \longrightarrow$ number of training examples

$M \longrightarrow$ number of features

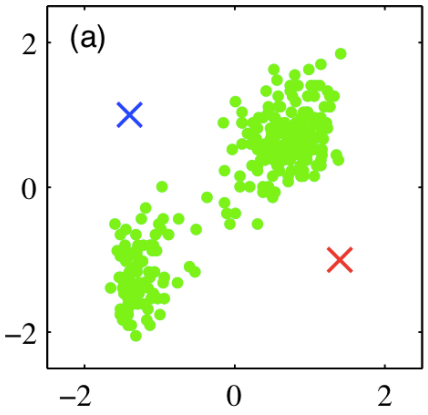
$L \longrightarrow$ cost of kernel function evaluation, at worst $\mathcal{O}(M)$

$\Phi \longrightarrow$ *NxM* matrix evaluating each feature for all training data

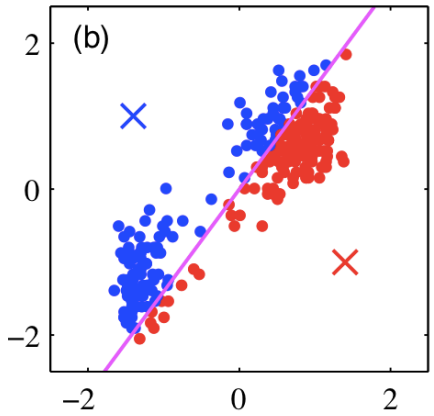
- Feature-based linear regression: $\mathcal{O}(NM^2 + M^3)$
- Kernel-based GP regression: $\mathcal{O}(LN^2 + N^3)$

K-Means and Clustering

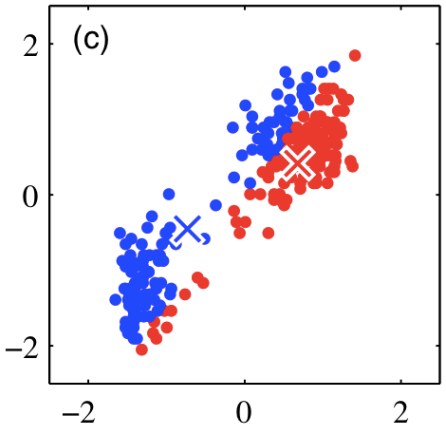
One of the simplest clustering algorithms out there, makes hard cluster assignments to data points. Also called hard EM (covered more in part 2)



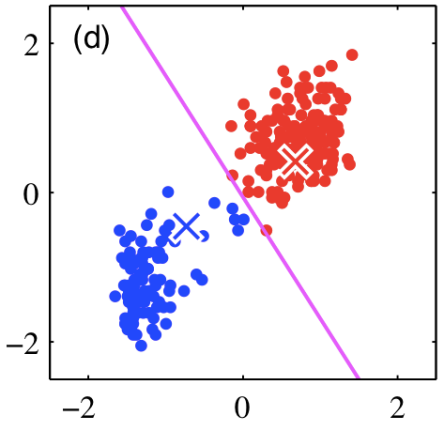
Pick some random centroids



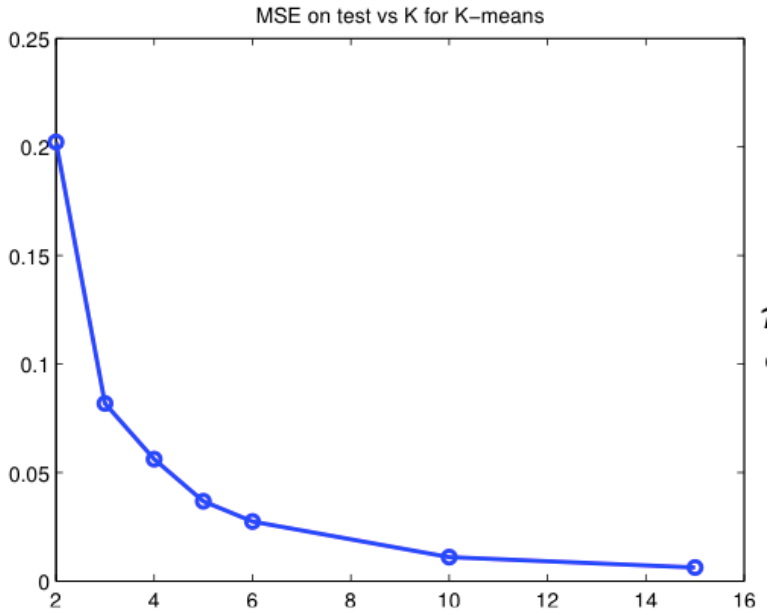
Calculate Euclidean Distances to centroids. Make assignments.



Recalculate centroids



Make new assignments & repeat



Test error, more K lowers MSE, but can overfit the data. MSE not the best metric for K-means to determine model selection.

Bishop et. al 2006

Summary

20,000 Feet Overview:

When you're interested in predicting the output of your data and your training data has labels, we often turn to supervised learning techniques.

Many of these techniques can be split into either generative or discriminative models. Generative models assume the generative process for our features while discriminative models work directly on maximizing our likelihood.

Choosing a good model requires understanding the inner workings of the model's behavior. An overly complicated model does not guarantee better performance.

Unsupervised learning is much harder (part 2) and inference/learning as a result becomes considerably more difficult.

Congratulate yourself if you feel you have some mastery over this material. This is a hard subject, but one well worth learning!