Multi-class classification with:

- Least-squares regression
- Logistic Regression
- K-NN
- Decision trees
Discriminant Functions for $K > 2$ classes

- Use $K - 1$ classifiers, each solving a two class problem of separating point in a class $C_k$ from points not in the class.
- Known as 1 vs all or 1 vs the rest classifier

PROBLEM: More than one good answer!
Discriminant Functions for $K > 2$ classes

- Introduce $K(K - 1)/2$ two-way classifiers, one for each possible pair of classes.
- Each point is classified according to majority vote amongst the disc. func.
- Known as the **1 vs 1 classifier**

**PROBLEM:** Two-way preferences need not be transitive
We can avoid these problems by considering a single K-class discriminant comprising \( K \) functions of the form

\[
y_k(x) = \mathbf{w}_k^T \mathbf{x} + w_{k,0}
\]

and then assigning a point \( \mathbf{x} \) to class \( C_k \) if

\[
\forall j \neq k \quad y_k(x) > y_j(x)
\]

Note that \( \mathbf{w}_k^T \) is now a vector, not the \( k \)-th coordinate.

The decision boundary between class \( C_j \) and class \( C_k \) is given by \( y_j(x) = y_k(x) \), and thus it’s a \((D - 1)\) dimensional hyperplane defined as

\[
(\mathbf{w}_k - \mathbf{w}_j)^T \mathbf{x} + (w_{k0} - w_{j0}) = 0
\]

What about the binary case? Is this different?

What is the shape of the overall decision boundary?
The decision regions of such a discriminant are always \textit{singly connected} and \textit{convex}.

In Euclidean space, an object is \textit{convex} if for every pair of points within the object, every point on the straight line segment that joins the pair of points is also within the object.

Which object is convex?
The decision regions of such a discriminant are always **singly connected** and **convex**.

Consider 2 points $x_A$ and $x_B$ that lie inside decision region $R_k$.

Any convex combination $\hat{x}$ of those points also will be in $R_k$.

\[ \hat{x} = \lambda x_A + (1 - \lambda) x_B \]
Proof

- A convex combination point, i.e., $\lambda \in [0, 1]$
  \[ \hat{x} = \lambda x_A + (1 - \lambda)x_B \]

- From the linearity of the classifier $y(x)$
  \[ y_k(\hat{x}) = \lambda y_k(x_A) + (1 - \lambda)y_k(x_B) \]

- Since $x_A$ and $x_B$ are in $R_k$, it follows that $y_k(x_A) > y_j(x_A)$, $y_k(x_B) > y_j(x_B)$, $\forall j \neq k$
- Since $\lambda$ and $1 - \lambda$ are positive, then $\hat{x}$ is inside $R_k$
- Thus $R_k$ is singly connected and convex
Multi-class classification via the "softmax"

- Associate a set of weights with each class, then use a normalized exponential output

\[ p(C_k|x) = y_k(x) = \frac{\exp(z_k)}{\sum_j \exp(z_j)} \]

where the activations are given by

\[ z_k = w_k^T x \]

- For the target vector, if there are \( K \) classes we often use a 1-of-\( K \) encoding, i.e., a vector of \( K \) target values containing a single 1 for the correct class and zeros elsewhere

- Let \( T \in \{0, 1\}^{N \times K} \) for \( N \) training examples and \( K \) classes
Multi-class Logistic Regression

- The likelihood

\[ p(T|\mathbf{w}_1, \cdots, \mathbf{w}_K) = \prod_{n=1}^{N} \prod_{k=1}^{K} p(C_k|x^{(n)})^{t_k^{(n)}} = \prod_{n=1}^{N} \prod_{k=1}^{K} y_k^{(n)}(x^{(n)})^{t_k^{(n)}} \]

with

\[ p(C_k|x) = y_k(x) = \frac{\exp(z_k)}{\sum_j \exp(z_j)} \]

and

\[ z_k = \mathbf{w}_k^T \mathbf{x} + w_{k0} \]

- What assumptions have I used to derive the likelihood?
- Derive the loss by computing the negative log-likelihood

\[ E(\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\log p(T|\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_k^{(n)} \log[y_k^{(n)}(x^{(n)})] \]

- This is known as the **cross-entropy** error for multiclass classification
- How do we obtain the weights?
Training Multi-class Logistic Regression

\[ E(\mathbf{w}_1, \cdots, \mathbf{w}_K) = -\log p(\mathbf{T}|\mathbf{w}_1, \cdots, \mathbf{w}_K) = - \sum_{n=1}^{N} \sum_{k=1}^{K} t_k^{(n)} \log[y_k^{(n)}(\mathbf{x}^{(n)})] \]

- Do gradient descent, where the derivatives are

\[ \frac{\partial y_j^{(n)}}{\partial z_k^{(n)}} = \delta(k, j)y_j^{(n)} - y_j^{(n)}y_k^{(n)} \]

and

\[ \frac{\partial E}{\partial z_k^{(n)}} = \sum_{j=1}^{K} \frac{\partial E}{\partial y_j^{(n)}} \cdot \frac{\partial y_j^{(n)}}{\partial z_k^{(n)}} = y_k^{(n)} - t_k^{(n)} \]

\[ \frac{\partial E}{\partial w_{k,j}} = \sum_{n=1}^{N} \sum_{j=1}^{K} \frac{\partial E}{\partial y_j^{(n)}} \cdot \frac{\partial y_j^{(n)}}{\partial z_k^{(n)}} \cdot \frac{\partial z_k^{(n)}}{\partial w_{k,j}} = \sum_{n=1}^{N} (y_k^{(n)} - t_k^{(n)}) \cdot x_j^{(n)} \]

- The derivative is the error times the input
Softmax for 2 Classes

- Let’s write the probability of one of the classes

\[ p(C_1 | x) = y_1(x) = \frac{\exp(z_1)}{\sum_j \exp(z_j)} = \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2)} \]

- We can equivalently write this as

\[ p(C_1 | x) = y_1(x) = \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2)} = \frac{1}{1 + \exp(-(z_1 - z_2))} \]

- So the logistic is just a special case that avoids using redundant parameters

- Rather than having two separate sets of weights for the two classes, combine into one

\[ z' = z_1 - z_2 = w_1^T x - w_2^T x = w^T x \]

- The over-parameterization of the softmax is because the probabilities must add to 1.
Multi-class K-NN

- Can directly handle multi class problems
Multi-class Decision Trees

- Can directly handle multi class problems
- How is this decision tree constructed?
Today (Part 2)

- Classification – Bayes classifier
- Estimate probability densities from data
- Making decisions: Risk
Generative vs Discriminative

Two approaches to classification:

- **Discriminative** classifiers estimate parameters of decision boundary/class separator directly from labeled sample
  - learn boundary parameters directly (logistic regression models $p(t_k|x)$)
  - learn mappings from inputs to classes (least-squares, neural nets)

- **Generative approach**: model the distribution of inputs characteristic of the class (Bayes classifier)
  - Build a model of $p(x|t_k)$
  - Apply Bayes Rule
Bayes Classifier

- Aim to diagnose whether patient has diabetes: classify into one of two classes (yes $C=1$; no $C=0$)
- Run battery of tests
- Given patient’s results: $\mathbf{x} = [x_1, x_2, \cdots, x_d]^T$ we want to update class probabilities using Bayes Rule:

$$p(C|\mathbf{x}) = \frac{p(\mathbf{x}|C)p(C)}{p(\mathbf{x})}$$

- More formally

$$\text{posterior} = \frac{\text{Class likelihood } \times \text{ prior}}{\text{Evidence}}$$

- How can we compute $p(\mathbf{x})$ for the two class case?

$$p(\mathbf{x}) = p(\mathbf{x}|C = 0)p(C = 0) + p(\mathbf{x}|C = 1)p(C = 1)$$
Start with single input/observation per patient: white blood cell count

\[ p(C = 1|x = 50) = \frac{p(x = 50|C = 1)p(C = 1)}{p(x = 50)} \]

- Need class-likelihoods, priors
- **Prior**: In the absence of any observation, what do I know about the problem?
- What would you use as prior?
Question: Which probability distribution makes sense for $p(x|C)$?
Let’s assume that the class-conditional densities are Gaussian

\[ p(x|C) = \frac{1}{\sqrt{2\pi\sigma}} \exp \left( \frac{(x - \mu)^2}{2\sigma^2} \right) \]

with \( \mu \in \mathbb{R} \) and \( \sigma^2 \in \mathbb{R}^+ \)

How can I fit a Gaussian distribution to my data?

Let’s try maximum likelihood estimation (MLE)

We are given a set of training examples \( \{x^{(n)}, y^{(n)}\}_{n=1,\ldots,N} \) with \( y^{(n)} \in \{0, 1\} \) and we want to estimate the model parameters \( \{\mu, \sigma\} \) for each class

First divide the training examples into two classes according to \( y^{(n)} \), and for each class take all the examples and fit a Gaussian to model \( p(x|C) \)
We assume that the data points that we have are independent and identically distributed

\[ p(x^{(1)}, \ldots, x^{(N)}|C) = \prod_{n=1}^{N} p(x^{(n)}|C) = \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp \left( -\frac{(x^{(n)} - \mu)^2}{2\sigma^2} \right) \]

Now we want to maximize the likelihood, or minimize its negative (if you think in terms of a loss)

\[ \ell_{\text{log-\text{loss}}} = -\ln p(x^{(1)}, \ldots, x^{(N)}|C) = -\ln \left( \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp \left( -\frac{(x^{(n)} - \mu)^2}{2\sigma^2} \right) \right) \]

\[ = \sum_{n=1}^{N} \ln(\sqrt{2\pi}\sigma) + \sum_{n=1}^{N} \frac{(x^{(n)} - \mu)^2}{2\sigma^2} = \frac{N}{2} \ln \left( 2\pi\sigma^2 \right) + \sum_{n=1}^{N} \frac{(x^{(n)} - \mu)^2}{2\sigma^2} \]

How would you do we minimize the function?

Write \( \frac{d\ell_{\text{log-\text{loss}}}}{d\mu} \) and \( \frac{d\ell_{\text{log-\text{loss}}}}{d\sigma^2} \) and equal it to 0 to find the parameters \( \mu \) and \( \sigma^2 \).
Computing the Mean

\[
\frac{\partial \ell_{\text{log-loss}}}{\partial \mu} = \frac{\partial}{\partial \mu} \left( \frac{N}{2} \ln (2\pi \sigma^2) + \sum_{n=1}^{N} \frac{(x^{(n)} - \mu)^2}{2\sigma^2} \right) = \frac{\partial}{\partial \mu} \left( \sum_{n=1}^{N} \frac{(x^{(n)} - \mu)^2}{2\sigma^2} \right) = \frac{d}{d\mu} \left( \sum_{n=1}^{N} \frac{(x^{(n)} - \mu)^2}{2\sigma^2} \right)
\]

\[
= - \sum_{n=1}^{N} \frac{2(x^{(n)} - \mu)}{2\sigma^2} = - \sum_{n=1}^{N} \frac{(x^{(n)} - \mu)}{\sigma^2} = \frac{N\mu - \sum_{n=1}^{N} x^{(n)}}{\sigma^2}
\]

And equating to zero we have

\[
\frac{d\ell_{\text{log-loss}}}{d\mu} = 0 = \frac{N\mu - \sum_{n=1}^{N} x^{(n)}}{\sigma^2}
\]

Thus

\[
\mu = \frac{1}{N} \sum_{n=1}^{N} x^{(n)}
\]
Computing the Variance

\[
\frac{d\ell_{\log\text{-loss}}}{d\sigma^2} = d \left( \frac{N}{2} \ln (2\pi\sigma^2) + \sum_{n=1}^{N} \frac{(x^{(n)} - \mu)^2}{2\sigma^2} \right)
\]

\[
= \frac{N}{2} \frac{1}{2\pi\sigma^2} 2\pi + \sum_{n=1}^{N} \frac{(x^{(n)} - \mu)^2}{2} \left(-\frac{1}{\sigma^4}\right)
\]

\[
= \frac{N}{2\sigma^2} - \sum_{n=1}^{N} \frac{(x^{(n)} - \mu)^2}{2\sigma^4}
\]

And equating to zero we have

\[
\frac{d\ell_{\log\text{-loss}}}{d\sigma^2} = 0 = \frac{N}{2\sigma^2} - \sum_{n=1}^{N} \frac{(x^{(n)} - \mu)^2}{2\sigma^4} = N\sigma^2 - \sum_{n=1}^{N} \frac{(x^{(n)} - \mu)^2}{2\sigma^4}
\]

Thus

\[
\sigma^2 = \frac{1}{N} \sum_{n=1}^{N} (x^{(n)} - \mu)^2
\]
**MLE of a Gaussian**

- We can compute the parameters in closed form for each class by taking the training points that belong to that class.

\[
\mu = \frac{1}{N} \sum_{n=1}^{N} x^{(n)} \\
\sigma^2 = \frac{1}{N} \sum_{n=1}^{N} (x^{(n)} - \mu)^2
\]
Given a new observation, the estimated class-likelihoods and the prior, we can obtain the **posterior probability** for class $C = 1$

$$p(C = 1|x) = \frac{p(x|C = 1)p(C = 1)}{p(x)}$$

$$= \frac{p(x|C = 1)p(C = 1)}{p(x|C = 0)p(C = 0) + p(x|C = 1)p(C = 1)}$$

Let’s see an example
Doctor has a prior $p(C = 0) = 0.8$ (how?)

Example $x = 50$, $p(x = 50|C = 0) = 0.11$, and $p(x = 50|C = 1) = 0.42$

How were $p(x = 50|C = 0)$ and $p(x = 50|C = 1)$ computed?

How can I compute $p(C = 1)$?

Which class is more likely? Do I have diabetes?
Bayes Classifier

- Use Bayes classifier to classify new patients (unseen test examples)
- Simple Bayes classifier: estimate posterior probability of each class
- What should the decision criterion be?
- The optimal decision is the one that minimizes the expected number of mistakes
Conditional risk of a classifier

\[ R(y|x) = \sum_{c=1}^{C} L(y, t) p(t = c|x) \]

\[ = 0 \cdot p(t = y|x) + 1 \cdot \sum_{c \neq y} p(t = c|x) \]

\[ = \sum_{c \neq y} p(t = c|x) = 1 - p(t = y|x) \]

- To minimize conditional risk given \( x \), the classifier must decide

\[ y = \arg \max_c p(t = c|x) \]

- This is the best possible classifier in terms of generalization, i.e. expected misclassification rate on new examples.
Optimal rule $y = \arg \max_c p(t = c|x)$ is equivalent to

$$y = c \iff \frac{p(t = c|x)}{p(t = j|x)} \geq 1 \quad \forall j \neq c$$

$$\iff \log \frac{p(t = c|x)}{p(t = j|x)} \geq 0 \quad \forall j \neq c$$

For the binary case

$$y = 1 \iff \log \frac{p(t = 1|x)}{p(t = 0|x)} \geq 0$$

Where have we used this rule before?
The Bayes classifier will construct a decision boundary: used to classify new patients (unseen test examples)

Can be viewed as a simple linear classifier

\[ C = \begin{cases} 
1 & \text{if } x \geq T \\
0 & \text{otherwise} 
\end{cases} \]
Classification – Multi-dimensional Bayes classifier

Estimate probability densities from data

Naive Bayes
Two approaches to classification:

- **Generative approach**: model the distribution of inputs characteristic of the class (Bayes classifier)
  - Build a model of $p(x|t_k)$
  - Apply Bayes Rule

- **Discriminative** classifiers estimate parameters of decision boundary/class separator directly from labeled sample
  - learn boundary parameters directly (logistic regression)
  - learn mappings from inputs to classes (least-squares, neural nets)
Bayes Classifier

- Aim to diagnose whether patient has diabetes: classify into one of two classes (yes C=1; no C=0)
- Run battery of tests
- Given patient’s results: $\mathbf{x} = [x_1, x_2, \cdots, x_d]^T$ we want to update class probabilities using Bayes Rule:

$$p(C|\mathbf{x}) = \frac{p(\mathbf{x}|C)p(C)}{p(\mathbf{x})}$$

- More formally

$$\text{posterior} = \frac{\text{Class likelihood} \times \text{prior}}{\text{Evidence}}$$

- How can we compute $p(\mathbf{x})$ for the two class case?

$$p(\mathbf{x}) = p(\mathbf{x}|C = 0)p(C = 0) + p(\mathbf{x}|C = 1)p(C = 1)$$
Last class we had a single input/observation per patient: white blood cell count

\[ p(C = 1 | x = 50) = \frac{p(x = 50 | C = 1)p(C = 1)}{p(x = 50)} \]

Add second observation: Plasma glucose value

Can construct bivariate normal (Gaussian) distribution of each class
Gaussian Bayes Classifier

- Gaussian (or normal) distribution:

\[ p(x|t = k) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp \left[ -\frac{1}{2} (x - \mu_k)^T \Sigma^{-1} (x - \mu_k) \right] \]

- Each class \( k \) has associated mean vector, but typically the classes share a single covariance matrix
Multivariate Data

- Multiple measurements (sensors)
- \( d \) inputs/features/attributes
- \( N \) instances/observations/examples

\[
\mathbf{X} = \begin{bmatrix}
  x_1^{(1)} & x_2^{(1)} & \cdots & x_d^{(1)} \\
  x_1^{(2)} & x_2^{(2)} & \cdots & x_d^{(2)} \\
  \vdots & \vdots & \ddots & \vdots \\
  x_1^{(N)} & x_2^{(N)} & \cdots & x_d^{(N)}
\end{bmatrix}
\]
Multivariate Parameters

- **Mean**
  \[ \mathbf{E}[\mathbf{x}] = [\mu_1, \cdots, \mu_d]^T \]

- **Covariance**
  \[ \Sigma = \text{Cov}(\mathbf{x}) = \mathbf{E}[(\mathbf{x} - \mu)^T(\mathbf{x} - \mu)] = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1d} \\ \sigma_{12} & \sigma_2^2 & \cdots & \sigma_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{d1} & \sigma_{d2} & \cdots & \sigma_d^2 \end{bmatrix} \]

- **Correlation** \( \text{Corr}(\mathbf{x}) \) is the covariance divided by the product of standard deviation
  \[ \rho_{ij} = \frac{\sigma_{ij}}{\sigma_i \sigma_j} \]
Multivariate Gaussian Distribution

- \( x \sim \mathcal{N}(\mu, \Sigma) \), a Gaussian (or normal) distribution defined as
  
  \[
p(x) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp \left[ -\frac{(x - \mu_k)^T \Sigma^{-1} (x - \mu_k)}{2} \right]
  \]

- Mahalanobis distance \( (x - \mu_k)^T \Sigma^{-1} (x - \mu_k) \) measures the distance from \( x \) to \( \mu \) in terms of \( \Sigma \)
- It normalizes for difference in variances and correlations
Bivariate Normal

- $\text{Cov}(x_1, x_2) = 0$, $\text{Var}(x_1) = \text{Var}(x_2)$
- $\text{Cov}(x_1, x_2) = 0$, $\text{Var}(x_1) > \text{Var}(x_2)$
- $\text{Cov}(x_1, x_2) > 0$
- $\text{Cov}(x_1, x_2) < 0$
Bivariate Normal

\[ \text{Cov}(x_1, x_2) = 0, \ Var(x_1) = \Var(x_2) \]

\[ \text{Cov}(x_1, x_2) = 0, \ Var(x_1) > \Var(x_2) \]

\[ \text{Cov}(x_1, x_2) > 0 \]

\[ \text{Cov}(x_1, x_2) < 0 \]
GBC decision boundary: based on class posterior

Take the class which has higher posterior probability

\[
\log p(t_k | x) = \log p(x | t_k) + \log p(t_k) - \log p(x)
\]

\[
= -\frac{d}{2} \log(2\pi) - \frac{1}{2} \log |\Sigma_k| - \frac{1}{2}(x - \mu_k)^T \sigma_k^{-1}(x - \mu_k) + + \log p(t_k) - \log p(x)
\]

Decision: which class has higher posterior probability
Decision Boundary

\[ P(t_1 | x) = 0.5 \]

discriminant: \[ P(t_1 | x) = 0.5 \]

**likelihoods**

**posterior for** \( t_1 \)
Shared Covariance Matrix
Learning Gaussian Bayes Classifier

- Learn the parameters using maximum likelihood

\[
\ell(\phi, \mu_0, \mu_1, \Sigma) = - \log \prod_{n=1}^{N} p(x^{(n)}, t^{(n)}|\phi, \mu_0, \mu_1, \Sigma) \\
= - \log \prod_{n=1}^{N} p(x^{(n)}|t^{(n)}, \mu_0, \mu_1, \Sigma)p(t^{(n)}|\phi)
\]

- What have I assumed?
More on MLE

- Assume the prior is Bernoulli (we have two classes)

\[ p(t|\phi) = \phi^t(1 - \phi)^{1-t} \]

- You can compute the ML estimate in closed form

\[
\phi = \frac{1}{N} \sum_{n=1}^{N} \mathbf{1}[t^{(n)} = 1]
\]

\[
\mu_0 = \frac{\sum_{n=1}^{N} \mathbf{1}[t^{(n)} = 0] \cdot x^{(n)}}{\sum_{n=1}^{N} \mathbf{1}[t^{(n)} = 0]}
\]

\[
\mu_1 = \frac{\sum_{n=1}^{N} \mathbf{1}[t^{(n)} = 1] \cdot x^{(n)}}{\sum_{n=1}^{N} \mathbf{1}[t^{(n)} = 1]}
\]

\[
\Sigma = \frac{1}{N} \sum_{n=1}^{N} (x^{(n)} - \mu_{t^{(n)}})(x^{(n)} - \mu_{t^{(n)}})^T
\]
For Gaussian Bayes Classifier, if input \( x \) is high-dimensional, then covariance matrix has many parameters.

Save some parameters by using a shared covariance for the classes.

Naive Bayes is an alternative Generative model: assumes features independent given the class.

\[
p(x|t = k) = \prod_{i=1}^{d} p(x_i|t = k)
\]

How many parameters required now? And before?
Diagonal Covariance

variances may be different

variances may be different
Diagonal Covariance, isotropic

Classification only depends on distance to the mean
Naive Bayes Classifier

Given

- prior
- assuming features are conditionally independent given the class
- likelihood for each $x_i$

The decision rule

$$y = \arg \max_k p(t = k) \prod_{i=1}^{d} p(x_i | t = k)$$

- If the assumption of conditional independence holds, NB is the optimal classifier
- If not, a heavily regularized version of generative classifier
- What’s the regularization?
Gaussian Naive Bayes

- Assume

\[ p(x_i | t = k) = \frac{1}{\sqrt{2\pi \sigma_{ik}}} \exp \left[ \frac{-(x_i - \mu_{ik})^2}{2\sigma_{ik}^2} \right] \]

- Maximum likelihood estimate of parameters

\[ \mu_{ik} = \frac{\sum_{n=1}^{N} 1[t^{(n)} = k] \cdot x_i^{(n)}}{\sum_{n=1}^{N} 1[t^{(n)} = k]} \]

- Similar for the variance
If you examine $p(t = 1|x)$ under GBC, you will find that it looks like this:

$$p(t|x, \phi, \mu_0, \mu_1, \Sigma) = \frac{1}{1 + \exp(-w(\phi, \mu_0, \mu_1, \Sigma)^T x)}$$

So the decision boundary has the same form as logistic regression!

When should we prefer GBC to LR, and vice versa?
GBC vs LR

- GBC makes stronger modeling assumption: assumes class-conditional data is multivariate Gaussian
- If this is true, GBC is asymptotically efficient (best model in limit of large $N$)
- But LR is more robust, less sensitive to incorrect modeling assumptions
- Many class-conditional distributions lead to logistic classifier
- When these distributions are non-Gaussian, in limit of large $N$, LR beats GBC