Module 5: Classification

Linear Methods: Logistic Regression

STAT/BIOSTAT 527, University of Washington

Emily Fox

May 22nd, 2014

Very convenient!

$\log\left(\frac{x}{1-x}\right)$ $E(y) = \beta_0 + \beta_1 x + \ldots$

$g = \text{logit} =$ link fn. $\rightarrow$ $\underbrace{g(E[y])}_{\mathbb{R}} = \underbrace{\beta_0 + \beta_1 x + \ldots}$

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

implies

$1 - P(y=0 \mid x)$ $\mathbb{R}$

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

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

$g(y)$

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

$\Rightarrow$ 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

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

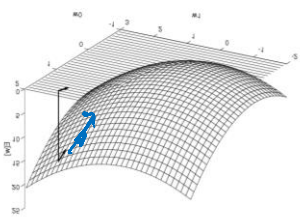
Good news: $l(\boldsymbol{\beta})$ is concave function of $\boldsymbol{\beta}$, no local optima problems

Bad news: no closed-form solution to maximize $l(\boldsymbol{\beta})$

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

$$\underbrace{\beta_j^{(t+1)}}_{\text{new est.}} \leftarrow \underbrace{\beta_j^{(t)}}_{\text{prev. est}} + \underbrace{\eta}_{\text{stepsize}} \frac{\partial l(\beta)}{\partial \beta_j}$$

- 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

revisit soon

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

Gradient ascent algorithm: iterate until change < ε

can do in || ∀j

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

For j=1,…,d,

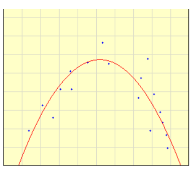
$$\beta_j^{(t+1)} \leftarrow \beta_j^{(t)} + \eta \underbrace{\sum_i \underline{x_{ij}} \left(y_i - \hat{p}(y=1 \mid x_i, \beta^{(t)})\right)}_{\frac{\partial L(\beta)}{\partial \beta_j}\ j=1,\dots,d}$$

repeat

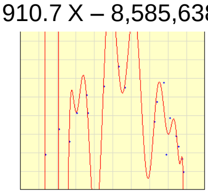
Regularization in Linear Regression

- Overfitting usually leads to very large parameter choices, e.g.:

-2.2 + 3.1 X – 0.30 X^2



-1.1 + 4,700,910.7 X – 8,585,638.4 X^2 + …



even for $n > p$, p large

- ***Regularized*** or ***penalized*** regression aims to impose a "complexity" penalty by penalizing large weights
  - "Shrinkage" method

$$\hat{\beta} = \arg\min_{\beta} \underbrace{\sum_{i=1}^{n} (y_i - (\beta_0 + \beta^T x_i))^2}_{\text{RSS}} + \underbrace{\lambda ||\beta||}_{\text{penalty}}$$

Linear Separability

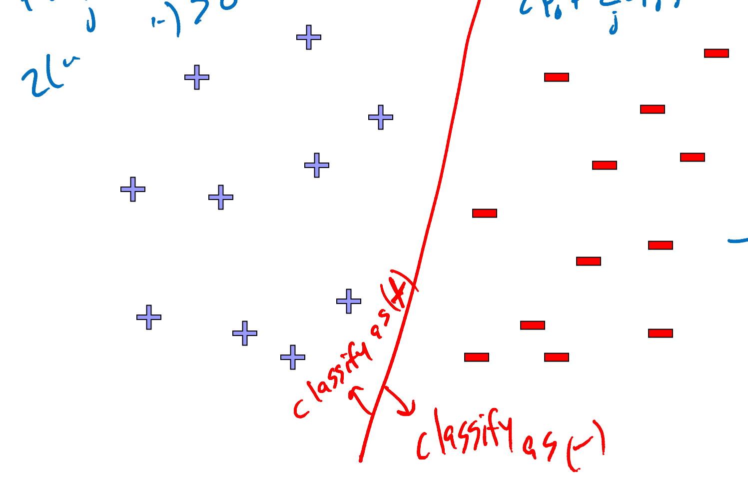
$\exists \beta$ s.t. all pos. examples have $\beta_0 + \sum_j \beta_j x_j > 0$ & neg. examples have " " < 0

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

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

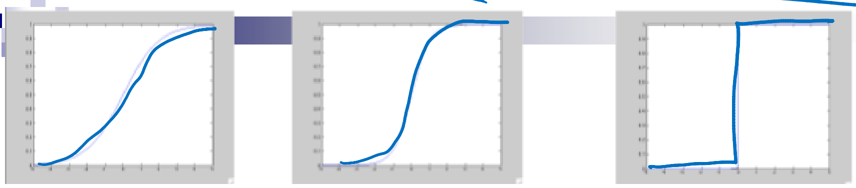
$\beta_0 + \sum_j \beta_j x_j > 0$

$2(\ldots) > 0$



$\rightarrow \alpha\vec{\beta}$ 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

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

  - In general, leads to overfitting: regularize!! $\rightarrow$ $\|\beta\|_2^2$, $\|\beta\|_1$
- Penalizing high weights can prevent overfitting…

Regularized Conditional Log Likelihood

- Add regularization penalty, e.g., L_2:

$$l(\beta) = \underbrace{\log \prod_{i=1}^{n} p(y_i \mid x_i, \beta)}_{\text{log-likelihood}} - \frac{\lambda}{2}||\beta||_2^2$$

- Practical note about β_0:

- Gradient of regularized likelihood:

Standard v. Regularized Updates

- Maximum conditional likelihood estimate

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

normal logistic

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

- Regularized maximum conditional likelihood estimate

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

regularized

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

pushes coeff. towards 0

Stopping Criterion

$$l(\beta) = \log \prod_{i=1}^{n} p(y_i \mid x_i, \beta) - \frac{\lambda}{2}||\beta||_2^2$$

- When do we stop doing gradient ascent?

$$||l(\beta^*) - l(\beta^{(t)})|| \le \varepsilon$$

- Because $l(\mathbf{w})$ is strongly concave:
  - i.e., because of some technical condition

$$l(\beta^*) - l(\beta) \le \frac{1}{2\lambda}||\nabla l(\beta)||_2^2$$

- Thus, stop when:

$$||l(\beta^*) - l(\beta)|| < \boxed{\frac{1}{2\lambda}||\nabla l(\beta)||_2^2 < \varepsilon}$$

stopping criterion

Digression: Logistic Regression for $K > 2$

- Logistic regression in more general case (K classes), where Y in $\{1,\ldots,K\}$

Digression: Logistic Regression for $K > 2$

- Logistic regression in more general case, where *Y in* {1,…,*K*}

for *k*<*K*

$$p(y = k|\mathbf{x}, \beta) = \frac{\exp(\beta_{k0} + \sum_{j=1}^{d} \beta_{kj} x_j)}{1 + \sum_{k'=1}^{K-1} \exp(\beta_{k'0} + \sum_{j=1}^{d} \beta_{k'j} x_j)}$$

for *k*=*K* (normalization, so no weights for this class)

$$p(y = K|\mathbf{x}, \beta) = \frac{1}{1 + \sum_{k'=1}^{K-1} \exp(\beta_{k'0} + \sum_{j=1}^{d} \beta_{k'j} x_j)}$$

Estimation procedure is basically the same as what we derived!

The Cost, The Cost!!! Think about the cost…

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

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

$\forall j$

$O(d)$

$\beta_0^{(t)}, \ldots, \beta_d^{(t)}$

$O(nd)$

Naively, $O(nd \cdot d = nd^2)$

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

However, if "n" is huge (or online streaming), this is slow.

Gradient ascent in Terms of Expectations

- "True" objective function:

$$l(\beta) = \underbrace{E_x[l(\beta,x)]}_{\text{expected loss}} = \int p(x)l(\beta,x)dx$$

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

- 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 it so approx. from sample

SGD: Stochastic Gradient Ascent (or Descent)

- "True" gradient: $\nabla l(\beta) = E_x[\nabla l(\beta,x)]$

- Sample based approximation: take x_i iid

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

use this for normal logistic reg.

- What if we estimate gradient with just one sample???
  - Unbiased estimate of gradient
  - 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:

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

$\frac{1}{n}\sum_{i=1}^{n} \nabla l(\beta, x_i)$

- Batch gradient ascent updates:

normal LR

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

new

- Stochastic gradient ascent updates:
  - Online setting: take 1 data pt. 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 this 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 22nd, 2014

Discriminative vs. Generative

class label

- So far, we have considered modeling/fitting

  $p(Y \mid X)$ input predictor

  "discriminative" method for fixed X, max P(Y|X) for fitting

- There are also a large set of ***generative*** methods
- Model:
  - Class-conditional densities $f_k(X) =$ $P(X|Y=k)$
  - Class prior probabilities π_k $= P(Y=k) \to \sum_k \pi_k = 1$
- Via Bayes' rule:

$$P(Y=k \mid X=x) = \frac{\pi_k f_k(x)}{\sum_l \pi_l f_l(x)} = P(Y=k, X)$$

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)

    → non-linear boundary
  - Nonparametric density estimation for $f_k(x)$

    KDE, very flexible
  - Naïve Bayes

    assumes a simple form for $f_k(x)$

Linear Discriminative Analysis

- Assume Gaussian class-conditional densities

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

$\uparrow \mathbb{R}^d$

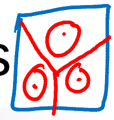
- Furthermore, consider equal covariances $\Sigma_k = \Sigma, \forall k$
- Log odds

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

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

(Σ's same) ↑ linear in x

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,
$$\log \frac{p(Y=k \mid X=x)}{p(Y=\ell \mid X=x)} = \delta_k(x) - \delta_\ell(x) = 0$$
where

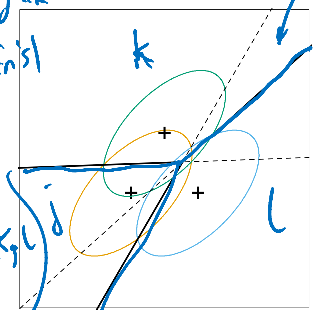
boundaries

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

linear

(linear discriminant fn's)

- Decision rule:
$$\hat{g}(x) = \arg\max_k \delta_k(x)$$
- Linear decision boundaries
$$\{x : \delta_k(x) = \delta_\ell(x)\}, \forall k,\ell$$



if $\Sigma = \sigma^2 I$,

From Hastie, Tibshirani, Friedman book

LDA Parameter Estimation

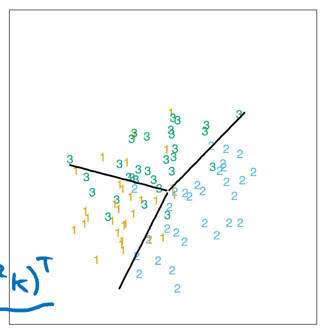
$$\log \frac{p(Y=k \mid X=x)}{p(Y=\ell \mid X=x)} = \log \frac{\pi_k}{\pi_\ell} - \frac{1}{2}(\mu_k+\mu_\ell)^T\Sigma^{-1}(\mu_k-\mu_\ell) + x^T\Sigma^{-1}(\mu_k-\mu_\ell)$$

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

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

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

$$\hat{\Sigma} = \frac{\sum_{k=1}^{K}\sum_{y_i=k}(x_i-\mu_k)(x_i-\mu_k)^T}{n-K}$$



From Hastie, Tibshirani, Friedman book

Quadratic Discriminative Analysis

- Same setup as LDA, but allow class-specific covariances

Σ_k are diff.

LDA $x^T \ldots$ $x^T A x$ QDA (quad.)

- Quadratic discriminant functions:

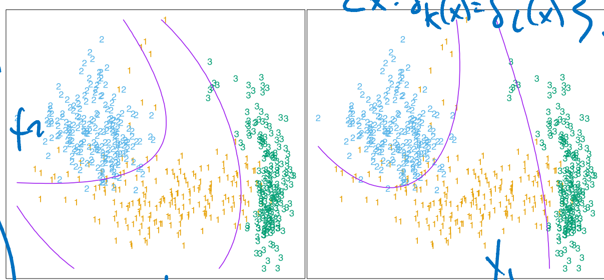
$$\delta_k(x) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}\underbrace{(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)}_{\text{quad.}} + \log\pi_k$$

- Quadratic decision boundaries

$\{x : \delta_k(x) = \delta_l(x)\}$

LDA

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_1x_2 \\ x_1^2 \\ x_2^2 \end{pmatrix}$$



x_2 $\vec{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$

QDA

x_1 x_2 x_1

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 parameters from Σ_k

- Can also consider shrinkage estimators

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

distinct Σ's common Σ common Σ spherical

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 - variance tradeoff

linear boundary

lower than in more complex classifiers

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

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

huh?

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 maximize likelihood kind of like estimating $p(x)$ nonparametrically empirically w/ mass "$\frac{1}{n}$ @ each x_i".
  - LDA:

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

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 (LDA)
- Marginal likelihood p(*X*) acts as a regularizer
  2-class, lin. separable + log.reg. → ML est. are undefined
  → 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 22nd, 2014

KDE for Classification

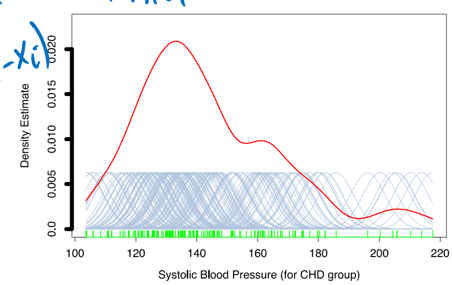
$\hat{\pi}_k$ as before

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

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

gaussian kernel

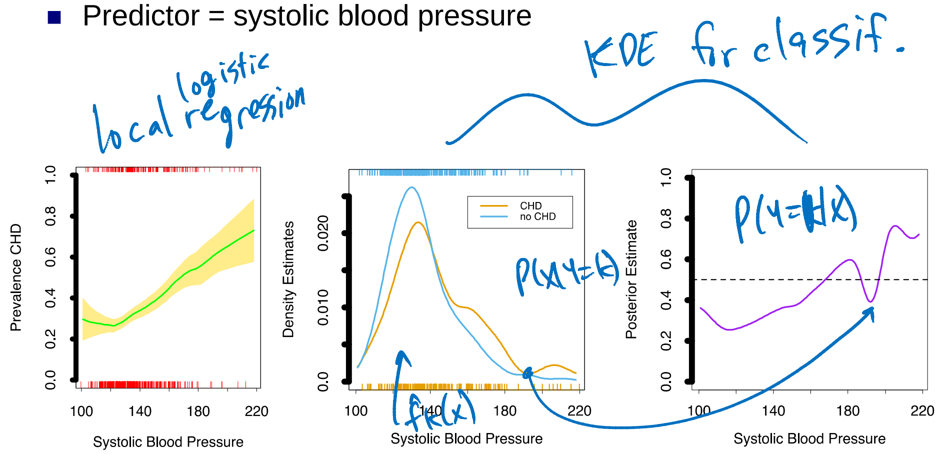
$$\hat{f}_k(x) = \frac{1}{n_k} \sum_{y_i = k} \phi_\lambda(x - x_i)$$



From Hastie, Tibshirani, Friedman book

Example: Heart Disease Data

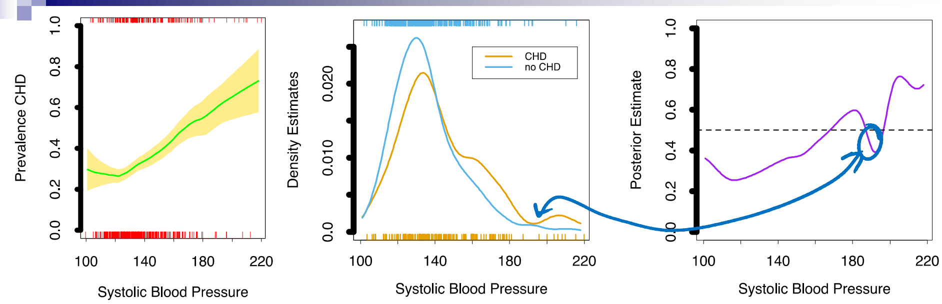
- Binary response = CHD (coronary heart disease)
- Predictor = systolic blood pressure



From Hastie, Tibshirani, Friedman book

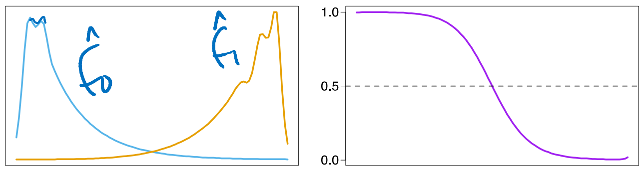
Example: Heart Disease Data

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

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

lots of params to choose

- Typical choice = Gaussian RBF → Gaussian KDE

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

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

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

$\uparrow$ $\mathbb{R}^d$

generally, not true, but often performs well.

- 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

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

$$\log \frac{p(Y = k \mid X = x)}{p(Y = \ell \mid X = x)} = \log \frac{\pi_k}{\pi_l} + \sum_i \log \frac{f_{kj}(x_j)}{f_{lj}(x_j)}$$

$$= \alpha_{k_0} + \sum_{j=1}^{d} g_{kj}(x_j)$$

"discriminative" $\downarrow$

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

NB: generative

Module 5: Classification

Mixture Models for Classification

STAT/BIOSTAT 527, University of Washington

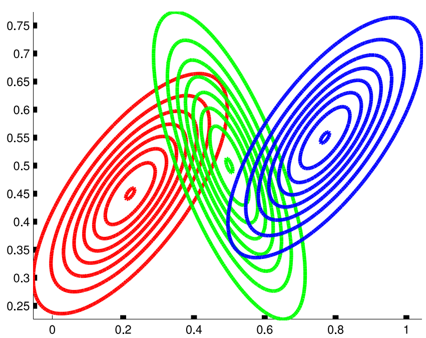
Emily Fox

May 22nd, 2014

Density as Mixture of Gaussians

- Approximate density with a mixture of Gaussians

Mixture of 3 Gaussians



$K=3$

$P = $

$$p(x_i \mid \pi, \mu, \Sigma) = \sum_{k=1}^{K} \underbrace{\pi_k}_{\text{mix. weights}} N(x_i \mid \underbrace{\mu_k, \Sigma_k}_{\text{shape params}})$$

$= [\pi_1, \ldots, \pi_K]$

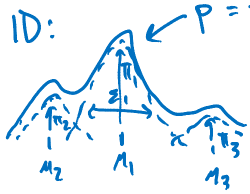
$\{\mu_k, \Sigma_k\}$

\# of mix comp.

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

In 1D:

P = target density



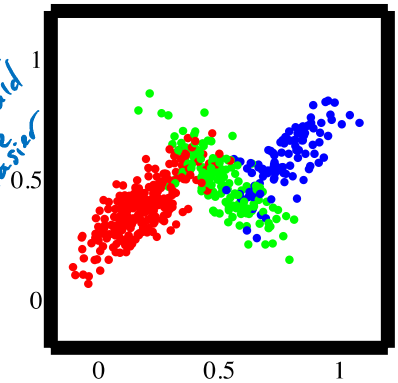
$\sum \pi_k = 1$

Clustering our Observations

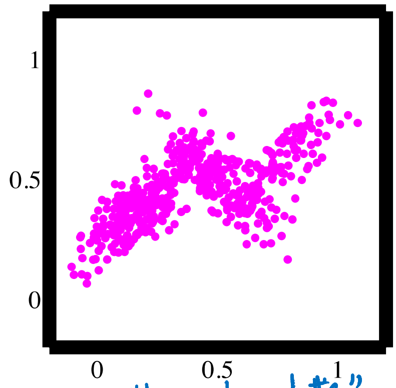
- Imagine we have an assignment of each x_i to a Gaussian

Our actual observations

life would be easier



Complete data labeled by true cluster assignments

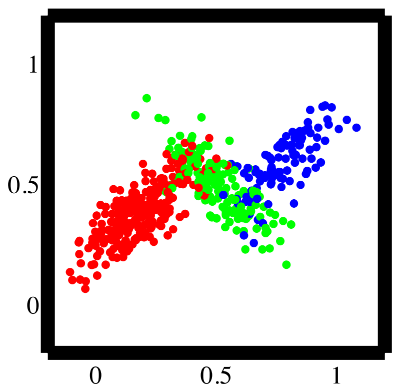


"Incomplete data"

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

$z_i \in \{1, \dots, K\}$

$Pr(z_i = k) = \pi_k$

- Then we have

$$p(x_i \mid z_i = k, \pi, \mu, \Sigma) =$$

$$N(x_i \mid \mu_{z_i = k}, \Sigma_{z_i = k})$$

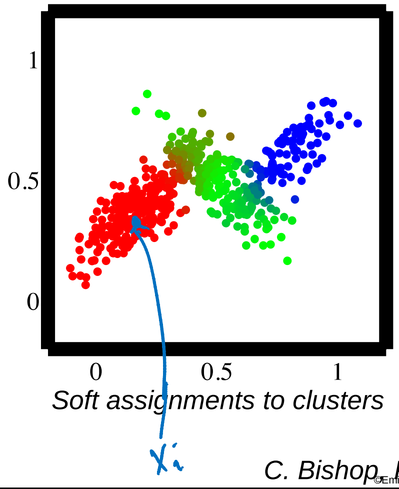
param. est. is easy if we have $\{z_i\}$

$\Rightarrow$ 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:

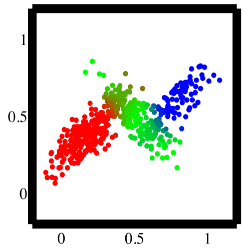
$$r_{ik} = p(z_i = k \mid x_i, \pi, \theta) = \frac{\pi_k N(x_i \mid \mu_k, \Sigma_k)}{\sum_j \pi_j N(x_i \mid \mu_j, \Sigma_j)}$$

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

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

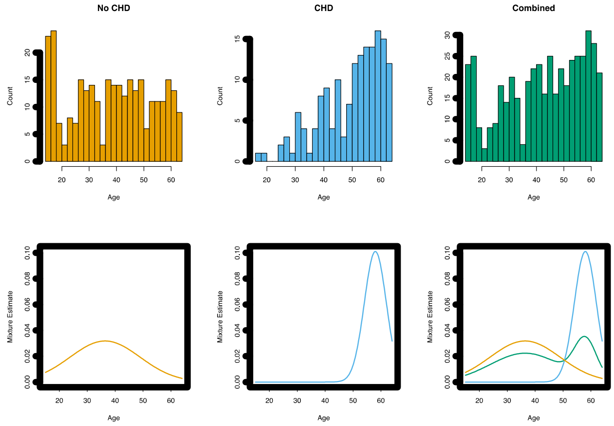
  - Maximize parameters given responsibilities

- 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