Module 5: Classification

Linear Methods: Logistic Regression

STAT/BIOSTAT 527, University of Washington
Emily Fox
May 28th, 2013

Very convenient!

$$p(y=0 \mid x,\beta) = \frac{1}{1+\exp(\beta_0 + \sum_j \beta_j x_j)}$$

implies

$$p(y=1 \mid x,\beta) = \frac{\exp(\beta_0 + \sum_j \beta_j x_j)}{1+\exp(\beta_0 + \sum_j \beta_j x_j)}$$

$1 - P(y=0 \mid x)$

Examine ratio:

$$\frac{p(y=1 \mid x,\beta)}{p(y=0 \mid x,\beta)} = \exp(\beta_0 + \sum_j \beta_j x_j) > 1$$

$\Rightarrow$ class 1 wins, else class 0 (under 0-1 loss)

linear classification rule!

implies

log odds

$$\log \frac{p(y=1 \mid x,\beta)}{p(y=0 \mid x,\beta)} = \beta_0 + \sum_j \beta_j x_j > 0$$

linear

$\Rightarrow$ class 1 wins as before

Maximizing Conditional Log Likelihood

$$p(y=0 \mid x,\beta) = \frac{1}{1+\exp(\beta_0+\sum_j \beta_j x_j)}$$

$$p(y=1 \mid x,\beta) = \frac{\exp(\beta_0+\sum_j \beta_j x_j)}{1+\exp(\beta_0+\sum_j \beta_j x_j)}$$

$$l(\beta) = \sum_i \log p(y_i \mid x_i, \beta)$$

$$= \sum_i y_i(\beta_0 + \sum_j \beta_j x_{ij}) - \log(1+\exp(\beta_0 + \sum_j \beta_j x_{ij}))$$

fixed in training data

$x \in \mathbb{R}^d$

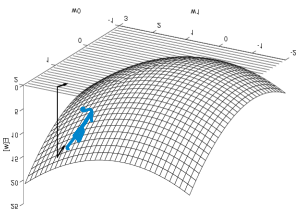
Good news: *l*(**β**) is concave function of **β**, no local optima problems

Bad news: no closed-form solution to maximize *l*(**β**)

Good news: concave functions easy to optimize

Optimizing Concave Function – Gradient Ascent

- Conditional likelihood for logistic regression is concave
- Find optimum with gradient ascent



Gradient: $\nabla_\beta l(\beta) = \left[\frac{\partial l(\beta)}{\partial \beta_0}, \ldots, \frac{\partial l(\beta)}{\partial \beta_d}\right]'$

Step size, η>0

Update rule: $\Delta\beta = \eta \nabla_\beta l(\beta)$

$$\beta_j^{(t+1)} \leftarrow \beta_j^{(t)} + \eta \frac{\partial l(\beta)}{\partial \beta_j}$$

new est. prev. est. stepsize

- Gradient ascent is simplest of optimization approaches
  - e.g., Conjugate gradient ascent can be much better

Often, esp. proofs, η gets smaller w/ iterations

e.g. $\eta_t = \frac{\alpha}{t}$ ← const.

Gradient Ascent for LR

start w/ $\beta^{(0)}$ (e.g. 0)

revisit soon

Gradient ascent algorithm: iterate until change < ε

$$\beta_0^{(t+1)} \leftarrow \beta_0^{(t)} + \eta \sum_i \left(y_i - \hat{p}(y = 1 \mid x_i, \beta^{(t)}) \right)$$

For j=1,…,d,

$$\beta_j^{(t+1)} \leftarrow \beta_j^{(t)} + \eta \sum_i x_{ij} \left(y_i - \hat{p}(y = 1 \mid x_i, \beta^{(t)}) \right)$$

repeat

can do in || $\forall j$

Linear Separability

$\exists \beta$ s.t. all pos. examples have $\beta_0 + \sum \beta_j x_j > 0$ + neg. examples have $\beta_0 + \sum \beta_j x_j < 0$

$\beta_0 + \sum_j \beta_j x_j = 0$

$2\beta_0 + \sum_j 2\beta_j x_j = 0$

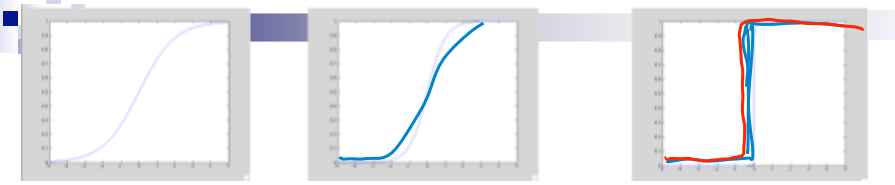
$\beta_0 + \sum_j \beta_j x_j > 0$

$2\beta_0 + \sum_j 2\beta_j x_j > 0$

"more sure" for no reason ... same separation

$\Rightarrow \alpha\beta$ is also a separating hyperplane $\forall \alpha > 0$

Large Parameters → Overfitting



$$\frac{1}{1+e^{-x}} \qquad \frac{1}{1+e^{-2x}} \qquad \frac{1}{1+e^{-100x}}$$

- If data is linearly separable, weights go to infinity

$$p(y=0 \mid \beta, x) = \frac{1}{1+e^{\beta_0+\sum_j \beta_j x_j}}$$ increases as $\|\beta\| \to \infty$

  - In general, leads to overfitting:
- Penalizing high weights can prevent overfitting…

regularization → $\|\beta\|_2^2$ ☆, $\|\beta\|_1$, ⋮

Standard v. Regularized Updates

- Maximum conditional likelihood estimate

$$\hat{\beta} = \arg\max_{\beta} \log \prod_{i=1}^{n} p(y_i \mid x_i, \beta)$$ ← w/o regularization

$$\beta_j^{(t+1)} \leftarrow \beta_j^{(t)} + \eta \sum_i x_{ij} \left(y_i - \hat{p}(y=1 \mid x_i, \beta^{(t)}) \right)$$

- Regularized maximum conditional likelihood estimate

$$\hat{\beta} = \arg\max_{\beta} \log \prod_{i=1}^{n} p(y_i \mid x_i, \beta) - \frac{\lambda}{2} \sum_{j=1}^{d} \beta_j^2$$

pushes toward 0

$$\beta_j^{(t+1)} \leftarrow \beta_j^{(t)} + \eta \left\{ -\lambda \beta_j^{(t)} + \sum_i x_{ij} \left(y_i - \hat{p}(y=1 \mid x_i, \beta^{(t)}) \right) \right\}$$

The Cost, The Cost!!! Think about the cost… in terms of n, d

- What's the cost of a gradient update step for LR???

$$\beta_j^{(t+1)} \leftarrow \beta_j^{(t)} + \eta \left\{ -\lambda \beta_j^{(t)} + \sum_i x_{ij} \left(y_i - \hat{p}(y = 1 \mid x_i, \beta^{(t)}) \right) \right\}$$

for each j

$O(d)$

$O(nd)$

$\beta^{(t)}, \beta_j^{(t)}, \beta_j^{(1)}$

Naively, $O(nd^2)$

but cache $\hat{p}$ (same $\forall j$) $\rightarrow O(nd)$

However, if n is huge (or streaming), this is very slow (infeasible) per little gradient step

Gradient ascent in Terms of Expectations

- "True" objective function:

$$l(\beta) = E_x[l(\beta, x)] = \int p(x) l(\beta, x) dx$$

expected loss

We have been min training loss s.t. complexity penalty, but we really want

- Taking the gradient:

$$\nabla_\beta l(\beta) = E_x[\nabla_\beta l(\beta, x)]$$

- "True" gradient ascent rule:

$$\beta^{(t+1)} \leftarrow \beta^{(t)} + \eta \, E_x[\nabla_\beta l(\beta, x)]$$

- How do we estimate expected gradient?

can't compute this, so approx w/ sample data

SGD: Stochastic Gradient Ascent (or Descent)

- "True" gradient: $\nabla l(\beta) = E_x[\nabla l(\beta, x)]$
- Sample based approximation: take x_i iid

$$E_x[\nabla_\beta l(\beta, x)] \approx \frac{1}{n} \sum_{i=1}^{n} \nabla_\beta l(\beta, x_i) \stackrel{\Delta}{=} \tilde{\nabla}_\beta l(\beta)$$

- What if we estimate gradient with just one sample???
  - Unbiased estimate of gradient $E[\tilde{\nabla}_\beta l(\beta)] = \nabla_\beta l(\beta)$
  - Very noisy! high var.
  - Called stochastic gradient ascent (or descent)
    - Among many other names
  - VERY useful in practice!!!

Stochastic Gradient Ascent for Logistic Regression

- Logistic loss as a stochastic function:

$$E_x[l(\beta, x)] = E_x\left[\log p(y \mid x, \beta) - \frac{\lambda}{2}||\beta||_2^2\right]$$

- Batch gradient ascent updates:

$$\beta_j^{(t+1)} \leftarrow \beta_j^{(t)} + \eta\left\{-\lambda\beta_j^{(t)} + \frac{1}{n}\sum_{i=1}^{n} x_{ij}\left(y_i - \hat{p}(y = 1 \mid x_i, \beta^{(t)})\right)\right\}$$

redefine by $\frac{1}{n}$

- Stochastic gradient ascent updates:
  - Online setting: take one data point at iter t: $x_{i(t)}$

$$\beta_j^{(t+1)} \leftarrow \beta_j^{(t)} + \eta\left\{-\lambda\beta_j^{(t)} + x_{i(t),j}\left(y_{i(t)} - \hat{p}(y = 1 \mid x_{i(t)}, \beta^{(t)})\right)\right\}$$

use same data pt $x_{i(t)}$ $\forall j$

What you should know…

- Classification: predict discrete classes rather than real values
- Logistic regression model: Linear model
  - Logistic function maps real values to [0,1]
- Optimize conditional likelihood
- Gradient computation
- Overfitting
- Regularization
- Regularized optimization
- Cost of gradient step is high, use stochastic gradient descent

Module 5: Classification

Linear Methods: LDA and QDA

STAT/BIOSTAT 527, University of Washington

Emily Fox

May 28th, 2013

Discriminative vs. Generative

- So far, we have considered modeling/fitting

$$p(Y \mid X)$$

class output ← input data (predictors)

"discriminative" method
Fixed X, max $p(Y|X)$ for fitting

- There are also a large set of ***generative*** methods
- Model:
  - Class-conditional densities $f_k(X) = p(X \mid Y=k)$
  - Class prior probabilities π_k ← $p(Y=k)$ w/ $\sum_{k=1}^{K} \pi_k = 1$
- Via Bayes' rule:

$$p(Y=k \mid X=x) = \frac{\pi_k f_k(x)}{\sum_\ell \pi_\ell f_\ell(x)} \leftarrow p(Y=k, X)$$

Generative Classifiers

$$p(Y = k \mid X = x) = \frac{\pi_k f_k(x)}{\sum_\ell \pi_\ell f_\ell(x)}$$

- Examples include:
  - Linear and quadratic discriminative analysis (LDA and QDA)
    → linear (+ quad) decision boundaries
  - Mixture of Gaussians (saw in BNP module)
    → nonlinear boundary
  - Nonparametric density estimation for $f_k(x)$
    Very flexible ... scalable?
  - Naïve Bayes
    assumes a simple form for $f_k(x)$

Linear Discriminative Analysis

- Assume Gaussian class-conditional densities

$$f_k(X) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)}$$

- Furthermore, consider equal covariances $\Sigma_k = \Sigma \quad \forall k$
- Log odds

$$\log \frac{p(Y=k \mid X=x)}{p(Y=\ell \mid X=x)} = \log\frac{\pi_k}{\pi_\ell} + \log\frac{f_k(x)}{f_\ell(x)}$$

$$= \log\frac{\pi_k}{\pi_\ell} - \frac{1}{2}(\mu_k+\mu_\ell)^T\Sigma^{-1}(\mu_k-\mu_\ell) + x^T\Sigma^{-1}(\mu_k-\mu_\ell)$$

bc all Σ's same

Linear Discriminative Analysis

$$\log \frac{p(Y=k \mid X=x)}{p(Y=\ell \mid X=x)} = \log\frac{\pi_k}{\pi_\ell} - \frac{1}{2}(\mu_k+\mu_\ell)^T\Sigma^{-1}(\mu_k-\mu_\ell) + x^T\Sigma^{-1}(\mu_k-\mu_\ell)$$

- Equivalently,

$$\log \frac{p(Y=k \mid X=x)}{p(Y=\ell \mid X=x)} = \delta_k(x) - \delta_\ell(x)$$

where

$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + \log\pi_k$$

linear discriminatory fcns

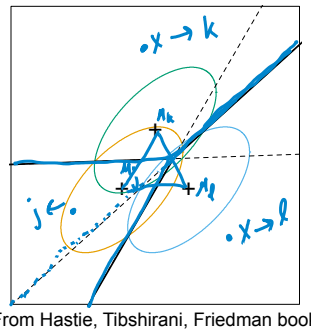
- Decision rule:

$$g(x) = \arg\max_k \delta_k(x)$$ if 0-1 loss

- Linear decision boundaries

$\{x : \delta_k(x) = \delta_\ell(x)\}$

If $\Sigma = \sigma^2 I$, then $\perp$ bisectors



From Hastie, Tibshirani, Friedman book

LDA Parameter Estimation

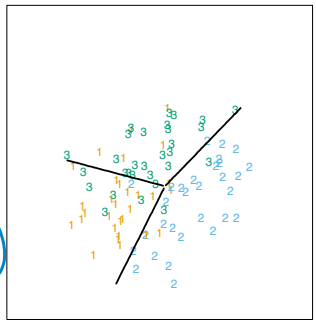
$$\log \frac{p(Y=k \mid X=x)}{p(Y=\ell \mid X=x)} = \log \frac{\pi_k}{\pi_\ell} - \frac{1}{2}(\mu_k + \mu_\ell)^T \Sigma^{-1}(\mu_k - \mu_\ell) + x^T \Sigma^{-1}(\mu_k - \mu_\ell)$$

- Based on the training class labels, $\{y_i\}_{i=1}^n$ estimate parameters:

$$\hat{\pi}_k = \frac{n_k}{n}$$

$$\hat{\mu}_k = \sum_{y_i=k} \frac{x_i}{n_k}$$

$$\hat{\Sigma} = \sum_{k=1}^{K} \sum_{y_i=k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T / (n-K)$$



From Hastie, Tibshirani, Friedman book

Quadratic Discriminative Analysis

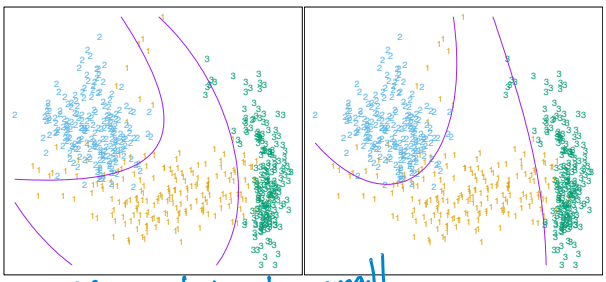
- Same setup as LDA, but allow class-specific covariances Σ_k
- Quadratic discriminant functions: ← quad. in x

$$\delta_k(x) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k) + \log \pi_k$$

- Quadratic decision boundaries $\{x : \delta_k(x) = \delta_\ell(x)\}$

LDA

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_1 x_2 \\ x_1^2 \\ x_2^2 \end{bmatrix}$$



QDA

diff. tend to be small

From Hastie, Tibshirani, Friedman book

QDA Parameter Estimation

- Based on the training class labels, estimate parameters:

$\hat{\pi}_k, \hat{\mu}_k$ same as before

$\hat{\Sigma}_k = \sum_{y_i=k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T / (n_k - K)$

- Number of parameters:

LDA	QDA
$(K-1)(d+1)$	$(K-1)\left(\frac{d(d+3)}{2}+1\right)$

many more params bc of Σ_k's

- Can also consider shrinkage estimators

$$\hat{\Sigma}_k(\alpha) = \alpha\hat{\Sigma}_k + (1-\alpha)\hat{\Sigma} \qquad \hat{\Sigma}(\gamma) = \gamma\hat{\Sigma} + (1-\gamma)\sigma^2 I$$

or

$$\hat{\Sigma}_k(\alpha,\gamma) = \alpha\hat{\Sigma}_k + (1-\alpha)\hat{\Sigma}(\gamma)$$

Notes on QDA and LDA

- LDA + QDA tend to perform very well in practice
- It is not true that data are Gaussian or, furthermore, that covariances are equal (LDA)
- Performance is likely attributed to the fact that the data can only support simple decision boundaries
  - Also, estimates for Gaussian models are stable

bias - var tradeoff

↑ linear bound

↖ lower than in more complex classifiers

(less believable for QDA bc of # of params)

LDA vs. Logistic Regression

- Both have linear log odds:

$$\log \frac{p(Y=k \mid X=x)}{p(Y=K \mid X=x)} = \alpha_{k0} + \alpha_k^T x$$

$$\log \frac{p(Y=k \mid X=x)}{p(Y=K \mid X=x)} = \beta_{k0} + \beta_k^T x$$

- Difference is in how the coefficients are estimated

$$p(X, Y=k) = p(X)\, p(Y=k \mid X)$$

what about this?

same form for both models

LDA vs. Logistic Regression

$$p(X, Y=k) = p(X)p(Y=k \mid X)$$

- Marginal likelihood term

  - Logistic regression: arbitrary... just max cond.-likelihood
    kind of like estimating $p(x)$ fully nonparametricly using the empirical dist. w/ masses $\frac{1}{n}$ @ obs. x_i
  - LDA:
    max based on joint $p(x, Y=k) = \pi_k N(x; \mu_k, \Sigma)$

$$p(x) = \sum_k \pi_k N(x; \mu_k, \Sigma)$$ mixture density

involves the params

LDA vs. Logistic Regression

- In LDA, the data inform the parameters more
  - If data are indeed Gaussian, then asymptotically maximizing just conditional likelihood requires 30% more data to perform as well
- Data far from boundary affect Σ in LDA, but are ignored by logistic regression → LDA is not robust to outliers
- Observations without class labels can be used in mixture model case, but not in logistic regression ← x_i's w/o y_i's
- Marginal likelihood p(X) acts as a regularizer

2-class, linearly separable + log. reg. → ML est. are undefined (used regul.)

LDA coeff. for same data are well-defined

- Logistic regression tends to be more robust than LDA and can handle qualitative X variables, but performance is often similar.

Module 5: Classification

Nonparametric Methods: KDE and Naïve Bayes

STAT/BIOSTAT 527, University of Washington
Emily Fox
May 28th, 2013

KDE for Classification

$\hat{\pi}_k$ as before $\frac{n_k}{n}$

$$p(Y = k \mid X = x) = \frac{\pi_k f_k(x)}{\sum_\ell \pi_\ell f_\ell(x)}$$

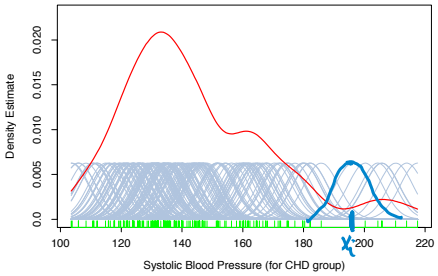
- Use KDE to estimate class-conditional densities
- Recall commonly used Gaussian KDE in 1D

Gauss kernel

$$\hat{f}_k(x) = \frac{1}{n_k} \sum_{y_i = k} \phi_\lambda(x - y_i)$$

$$= (\hat{F}_k * \phi_\lambda)(x)$$

↑ emp. dist. of x



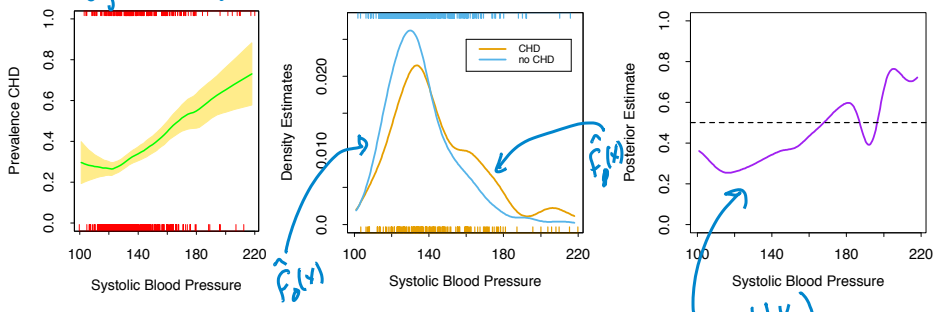
From Hastie, Tibshirani, Friedman book

Example: Heart Disease Data

- Binary response = CHD (coronary heart disease)
- Predictor = systolic blood pressure

KDE for classification

local linear logistic regression

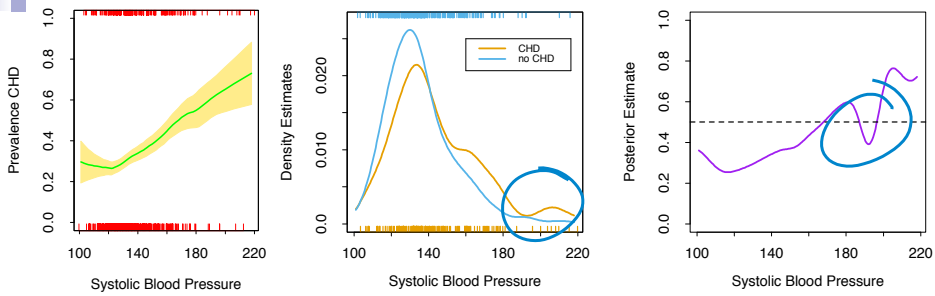


$\hat{f}_0(x)$ $\hat{f}_1(x)$ $p(y_i = 1 \mid x)$

From Hastie, Tibshirani, Friedman book

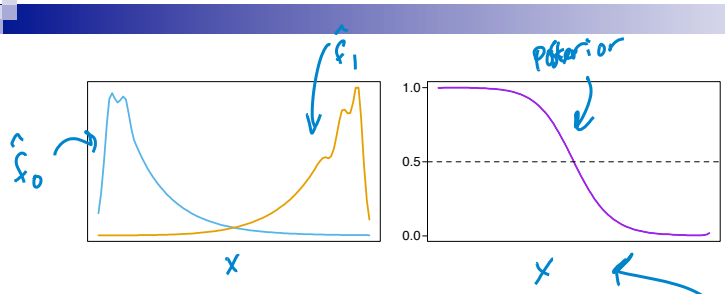
Example: Heart Disease Data

From Hastie, Tibshirani, Friedman book



- KDE estimates are poor in regions with little data
- Local linear model uses variable bandwidth based on k-NN → smooths out over these regions
- For classification tasks, do not need to estimate each class-conditional density well. Just need good estimates of the posterior near the decision boundary

Class-Conditionals vs. Posterior



- Example:
  - Both densities are multimodal
  - Might opt for rougher, high-variance estimator to capture features
  - However, posterior is quite smooth
  - Fine-scale features are irrelevant for classification here

Multivariate KDE

- In 1d $$\hat{p}(x_0) = \frac{1}{n\lambda} \sum_{i=1}^{n} K_\lambda(x_0, x_i)$$
- In R^d, assuming a product kernel, $x \in \mathbb{R}^d$

$$\hat{p}(x_0) = \frac{1}{n\lambda_1 \cdots \lambda_d} \sum_{i=1}^{n} \left\{ \prod_{j=1}^{d} K_{\lambda_j}(x_{0j}, x_{ij}) \right\}$$

- Typical choice = Gaussian RBF → Gaussian KDE

lots of params to choose

$$e^{-\frac{\|x_0 - x\|}{\lambda}}$$

recall, risk increases rapidly w/ dim d

Naïve Bayes Classifier

$$p(Y = k \mid X = x) = \frac{\pi_k f_k(x)}{\sum_\ell \pi_\ell f_\ell(x)}$$

- Useful in high-dimensional settings (*d* large)
- Assumes factored form for class-conditional densities

$$f_k(X) = \prod_{j=1}^{d} f_{kj}(X_j)$$

generally, not true, but often performs well

again, bias-variance tradeoff

- Benefits:
  - Estimate $f_{kj}(X_j)$ separately for each *j* using only 1D KDE
  - If X_j of X is discrete, then can combine using a histogram estimate

allows mixed type predictors

Naïve Bayes Classifier

$$p(Y = k \mid X = x) = \frac{\pi_k \prod_j f_{kj}(x_j)}{\sum_\ell \pi_\ell \prod_j f_{\ell j}(x_j)}$$

- Log odds

$$\log \frac{p(Y = k \mid X = x)}{p(Y = \ell \mid X = x)} = \log \frac{\pi_k}{\pi_\ell} + \sum_j \log \frac{f_{kj}(x_j)}{f_{\ell j}(x_j)}$$

$$= \alpha_{k0} + \sum_{j=1}^{d} g_{kj}(x_j)$$

- Has form of GAM, but fit very differently
  - Analogous to difference between LDA and logistic regression

NB: generative method GAM fitting: discriminative

Module 5: Classification

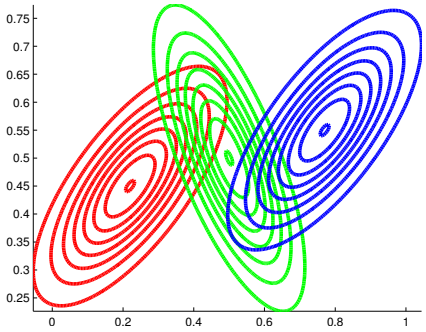
Mixture Models for Classification

STAT/BIOSTAT 527, University of Washington
Emily Fox
May 28th, 2013

Density as Mixture of Gaussians

- Approximate density with a mixture of Gaussians

Mixture of 3 Gaussians



K = 3

$$p(x_i \mid \pi, \mu, \Sigma) = \sum_{k=1}^{K} \pi_k \, N(x_i \mid \mu_k, \Sigma_k)$$

Gauss. kernel, just like in KDE, but not centered at obs.

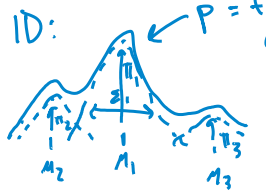
# of mix comp.

mix. weights

shape params

$\Sigma \pi_k = 1$

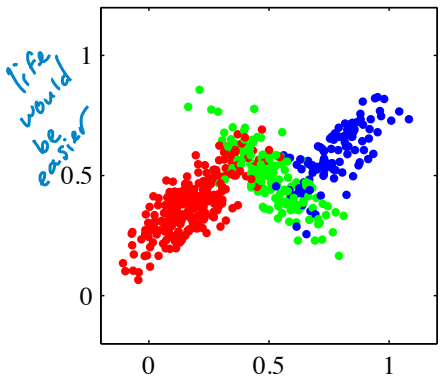
In 1D:



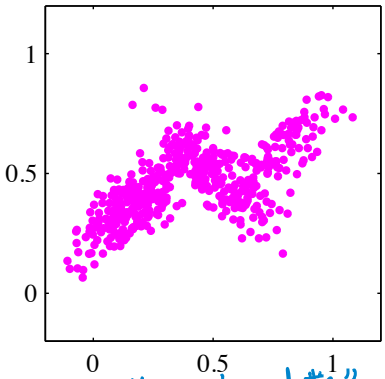
Clustering our Observations

- Imagine we have an assignment of each x_i to a Gaussian

Our actual observations



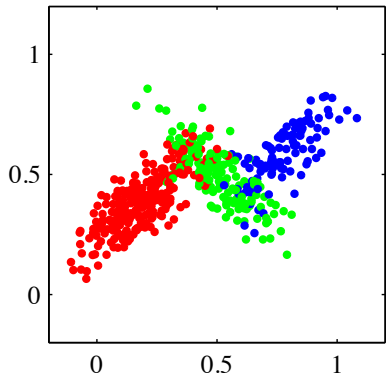
Complete data labeled by true cluster assignments



C. Bishop, Pattern Recognition & Machine Learning

Clustering our Observations

- Imagine we have an assignment of each x_i to a Gaussian



Complete data labeled by true cluster assignments

- Introduce latent cluster indicator variable z_i

$z_i \in \{1, \ldots, K\}$ ← think of as class indicator y_i, but no training data labels

$\Pr(z_i = k) = \pi_k$

- Then we have

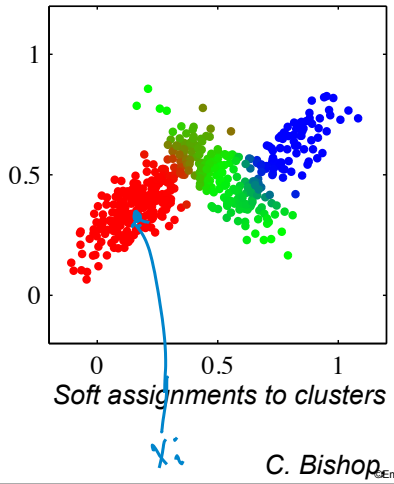
$$p(x_i \mid z_i = k, \pi, \mu, \Sigma) = N(x_i \mid \mu_{z_i = k}, \Sigma_{z_i = k})$$

Param. est. is easy if we have $\{z_i\}$ ⇒ decoupled into K Gauss. est.

C. Bishop, Pattern Recognition & Machine Learning

Clustering our Observations

- We must infer the cluster assignments from the observations



Soft assignments to clusters

- Posterior probabilities of assignments to each cluster *given* model parameters:

"responsibility"

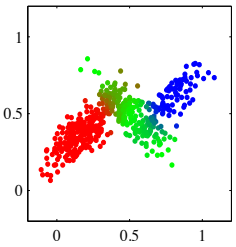
$$r_{ik} = p(z_i = k \mid x_i, \pi, \theta) = \frac{\pi_k N(x_i \mid \mu_k, \Sigma_k)}{\sum_j \pi_j N(x_i \mid \mu_j, \Sigma_j)}$$

motivates an iterative alg.

C. Bishop, Pattern Recognition & Machine Learning

Mixture Models for Classification

- Can use mixture models as a generative classifier in the unsupervised setting



- EM algorithm = iteratively:
  - Estimate responsibilities given parameter estimates

$$\hat{r}_{ik} = \frac{\hat{\pi}_k N(x_i, \hat{\mu}_k, \hat{\Sigma}_k)}{\sum_\ell \hat{\pi}_\ell N(x_i, \hat{\mu}_\ell, \hat{\Sigma}_\ell)}$$

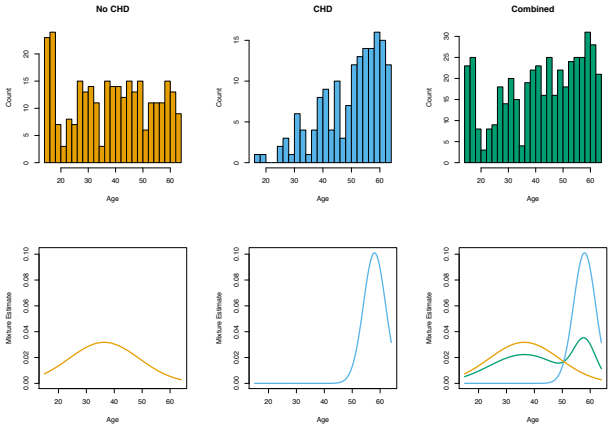
  - Maximize parameters given responsibilities

- For classification, threshold the estimated responsibilities
  - E.g., $\hat{g}(x_i) = \arg\max_k \hat{r}_{ik}$

- Note: allows non-linear boundaries as in QDA

Example: Heart Disease Data

- Binary response = CHD (coronary heart disease)
- Predictor = systolic blood pressure



From Hastie, Tibshirani, Friedman book

What you need to know

- Discriminative vs. Generative classifiers
- LDA and QDA assume Gaussian class-conditional densities
  - Results in linear and quadratic decision boundaries, respectively
- KDE for classification
  - Challenging in areas with little data or in high dimensions
  - Estimating class-conditionals is not optimizing classification objective
- Naïve Bayes assumes factored form
  - Results in log odds that have GAM form
- Mixture models allow for unsupervised generative approach

Readings

- Hastie, Tibshirani, Friedman – 4.3, 4.4.5, 6.6.2-6.6.3, 6.8