

CSE 6740 Lecture 6

How Do I Predict a Discrete Variable? (Classification)

Alexander Gray

agray@cc.gatech.edu

Georgia Institute of Technology

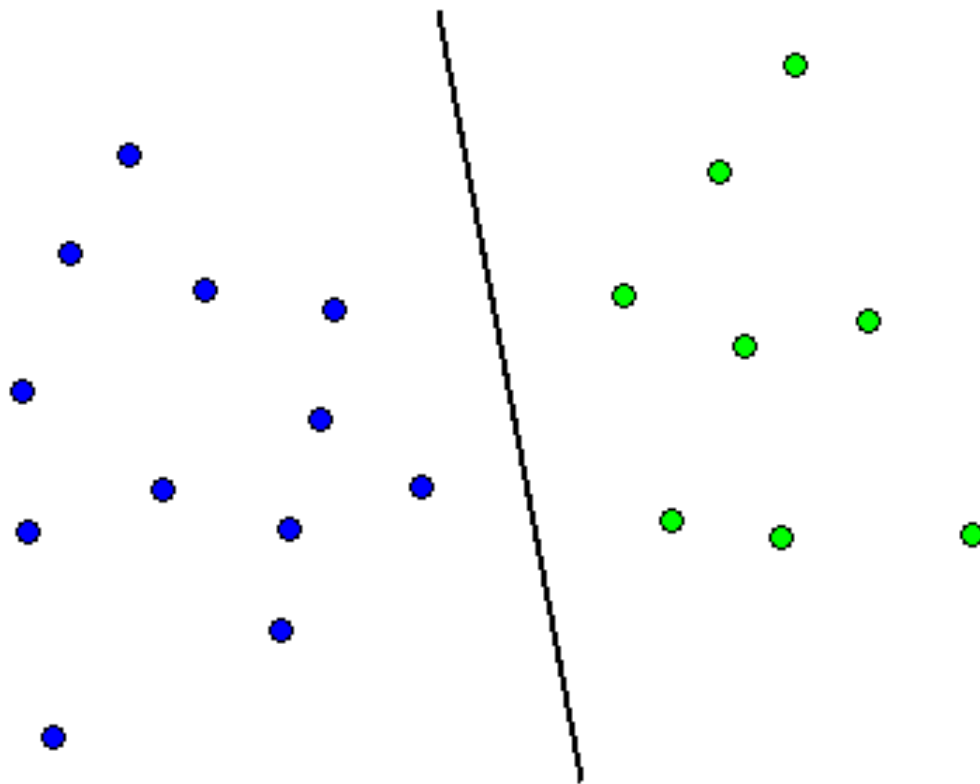
Today

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

Classification

How can I predict a discrete variable?

Classification



Classification Loss

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

$$L(Y, \hat{c}(X)) = I(Y \neq \hat{c}(X)). \quad (1)$$

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

$$L(Y, \hat{c}(X)) = C_{ab} \quad (2)$$

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

Classification Loss

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

$$E = \mathbb{E} [L(Y, \hat{c}(X))] \quad (3)$$

$$= \mathbb{E}_{X,Y} [L(Y, \hat{c}(X))] \quad (4)$$

$$= \mathbb{E}_X \mathbb{E}_{Y|X} [L(Y, \hat{c}(X))] \quad (5)$$

or, for a given x ,

$$E(x) = \mathbb{E}_{Y|X} [L(Y, \hat{c}(x))]. \quad (6)$$

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*:

$$g(x) = \mathbb{E}(Y|X = x) \quad (7)$$

$$= \int y f(y|x) dy \quad (8)$$

$$= 1 \cdot \mathbb{P}(Y = 1|X = x) + 0 \cdot \mathbb{P}(Y = 0|X = x) \quad (9)$$

$$= \mathbb{P}(Y = 1|X = x) \quad (10)$$

$$= \frac{f(x|Y = 1)\mathbb{P}(Y = 1)}{f(x|Y = 1)\mathbb{P}(Y = 1) + f(x|Y = 0)\mathbb{P}(Y = 0)} \quad (11)$$

$$= \frac{\pi_1 f_1(x)}{\pi_1 f_1(x) + \pi_0 f_0(x)}. \quad (12)$$

Bayes Classifier

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

$$c(x) = \begin{cases} 1 & \text{if } g(x) > 1/2 \\ 0 & \text{otherwise} \end{cases} \quad (13)$$

$$= \begin{cases} 1 & \text{if } \mathbb{P}(Y = 1|X = x) > \mathbb{P}(Y = 0|X = x) \\ 0 & \text{otherwise} \end{cases} \quad (14)$$

$$= \begin{cases} 1 & \text{if } \pi_1 f_1(x) > \pi_0 f_0(x) \\ 0 & \text{otherwise.} \end{cases} \quad (15)$$

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

$$\{x : \mathbb{P}(Y = 1|X = x) = \mathbb{P}(Y = 0|X = x)\} \quad (16)$$

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?

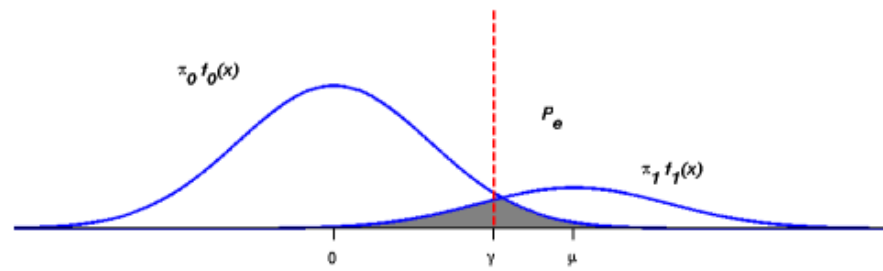
Bayes Classifier: Gaussian

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

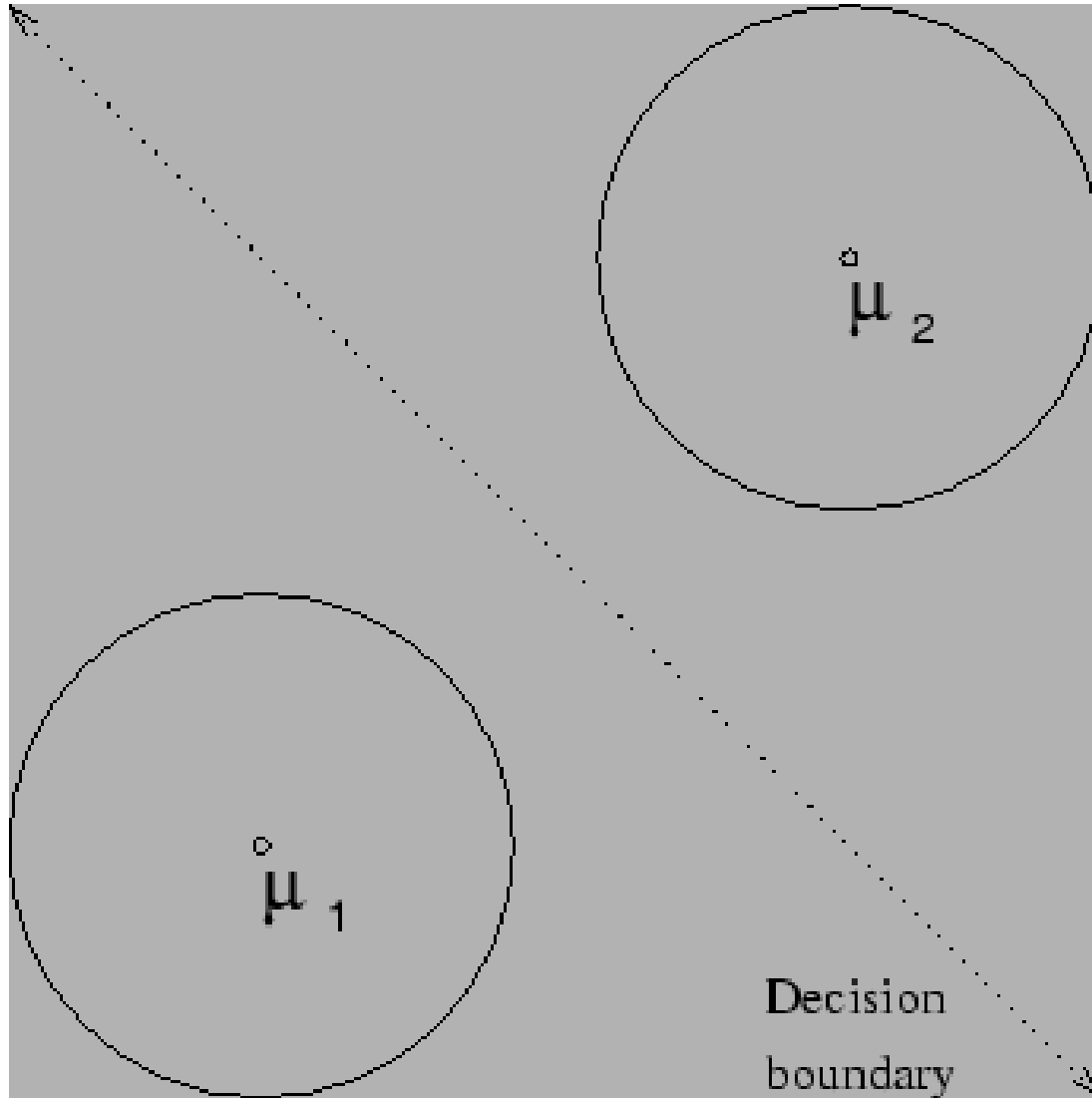
$$f_k(x) = \frac{1}{|\Sigma_k|^{1/2} (2\pi)^{D/2}} \exp \left\{ -\frac{1}{2} (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) \right\}, \quad (17)$$

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

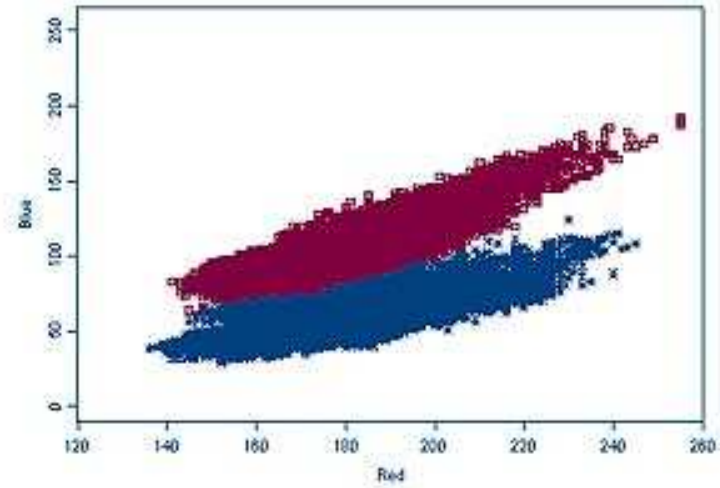
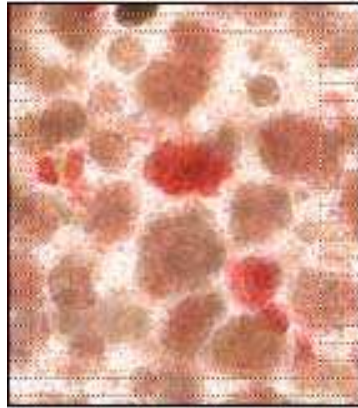
Bayes Classifier



Bayes Classifier



Bayes Classifier



Bayes Classifier: Gaussian

The Bayes classifier is then

$$c(x) = \arg \max_k \left\{ -\frac{1}{2} \log |\Sigma_k| - \frac{1}{2} (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) + \log \pi_k \right\} \quad (18)$$

or equivalently

$$c(x) = \begin{cases} 1 & \text{if } m_1^2 < m_0^2 + 2 \log \left(\frac{\pi_1}{\pi_0} \right) + \left(\frac{|\Sigma_1|}{|\Sigma_0|} \right) \\ 0 & \text{otherwise,} \end{cases} \quad (19)$$

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

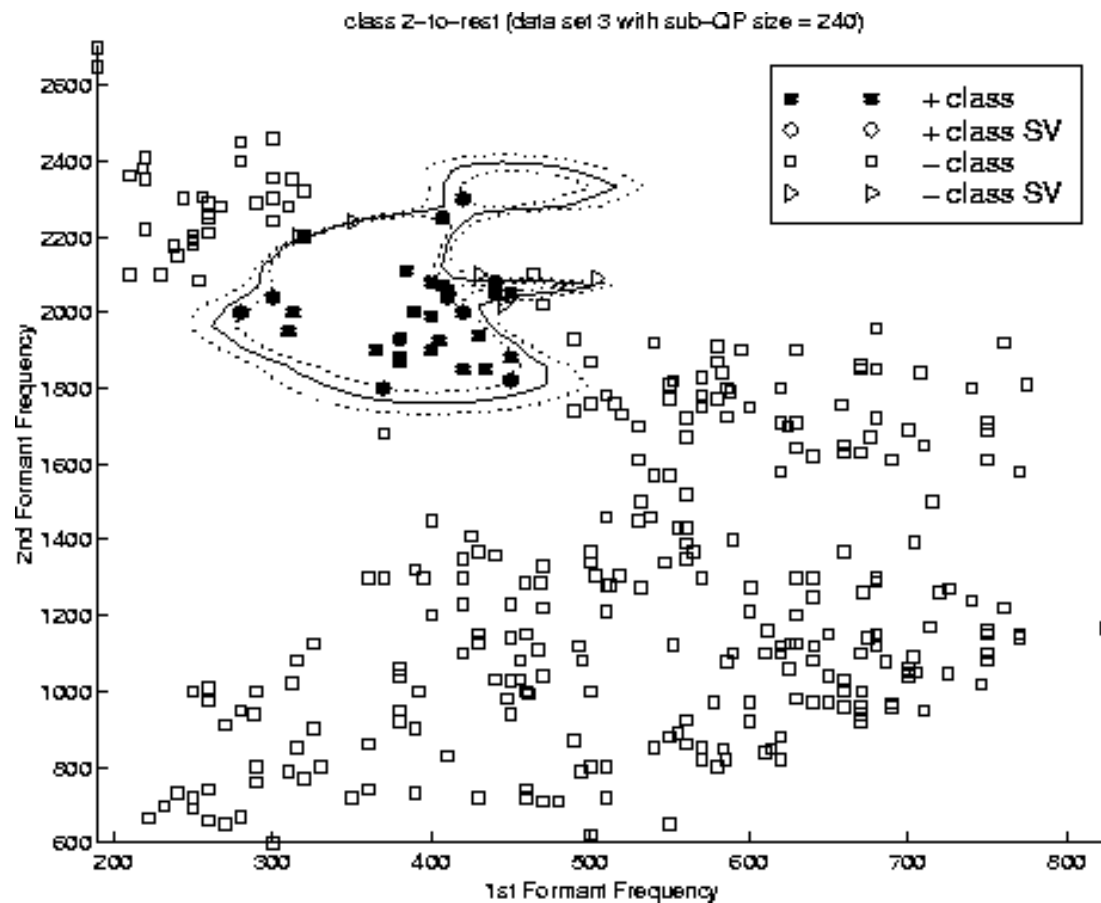
In the case where we assume that $\Sigma_0 = \Sigma_1 = \Sigma$ (but the covariance matrix may still be arbitrary), the Bayes classifier is

$$c(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k. \quad (20)$$

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:

$$g(x) = \mathbb{E}(Y|X = x) \quad (21)$$

$$= \mathbb{P}(Y = 1|X = x) \quad (22)$$

$$= \frac{\pi_1 f_1(x)}{\pi_1 f_1(x) + \pi_0 f_0(x)}. \quad (23)$$

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

$$c(x) = \begin{cases} 1 & \text{if } g(x) > 1/2 \\ 0 & \text{otherwise} \end{cases} \quad (24)$$

This is equivalent to modeling the decision boundary.

Linear Regression

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

$$Y = g(x) + \epsilon = \sum_d \beta_d X_d + \epsilon \quad (25)$$

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:

$$\mathbb{P}(Y = 1|X = x) = \frac{e^{\sum_d \beta_d x_d}}{1 + e^{\sum_d \beta_d x_d}} = p(\beta) = p \quad (26)$$

where we define

$$\text{logit}(p_i) = \log \left(\frac{p_i}{1 - p_i} \right) = \sum_d \beta_d x_{id}. \quad (27)$$

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:

$$Y_i | X_i = x_i \sim \text{Bernoulli}(p_i), \quad (28)$$

which has (conditional) likelihood function

$$L(\beta) = \prod_{i=1}^N p_i(\beta)^{Y_i} (1 - p_i(\beta))^{1 - Y_i}. \quad (29)$$

Logistic Regression

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

$$\log \left(\frac{\mathbb{P}(Y = 1|X = x)}{\mathbb{P}(Y = 0|X = x)} \right) = -\frac{1}{2}(\mu_0 + \mu_1)^T \Sigma^{-1}(\mu_1 - \mu_0) \quad (30)$$

$$\begin{aligned} & \log \left(\frac{\pi_0}{\pi_1} \right) + x^T \Sigma^{-1}(\mu_1 - \mu_0) \quad (31) \\ & = \alpha_0 + \alpha^T x. \quad (32) \end{aligned}$$

In logistic regression the model is by assumption

$$\log \left(\frac{\mathbb{P}(Y = 1|X = x)}{\mathbb{P}(Y = 0|X = x)} \right) = \beta_0 + \beta^T x. \quad (33)$$

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

$$\prod_i f(x_i, y_i) = \underbrace{\prod_i f(x_i|y_i)}_{\text{Bernoulli Gaussian}} \underbrace{f(y_i)}_{\text{Bernoulli}} . \quad (34)$$

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

$$\prod_i f(x_i, y_i) = \underbrace{\prod_i f(y_i|x_i)}_{\text{Bernoulli(Logistic)}} f(x_i). \quad (35)$$

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: classification

Model class: all possible logistic regressors (parametric)

Loss: likelihood (conditional)

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

Generalization mechanism: none

Perceptron

Here's another method for finding a linear decision boundary, which originally came from EE, called the *perceptron* or *linear machine*.

Now let $Y \in \{-1, 1\}$. Let's find the hyperplane defined by

$$g(x) = \beta_0 + \beta^T x = 0 \quad (36)$$

such that its distance to all misclassified points is minimized.

The (signed) distance of a point x to the hyperplane is given by $\frac{1}{\|\beta\|}(\beta_0 + \beta^T x)$, where $\|\beta\| = \sum_j \beta_j^2$, and our classification rule is

$$c(x) = \text{sgn}(\beta_0 + \beta^T x) = \text{sgn}g(x). \quad (37)$$

Perceptron

If a target $y_i = 1$ is misclassified, then $\beta_0 + \beta^T x < 0$ and vice versa. The goal is to minimize

$$-\sum_{i \in \mathcal{M}} y_i (\beta_0 + \beta^T x) \quad (38)$$

where \mathcal{M} is the set of indices of misclassified points.

We can make a gradient descent method for optimizing the parameters using the gradient of this with respect to β ,

$-\sum_{i \in \mathcal{M}} y_i x_i$, and its gradient with respect to β_0 , $-\sum_{i \in \mathcal{M}} y_i$.

We update the last value of the parameters by adding the gradient times some stepsize.

Perceptron

The original optimization method used to find the parameters was *stochastic gradient descent*, which is online (adjusts the parameters taking one data point at a time), has a stepsize parameter, and is slow.

When the classes are *linearly separable*, *i.e.* you can find a hyperplane which perfectly separates them, there are many solutions, and which one is found depends on the starting values of the parameters. This indeterminacy is a source of variance.

Neural Network

Neural networks are models of the form

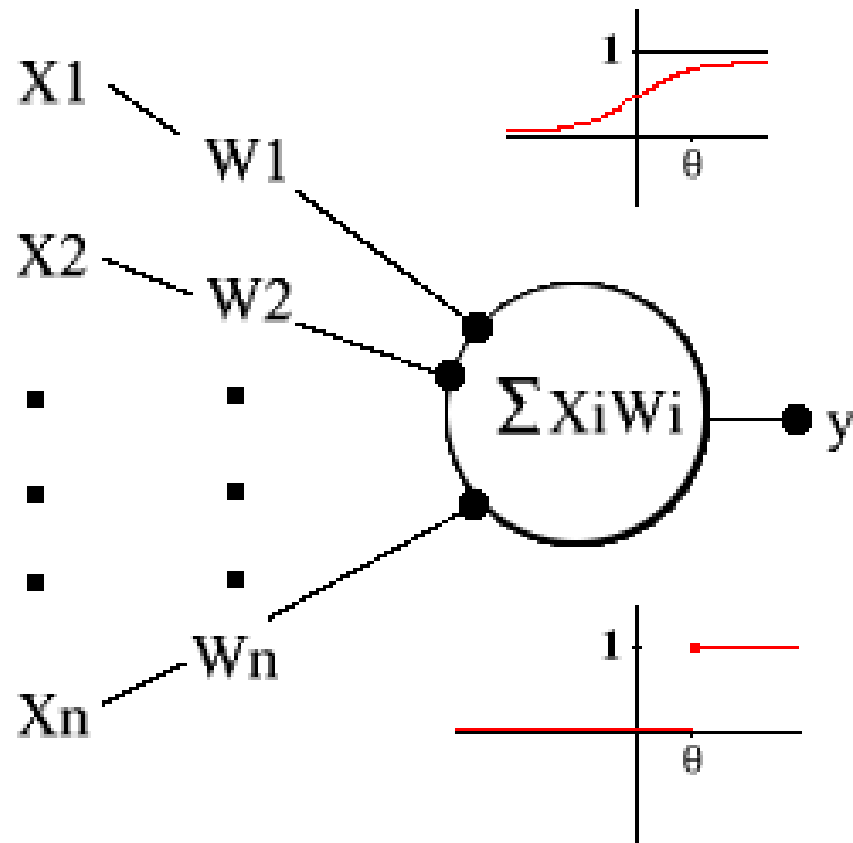
$$Y = \beta_0 + \sum_j \beta_j \sigma(\alpha_0 + \alpha^T X) \quad (39)$$

where σ is a smooth *sigmoidal* function, often having the form

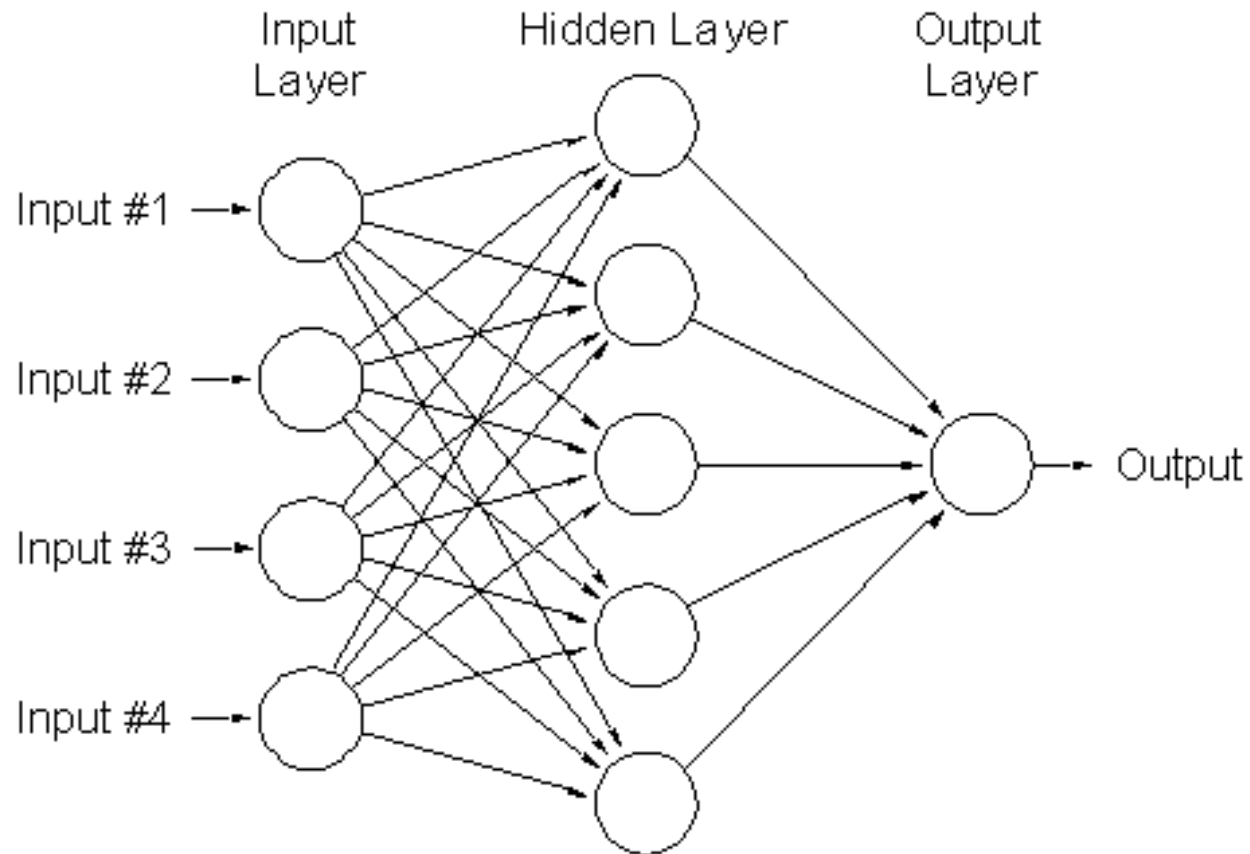
$$\sigma(t) = \frac{e^t}{1 + e^t}. \quad (40)$$

You can represent this as a graph having multiple layers of nodes. You can also seamlessly train for multiple targets at the same time.

Neural Network



Neural Network



Neural Network

This was traditionally optimized using the same method we saw for the perceptron. It was a big deal when various people figured out how to compute the derivative in a model where there are multiple layers, leading to what was called *back-propagation*.

Neural Network

There is also a regression version, which just uses a different loss function.

Task: Regression or classification.

Model class: Nonparametric (varying number of units).

Loss: Squared error (regression case), 0-1 loss (classification case).

Optimizer: Conjugate gradient or other unconstrained optimization, including stochastic gradient descent (called “back-propagation”).

Generalization mechanism: Cross-validation over regularization parameter (sum of squared weights, called “weight decay”).

Main Things You Should Know

- What generative vs. discriminative classification is
- What a Bayes classifier is
- What a perceptron is
- What a neural network is

Sample Final Questions

1. (T/F) Learning a 2-class generative classifier boils down to learning two separate density estimators.
2. (T/F) The perceptron is a generative classifier.
3. (T/F) The perceptron is a nonparametric classifier.