

# CSE 6740 Lecture 7

## *How Do I Predict a Discrete Variable? (Classification)*

Alexander Gray

agray@cc.gatech.edu

Georgia Institute of Technology

# Today

1. Regression (cont'd) (*How can I predict a continuous variable?*)
2. Classification (*How can I predict a discrete variable?*)
3. Generative classification (*What if I reduce classification to density estimation?*)
4. Discriminative classification (*Can I do classification while avoiding density estimation?*)

# Regression (cont'd)

How can I predict a continuous variable?

# Kernel Regression: Nadaraya-Watson

Recall the definition of the regression function

$$r(x) = \mathbb{E}(Y|X = x) = \int y f(y|x) dy \quad (1)$$

$$= \frac{\int y f(x, y) dy}{\int f(x, y) dy} \quad (2)$$

Let's estimate this somewhat explicitly:

$$\hat{r}(x) = \frac{\sum_{i=1}^N K\left(\frac{\|x-x_i\|}{h}\right) y_i}{\sum_{i=1}^N K\left(\frac{\|x-x_i\|}{h}\right)}. \quad (3)$$

# Kernel Regression: Nadaraya-Watson

This can be viewed as taking a weighted average of the  $y$ 's, giving higher weight to points near  $x$ :

$$\hat{r}(x) = \sum_{i=1}^N w_i(x) y_i \quad (4)$$

where

$$w_i(x) = \frac{K\left(\frac{\|x-x_i\|}{h}\right)}{\sum_{i=1}^N K\left(\frac{\|x-x_i\|}{h}\right)}. \quad (5)$$

# Kernel Regression: Nadaraya-Watson

**Task:** regression

**Model class:** Sobolev (nonparametric)

**Loss:**  $L_2$  error

**Optimizer:** exhaustive or gradient descent

**Generalization mechanism:** cross-validation

**Evaluation algorithm:** generalized  $N$ -body algorithm

We can compute the risk of the Nadaraya-Watson estimator, as we did for the kernel density estimator. We can then use this to find that the optimal bandwidth decreases at rate  $N^{-1/5}$  and with this choice the risk decreases at rate  $N^{-4/5}$ . This turns out to be the optimal rate of convergence.

# Kernel Regression: Nadaraya-Watson

Cross-validation is straightforward in the supervised case:

$$\text{CV}_{L_2}(h) = \frac{1}{N} \sum_{i=1}^N (y_i - \widehat{r}_h^{-i}(x_i))^2. \quad (6)$$

We actually don't need to compute separate estimates in order to leave one out of each, by rewriting this as

$$\text{CV}_{L_2}(h) = \frac{1}{N} \sum_{i=1}^N (y_i - \widehat{r}_h(x_i))^2 / \left( 1 - \frac{K(0)}{\sum_{j=1}^N K\left(\frac{\|x_i - x_j\|}{h}\right)} \right)^2. \quad (7)$$

# Kernel Regression: Nadaraya-Watson

An approximate  $1 - \alpha$  confidence band for  $\bar{r}(x)$  is

$$l(x) = \hat{r}(x) - q\widehat{\mathbf{se}}(x), \quad u(x) = \hat{r}(x) + q\widehat{\mathbf{se}}(x) \quad (8)$$

where

$$\hat{\sigma}^2 = \frac{1}{2(N-1)} \sum_{i=1}^{N-1} (y_{i+1} - y_i)^2, \quad (9)$$

$$\widehat{\mathbf{se}}(x) = \hat{\sigma} \sqrt{\sum_{i=1}^N w_i^2(x)}, \quad (10)$$

$$(11)$$

# Kernel Regression: Nadaraya-Watson

$$q = \Phi^{-1} \left( \frac{1 + (1 - \alpha)^{1/m}}{2} \right), \quad (12)$$

$$m = (b - a) / \tilde{h}, \quad (13)$$

where  $\tilde{h}$  is the effective width of the kernel (use  $3h$  for the Gaussian kernel).

Note that this is not exactly a confidence band on the true regression function for technical reasons.

# Kernel Regression: Locally Linear

Problem: Points near the boundary of the training data will be poorly estimated.

Consider *locally linear regression*, where we will solve a separate weighted least-squares problem at each point  $x$  to be predicted, finding the  $\beta(x)$  which minimizes

$$\sum_{i=1}^N K \left( \frac{\|x - x_i\|}{h} \right) (y_i - \beta(x)x_i)^2. \quad (14)$$

# Kernel Regression: Locally Linear

The estimate is then, where  $W(x)$  is the  $N \times N$  diagonal matrix with  $i^{th}$  diagonal element  $K\left(\frac{\|x-x_i\|}{h}\right)$ ,

$$\hat{r}(x) = \hat{\beta}(x)x \quad (15)$$

$$= x^T \left( \mathbf{X}^T W(x) \mathbf{X} \right)^{-1} \mathbf{X}^T W(x) \mathbf{Y} \quad (16)$$

$$= \sum_{i=1}^N w_i(x) y_i. \quad (17)$$

Note that the overall estimate is linear in  $y_i$ , since the weights  $w_i(x)$  do not involve  $y_i$ . They can be thought of as representing an *equivalent kernel*.

# Kernel Regression: Extensions

There are further extensions possible.

- *Local polynomial regression.* We can consider any polynomial order. However there is a bias-variance (complexity) tradeoff, as usual. The general consensus is that going past linear increases variance without decreasing bias much, since local linear regression captures most of the boundary bias, and asymptotically boundary effects dominate the MSE.

# Kernel Regression: Extensions

- *Variable-bandwidth kernels.* For example let the bandwidth for each training point be inversely proportional to its  $k^{th}$  nearest-neighbor's distance. Generally a good idea in practice, though there is less theoretical consensus on how to choose the parameters here.

None of these improves the convergence rate. However, they can still give better finite-sample results.

Note that all of this can be done in kernel density estimation as well.

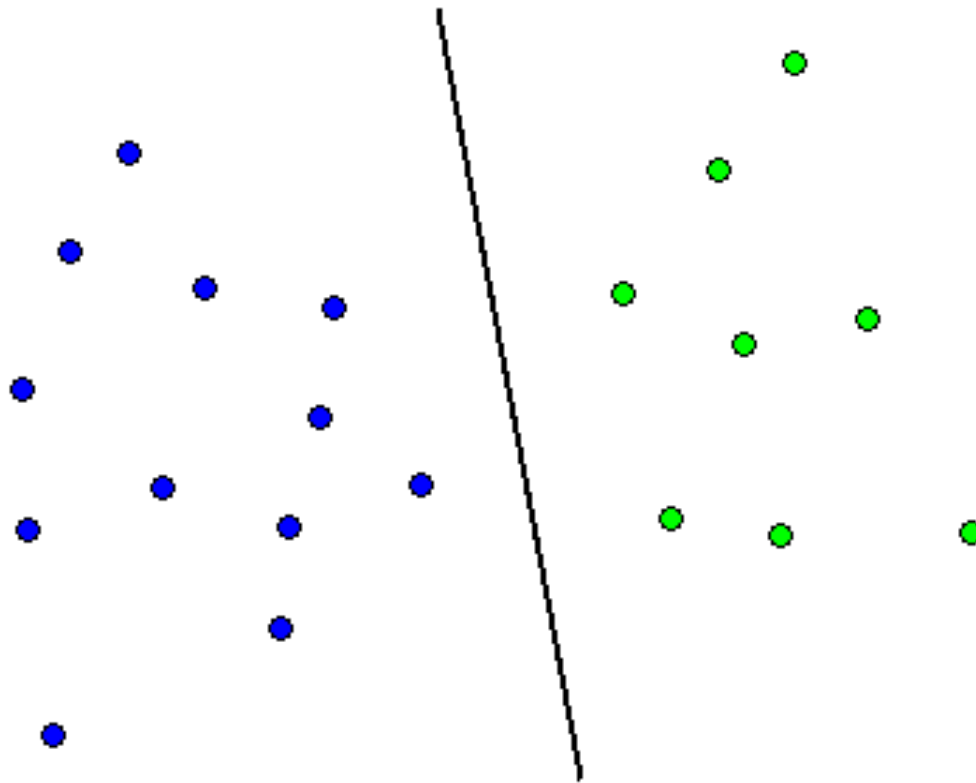
# Main Things You Should Know

- What linear regression is
- What kernel regression is
- What regularization is, and its importance

# Classification

How can I predict a discrete variable?

# Classification



# Classification Loss

The most common loss function for classification is *zero-one loss*:

$$L(Y, \hat{f}(X)) = I(Y \neq \hat{f}(X)). \quad (18)$$

We can generalize this to specify arbitrary costs for misclassifying one class as another:

$$L(Y, \hat{f}(X)) = C_{ab} \quad (19)$$

where  $C$  is a  $K \times K$  matrix,  $a = Y$ , and  $b = \hat{f}(X)$ .

# Classification Loss

The test error, or expected loss, called the *error rate* in this case, is

$$E = \mathbb{E} \left[ L(Y, \hat{f}(X)) \right] \quad (20)$$

$$= \mathbb{E}_{X,Y} \left[ L(Y, \hat{f}(X)) \right] \quad (21)$$

$$= \mathbb{E}_X \mathbb{E}_{Y|X} \left[ L(Y, \hat{f}(X)) \right] \quad (22)$$

or, for a given  $x$ ,

$$E(x) = \mathbb{E}_{Y|X} \left[ L(Y, \hat{f}(x)) \right]. \quad (23)$$

# Discriminant Function

Suppose  $Y = \{0, 1\}$ . The value  $\hat{f}^*(x)$  that minimizes  $E(x)$  is the regression function, which we now call the *discriminant function*:

$$g(x) = \mathbb{E}(Y|X = x) \quad (24)$$

$$= \int y f(y|x) dy \quad (25)$$

$$= 1 \cdot \mathbb{P}(Y = 1|X = x) + 0 \cdot \mathbb{P}(Y = 0|X = x) \quad (26)$$

$$= \mathbb{P}(Y = 1|X = x) \quad (27)$$

$$= \frac{f(x|Y = 1)\mathbb{P}(Y = 1)}{f(x|Y = 1)\mathbb{P}(Y = 1) + f(x|Y = 0)\mathbb{P}(Y = 0)} \quad (28)$$

$$= \frac{\pi_1 f_1(x)}{\pi_1 f_1(x) + \pi_0 f_0(x)}. \quad (29)$$

# Bayes Classifier

Making this yield a binary prediction gives the *Bayes classifier*, or *Bayes rule*:

$$c(x) = \begin{cases} 1 & \text{if } g(x) > 1/2 \\ 0 & \text{otherwise} \end{cases} \quad (30)$$

$$= \begin{cases} 1 & \text{if } \mathbb{P}(Y = 1|X = x) > \mathbb{P}(Y = 0|X = x) \\ 0 & \text{otherwise} \end{cases} \quad (31)$$

$$= \begin{cases} 1 & \text{if } \pi_1 f_1(x) > \pi_0 f_0(x) \\ 0 & \text{otherwise.} \end{cases} \quad (32)$$

This is easily generalized to any number of classes  $K$ .

# Optimal Classification

Keep in mind that this is “Bayes” only in the sense of conditional distributions, not in the sense of Bayesian inference.

The Bayes classifier is optimal, *i.e.* if  $c'(x)$  is any other classification rule then  $\mathbb{E} [L(Y, c'(X))] > \mathbb{E} [L(Y, c(X))]$ .

# Generative vs. Discriminative

The set

$$\{x : \mathbb{P}(Y = 1|X = x) = \mathbb{P}(Y = 0|X = x)\} \quad (33)$$

is called the *decision boundary*.

In a *generative* classifier, we'll model the class-conditional densities  $f_k(x)$  explicitly. This means we'll be doing two separate *density estimates*.

In a *discriminative* classifier, we'll avoid that and directly model the discriminant function, which is tantamount to modeling the decision boundary.

# Classification: Generative Approaches

What if I reduce classification to density estimation?

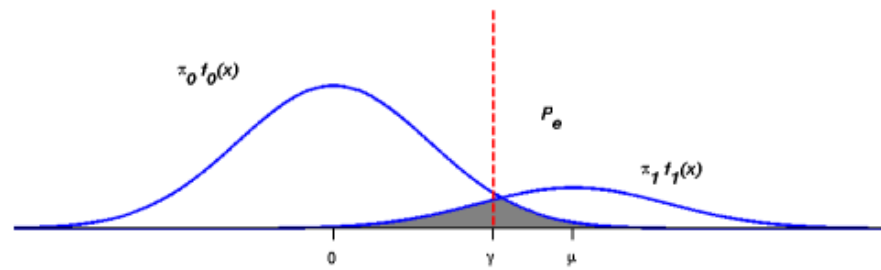
# Bayes Classifier: Gaussian

What should we use for  $f_0(x)$  and  $f_1(x)$ ? Let's start with the Gaussian

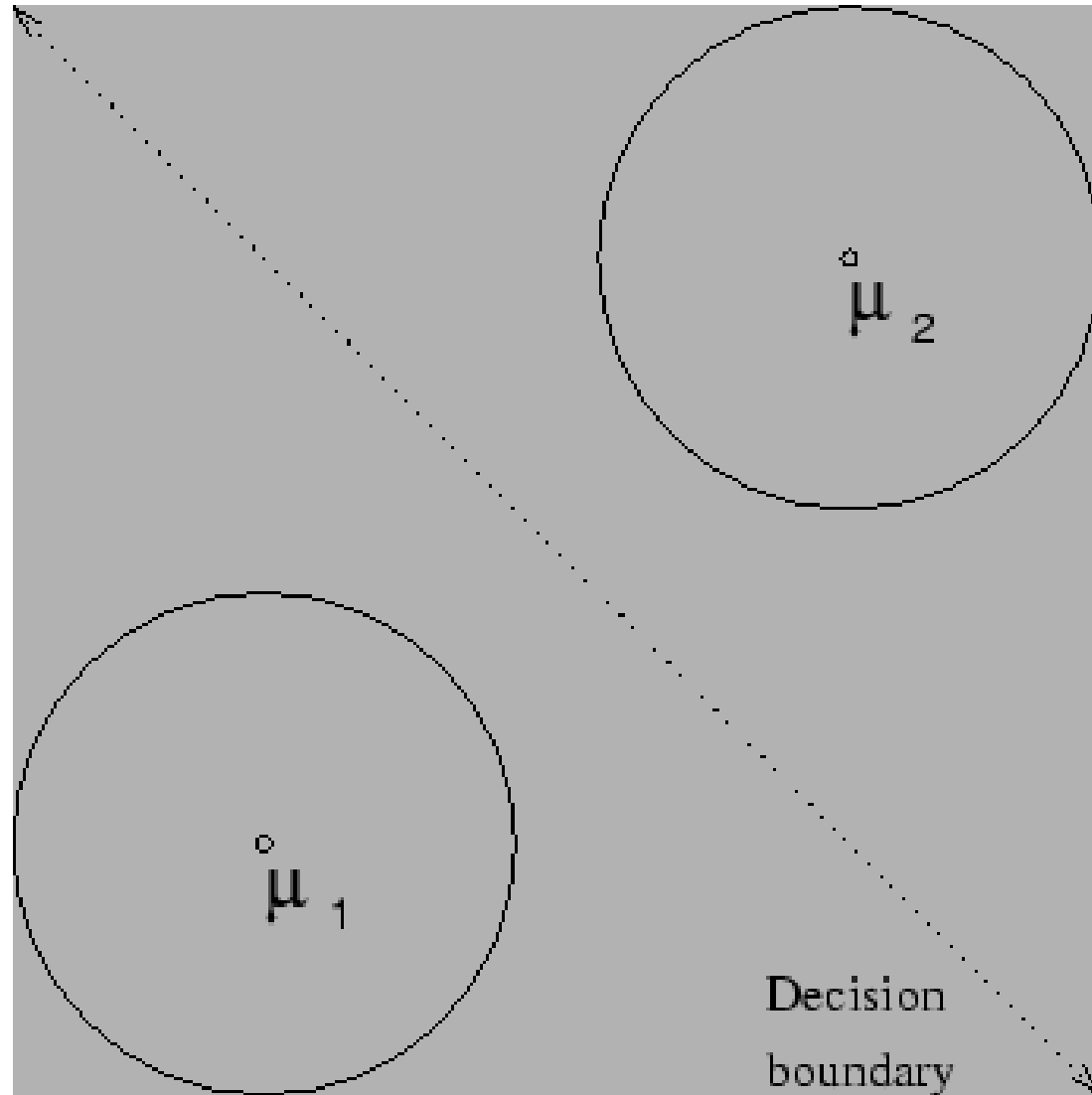
$$f_k(x) = \frac{(2\pi)^{D/2}}{|\Sigma_k|^{1/2}} \exp \left\{ -\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) \right\}, \quad (34)$$

*i.e.*  $X|Y = 0 \sim N(\mu_0, \Sigma_0)$  and  $X|Y = 1 \sim N(\mu_1, \Sigma_1)$ .

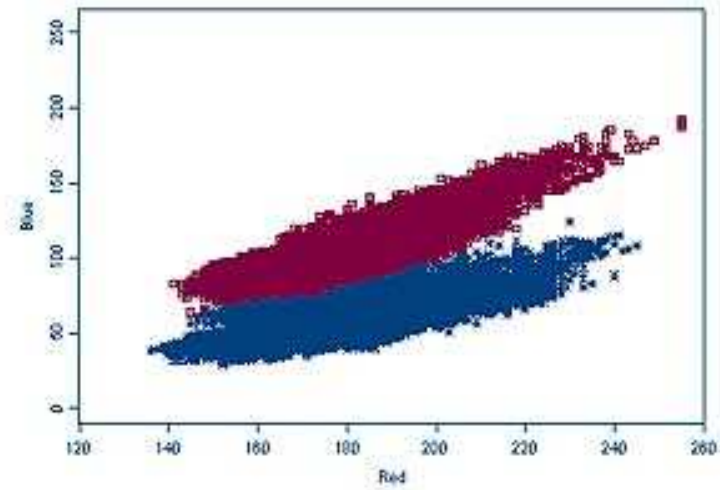
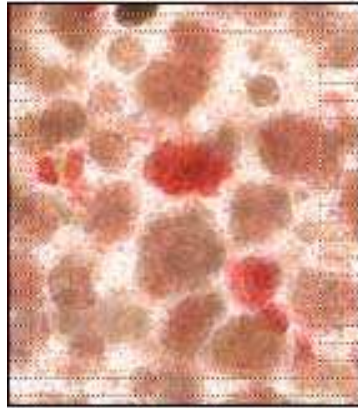
# Bayes Classifier



# Bayes Classifier



# Bayes Classifier



# Bayes Classifier: Gaussian

The Bayes classifier is then

$$c(x) = \arg \max_k \left\{ -\frac{1}{2} \log |\Sigma_k| - \frac{1}{2} (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) + \log \pi_k \right\} \quad (35)$$

or equivalently

$$c(x) = \begin{cases} 1 & \text{if } m_1^2 < m_0^2 + 2 \log \left( \frac{\pi_1}{\pi_0} \right) + \left( \frac{\Sigma_1}{\Sigma_0} \right) \\ 0 & \text{otherwise,} \end{cases} \quad (36)$$

where  $m_k^2 = (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k)$  is the *Mahalanobis distance*.

# Bayes Classifier: Gaussian

To obtain the empirical version of this rule, we estimate the parameters using the data for each class separately.

This is sometimes called *quadratic discriminant analysis* because the decision boundary has a conic form. The special case where both covariance matrices are diagonal is called *naive Bayes* or *idiot Bayes*.

**Task:** classification

**Model class:** all possible Gaussians, for each class

**Loss:** likelihood

**Optimizer:** none (sample means and covariances)

**Generalization mechanism:** none

# LDA and FLD

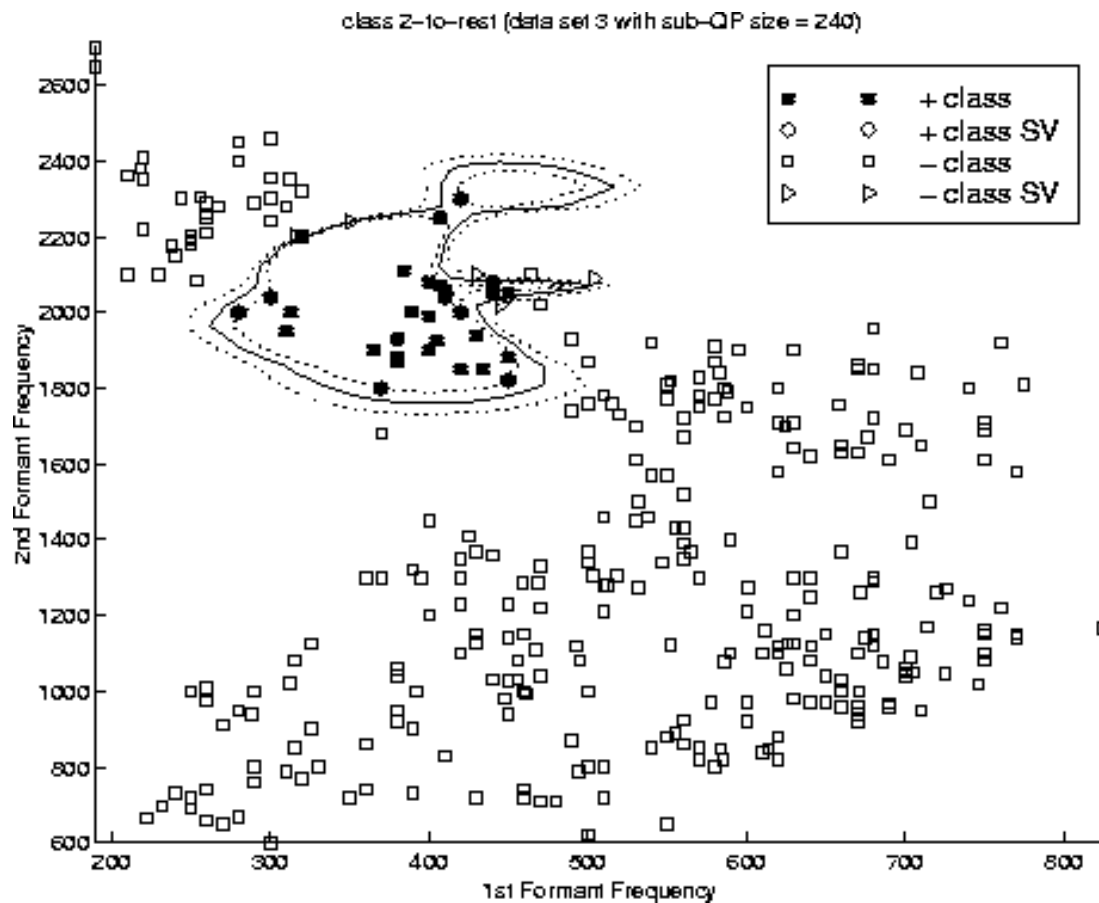
In the case where we assume that  $\Sigma_0 = \Sigma_1 = \Sigma$  (but the covariance matrix may still be arbitrary), the Bayes classifier is

$$c(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k. \quad (37)$$

This is called *linear discriminant analysis* (LDA) because the decision boundary is linear.

*Fisher's linear discriminant* (FLD) is a special case where the priors are set equal. It is only interesting because it is derived in a completely different way. It attempts to find a projection of all the data onto a line, then find a decision threshold along that line. The best projection is the one which maximizes the separation between the two groups.

# Complex Decision Boundary



# Mixture Bayes Classifier

How can we do better than the simple Gaussian for the class-conditional densities?

We can model each class with a mixture of Gaussians. Let's call this the *mixture Bayes classifier*. Done by estimating each mixture separately.

# Nonparametric Bayes Classifier

Finally we can model each class with a kernel density estimate. Let's call this the *nonparametric Bayes classifier*, sometimes called *kernel discriminant analysis*. Classically done by estimating each density separately. Also easy to do this discriminatively.

**Task:** classification

**Model class:** Sobolev (nonparametric)

**Loss:**  $L_2$  error (generative), or 0-1 loss (discriminative)

**Optimizer:** exhaustive or gradient descent

**Generalization mechanism:** cross-validation

**Evaluation algorithm:** generalized  $N$ -body algorithm

# Classification: Discriminative Approaches

Can I do classification while avoiding density estimation?

# Discrimination as Regression

Let  $Y \in \{0, 1\}$ . Recall the discriminant function, or regression function:

$$g(x) = \mathbb{E}(Y|X = x) \quad (38)$$

$$= \mathbb{P}(Y = 1|X = x) \quad (39)$$

$$= \frac{\pi_1 f_1(x)}{\pi_1 f_1(x) + \pi_0 f_0(x)}. \quad (40)$$

We'll now directly model  $g(x)$ , and use the rule

$$c(x) = \begin{cases} 1 & \text{if } g(x) > 1/2 \\ 0 & \text{otherwise} \end{cases} \quad (41)$$

This is equivalent to modeling the decision boundary.

# Linear Regression

Let's try a linear model for  $g(x)$ :

$$Y = g(x) + \epsilon = \sum_d \beta_d X_d + \epsilon \quad (42)$$

where  $\mathbb{E}(\epsilon) = 0$ .

We can treat the targets  $Y \in \{0, 1\}$  as continuous and use linear regression, minimizing the squared error. This works but in the classification setting is unnecessarily non-robust.

# Logistic Regression

Let's now come up with a different parametric model which assumes the targets are discrete:

$$\mathbb{P}(Y = 1|X = x) = \frac{e^{\sum_d \beta_d x_d}}{1 + e^{\sum_d \beta_d x_d}} = p(\beta) = p \quad (43)$$

where we define

$$\text{logit}(p_i) = \log \left( \frac{p_i}{1 - p_i} \right) = \sum_d \beta_d x_{id}. \quad (44)$$

This is called *logistic regression* because the function  $e^x / (1 + e^x)$  is called the logistic function. Its name is due to its roots in regression, even though it is a method for classification.

# Logistic Regression

Because  $Y$  is binary, it can be modeled as Bernoulli:

$$Y_i | X_i = x_i \sim \text{Bernoulli}(p_i), \quad (45)$$

which has (conditional) likelihood function

$$L(\beta) = \prod_{i=1}^N p_i(\beta)^{Y_i} (1 - p_i(\beta))^{1-Y_i}. \quad (46)$$

# Logistic Regression

It turns out that logistic regression and LDA are using the same model. In LDA,

$$\log \left( \frac{\mathbb{P}(Y = 1|X = x)}{\mathbb{P}(Y = 0|X = x)} \right) = -\frac{1}{2}(\mu_0 + \mu_1)^T \Sigma^{-1} (\mu_1 - \mu_0) \quad (47)$$

$$\log \left( \frac{\pi_0}{\pi_1} \right) + x^T \Sigma^{-1} (\mu_1 - \mu_0) \quad (48)$$

$$= \alpha_0 + \alpha^T x. \quad (49)$$

In logistic regression the model is by assumption

$$\log \left( \frac{\mathbb{P}(Y = 1|X = x)}{\mathbb{P}(Y = 0|X = x)} \right) = \beta_0 + \beta^T x. \quad (50)$$

The difference is how they estimate parameters.

# Logistic Regression

In LDA we estimated the whole joint density of an observation  $f(x, y) = f(x|y)f(y) = f(y|x)f(x)$  by effectively maximizing the mixture likelihood

$$\prod_i f(x_i, y_i) = \underbrace{\prod_i f(x_i|y_i)}_{\text{Gaussian}} \underbrace{f(y_i)}_{\text{Bernoulli}} . \quad (51)$$

In logistic regression we maximize only the conditional likelihood, leaving the marginal term unspecified:

$$\prod_i f(x_i, y_i) = \underbrace{\prod_i f(y_i|x_i)}_{\text{Bernoulli(Logistic)}} f(x_i). \quad (52)$$

# Logistic Regression

So logistic regression is “less parametric” than LDA. But it is still overall parametric because it cannot model any possible decision boundary.

**Task:** regression

**Model class:** all possible logistic regressors (parametric)

**Loss:** likelihood (conditional)

**Optimizer:** unconstrained optimization: iterative reweighted least squares, conjugate gradient

**Generalization mechanism:** none