syde675 pattern recognition

Feature Extraction: Linear Discriminant Analysis, Feature Selection, Cross Validation

J.Zelek 2022

Feature Extraction Methods

Feature extraction:

- These methods select d $(d < D)$ features out of D dimensions and discard $D - d$ dimensions.
  - Given a feature space $\boldsymbol{x} \in \boldsymbol{R^D}$ find a mapping $\boldsymbol{y} = f(\boldsymbol{x}) : \mathbb{R}^D \rightarrow \mathbb{R}^d$ with $d < D$ such that transformed feature vector $\boldsymbol{y} \in \mathbb{R}^d$ optimises an objective function

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \xrightarrow{\text{feature extraction}} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_d \end{bmatrix} = f\left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix}\right)$$

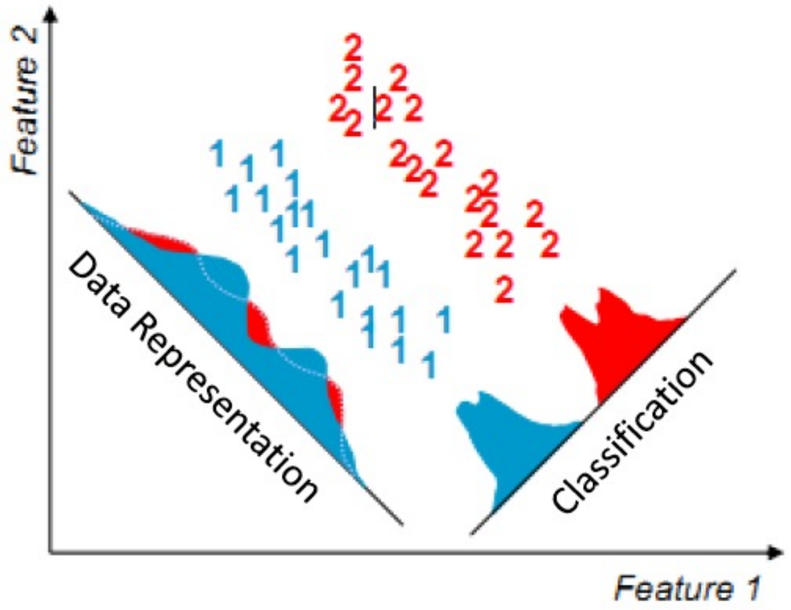
Feature extraction methods can be:

- Supervised / Unsupervised
- Linear / Non-Linear

Data Representation vs. Classification

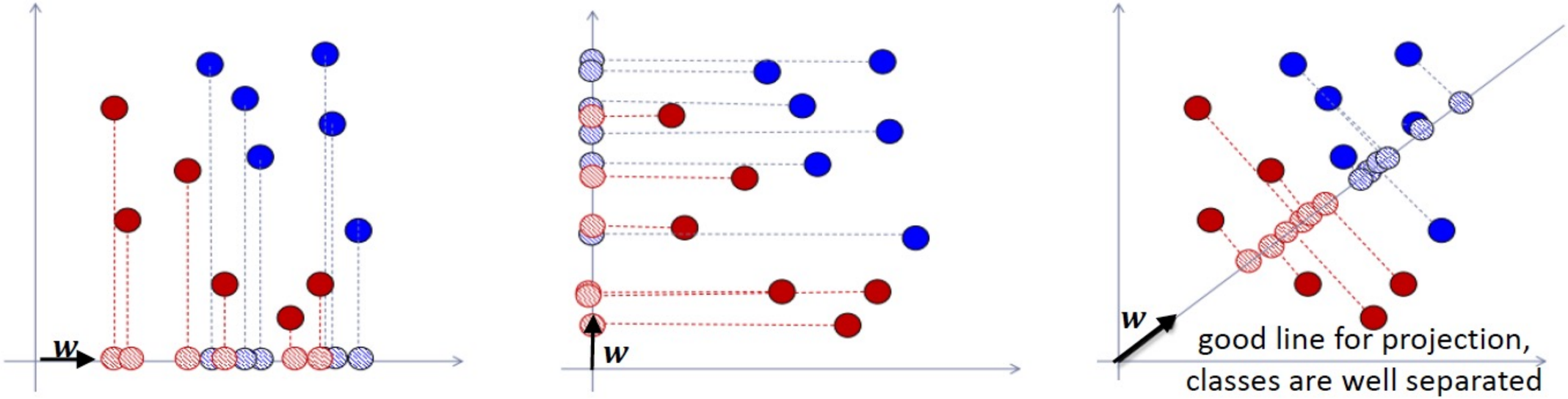
- **Data Representation:** The goal of feature extraction mapping is to **represent samples accurately** in a lower-dimensional space
  - Here, we study Principal Components Analysis (PCA) (previous lecture)

- **Classification:** The goal of feature extraction mapping is to **improve class-discriminatory** information in a lower-dimensional space
  - Here, we study Linear Discriminant Analysis (LDA) (current lecture)



LDA (Linear Discriminant Analysis)

- For the following 2D example, from all possible lines, LDA select the one that samples from different classes are well separated
- The objective of LDA is to perform dimensionality reduction while preserving as much of the class discriminatory information as possible



- **How measure the degree of separation in the projected subspace?**

Figure by [Soleymani]

LDA Intuition

- Suppose a 2-class and d-dim samples $x_1, x_2, \ldots, x_N$ where
  - N_k shows the number of samples of class k and $N_1 + N_2 = N$
- The projection of samples into a line direction is given by:

$$y = \boldsymbol{w}^T \boldsymbol{x}$$

- In order to find a good projection vector, we need to define a measure of separation between the projections

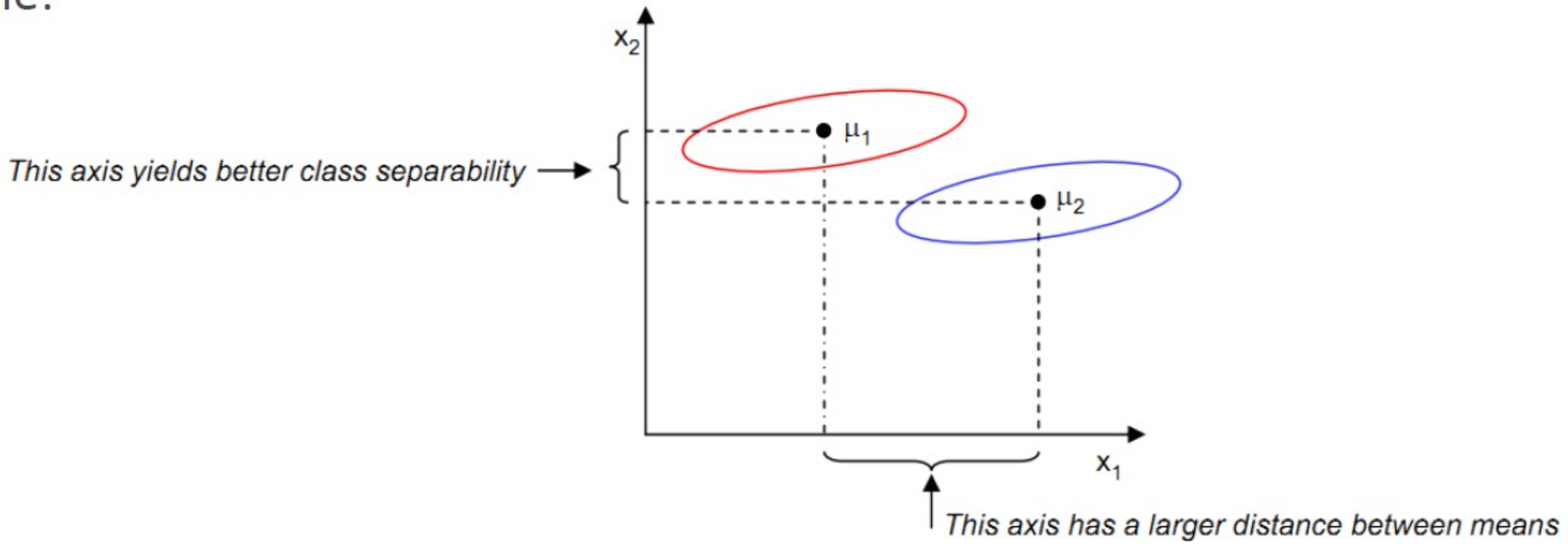
- One possible solution can be maximizing **"distance between mean vector of each class"**

$$\underset{\boldsymbol{w}}{\operatorname{argmax}}[J(\boldsymbol{w})] \qquad J(\boldsymbol{w}) = |\tilde{\mu}_1 - \tilde{\mu}_2| = |\boldsymbol{w}^T(\boldsymbol{\mu_1} - \boldsymbol{\mu_2})|$$

  - Is it a very good measure?

LDA Intuition

- The distance between the projected means is not a very good measure
- For example:



- It is not very good because it does not take into account the scatter of samples within the classes

LDA Intuition

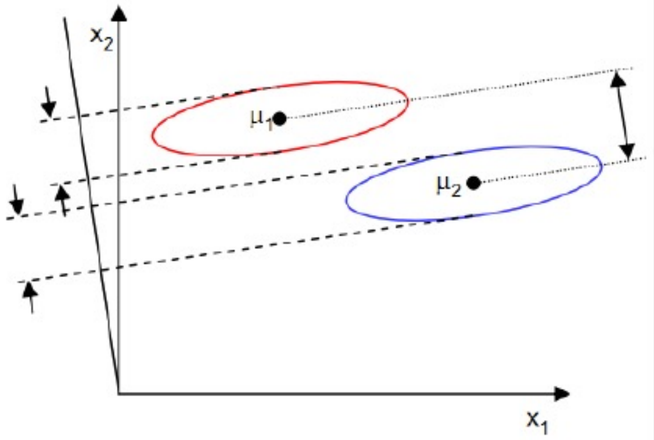
- A solution proposed by Fisher is to:
  - **maximize distance between projected class means**
  - **while also achieving a small variance within each class**

- Therefore, the **Fisher linear discriminant** (also called LDA) tries to find $y = \boldsymbol{w}^T \boldsymbol{x}$ that maximizes:

$$\underset{\boldsymbol{w}}{\text{argmax}}[J(\boldsymbol{w})] \qquad J(\boldsymbol{w}) = \frac{|\tilde{\mu}_1 - \tilde{\mu}_2|}{\tilde{s}_1^2 + \tilde{s}_2^2} \qquad \tilde{s}_k^2 = \sum_{\boldsymbol{x} \in C_k} (y - \tilde{\mu}_k)^2$$

  - Where $(\tilde{s}_1^2 + \tilde{s}_2^2)$ is called **within-class scatter** of the projected examples

  - Therefore, we will be looking for a projection where examples from the same class are projected very close to each other and, at the same time, the projected means are as farther apart as possible



LDA Derivation (Two Class)

- In order to find the optimum projection w*, we need to express $J(\boldsymbol{w}) = \frac{|\tilde{\mu}_1 - \tilde{\mu}_2|}{\tilde{s}_1^2 + \tilde{s}_2^2}$ as an explicit function of w
- The difference between the projected means can be expressed as:

$$|\tilde{\mu}_1 - \tilde{\mu}_2| = (\tilde{\mu}_1 - \tilde{\mu}_2)^2 = \left(\boldsymbol{w}^T\boldsymbol{\mu}_1 - \boldsymbol{w}^T\boldsymbol{\mu}_2\right)^2 = \boldsymbol{w}^T(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T\boldsymbol{w} = \boldsymbol{w}^T\boldsymbol{S}_B\boldsymbol{w}$$

  - The matrix S_B is called the **between-class scatter**

- The scatter of the projection y can then be expressed as

$$\tilde{s}_k^2 = \sum_{x \in C_k} (y - \tilde{\mu}_k)^2 = \sum_{x \in C_k} \left(\boldsymbol{w}^T\boldsymbol{x} - \boldsymbol{w}^T\boldsymbol{\mu}_k\right)^2 = \sum_{x \in C_k} \boldsymbol{w}^T(\boldsymbol{x} - \boldsymbol{\mu}_k)(\boldsymbol{x} - \boldsymbol{\mu}_k)\boldsymbol{w} = \boldsymbol{w}^T S_k \boldsymbol{w}$$

- Therefore

$$\tilde{s}_1^2 + \tilde{s}_2^2 = \boldsymbol{w}^T S_1 \boldsymbol{w} + \boldsymbol{w}^T S_2 \boldsymbol{w} = \boldsymbol{w}^T (S_1 + S_2)\boldsymbol{w} = \boldsymbol{w}^T S_W \boldsymbol{w}$$

  - The matrix S_W is called the **within-class scatter**

- Finally the Fisher criterion become:

$$J(\boldsymbol{w}) = \frac{|\tilde{\mu}_1 - \tilde{\mu}_2|}{\tilde{s}_1^2 + \tilde{s}_2^2} \quad \Rightarrow \quad \boxed{J(\boldsymbol{w}) = \frac{\boldsymbol{w}^T\boldsymbol{S}_B\boldsymbol{w}}{\boldsymbol{w}^T S_W \boldsymbol{w}}}$$

How find w* that maximize J(w)?

LDA Derivation (Two Class)

- The maxima of $J(w)$ is computed by the taking derivative with respect to w and setting it to 0:

$$\frac{\partial}{\partial w}[J(w)] = \frac{\partial}{\partial w}\left[\frac{\boldsymbol{w^T S_B w}}{\boldsymbol{w^T S_W w}}\right] \Rightarrow \frac{\partial[\boldsymbol{w^T S_B w}]}{\partial w} \times [\boldsymbol{w^T S_W w}] - \frac{\partial[\boldsymbol{w^T S_W w}]}{\partial w} \times [\boldsymbol{w^T S_B w}] = 0$$

$$2\boldsymbol{S_B w} \times [\boldsymbol{w^T S_W w}] - 2\boldsymbol{S_W w} \times [\boldsymbol{w^T S_B w}] = 0$$

- Dividing by $\boldsymbol{w^T S_W w}$:

$$2\boldsymbol{S_B w} \times \frac{[\boldsymbol{w^T S_W w}]}{\boldsymbol{w^T S_W w}} - 2\boldsymbol{S_W w} \times \frac{[\boldsymbol{w^T S_B w}]}{\boldsymbol{w^T S_W w}} = 0 \quad \Rightarrow \quad \boldsymbol{S_B w} = J\boldsymbol{S_W w}$$

$$\boldsymbol{S_B w} = J\boldsymbol{S_W w} \xrightarrow{if\ S_W\ is\ invertible} \boldsymbol{S_W^{-1} S_B w} = J\boldsymbol{w}$$

- It means that $\boldsymbol{w}$ is the eigenvector of $\boldsymbol{S_W^{-1} S_B}$
- Thus, we can solve eigenvalue problem to find $\boldsymbol{w}$. It leads to:

$$\boldsymbol{w}^* = \text{argmax}\left[\frac{\boldsymbol{w^T S_B w}}{\boldsymbol{w^T S_W w}}\right] \xrightarrow{\text{solve eigenvalue problem}} \boxed{\boldsymbol{w}^* = \boldsymbol{S_W^{-1}(\mu_1 - \mu_2)}}$$

LDA Derivation

- $S_B \boldsymbol{w}$ for any vector w, points in the same direction as $(\boldsymbol{\mu_1} - \boldsymbol{\mu_2})$

$$S_B \mathrm{w} = (\boldsymbol{\mu_1} - \boldsymbol{\mu_2}) \underbrace{(\boldsymbol{\mu_1} - \boldsymbol{\mu_2})^T \boldsymbol{w}}_{\alpha}$$

$$\boldsymbol{w}^* = S_W^{-1}(\boldsymbol{\mu_1} - \boldsymbol{\mu_2})$$

LDA Derivation (two class)

- In summary:

$$\boldsymbol{w}^* = \text{argmax}\left[\frac{\boldsymbol{w}^T \boldsymbol{S_B} \boldsymbol{w}}{\boldsymbol{w}^T S_W \boldsymbol{w}}\right] \xrightarrow{\text{solve eigenvalue problem}} \boldsymbol{w}^* = S_W^{-1}(\boldsymbol{\mu_1} - \boldsymbol{\mu_2})$$

  - **Between-class scatter** matrix is:

$$\mathrm{S_B} = (\boldsymbol{\mu_1} - \boldsymbol{\mu_2})(\boldsymbol{\mu_1} - \boldsymbol{\mu_2})^T$$

  - **Within-class scatter** matrix is:

$$S_W = S_1 + S_2$$

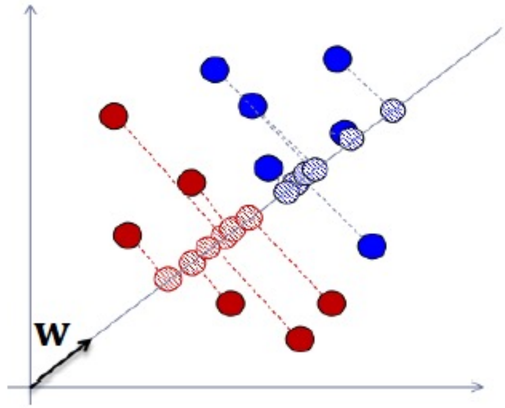
    - Where

$$\boldsymbol{\mu_k} = \frac{1}{N_k} \sum_{\boldsymbol{x} \in C_k} x \quad \textit{and} \quad S_k = (\boldsymbol{x} - \boldsymbol{\mu_k})(\boldsymbol{x} - \boldsymbol{\mu_k})$$

LDA Algorithm (two class)

- **Input:** Data $X_{D\times N}$ (each column a D-dim sample)

1. X_1, X_2: a $D \times N_1$ and $D \times N_2$ matrix including samples of class C_1 and C_2
2. $\boldsymbol{\mu_1}, \boldsymbol{\mu_2}$ = sample mean of X_1 and X_2
3. S_1, S_2 = scatter matrix of X_1 and X_2
4. $S_W = S_1 + S_2$ (within class scatter matrix)
5. $\mathbf{w} = S_W^{-1}(\boldsymbol{\mu_1} - \boldsymbol{\mu_2})$
6. The transformed samples will be $Y = \boldsymbol{w}^T X$



- **Output:** Transformed data $Y_{1\times N}$ (each column a 1-dim sample)

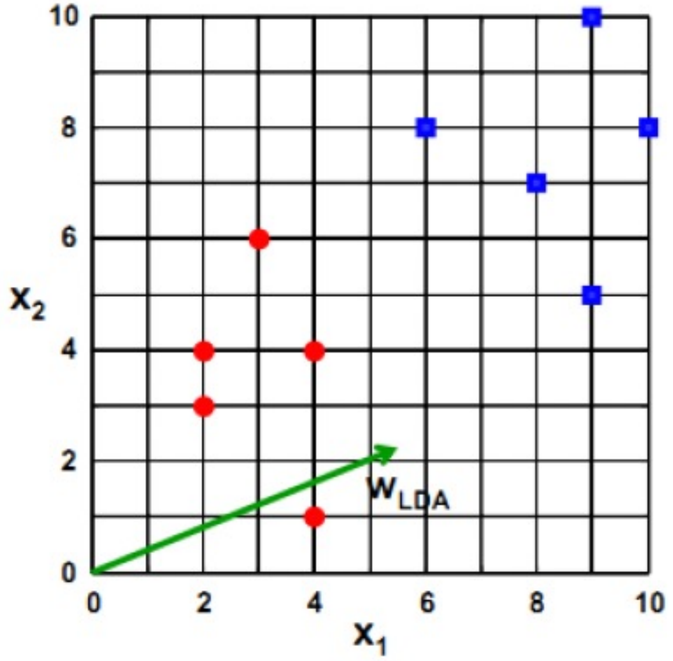
- In the transformed domain we can classify x using a threshold on $\boldsymbol{w}^T \boldsymbol{x}$

Exercise: LDA (two class)

- Compute the Linear Discriminant projection for the following two-dimensional dataset
  - X1=(x1,x2)={(4,1),(2,4),(2,3),(3,6),(4,4)}
  - X2=(x1,x2)={(9,10),(6,8),(9,5),(8,7),(10,8)}

- Find the solution by finding the eigenvalue of $S_W^{-1}S_B$

- Find the solution directly by $\boldsymbol{w}^* = S_W^{-1}(\boldsymbol{\mu_1} - \boldsymbol{\mu_2})$



LDA (Multi class)

- Fisher's LDA can be generalized to C-class problems

$$y = \boldsymbol{w^T x} \qquad \Rightarrow \qquad \boldsymbol{y} = W^T \boldsymbol{x}$$

  - The generalization of the within-class scatter matrix is:

$$S_W = \sum_{k=1}^{C} S_k = \sum_{k=1}^{C} \sum_{\boldsymbol{x} \in C_k} (\boldsymbol{x} - \boldsymbol{\mu_k})(\boldsymbol{x} - \boldsymbol{\mu_k})$$

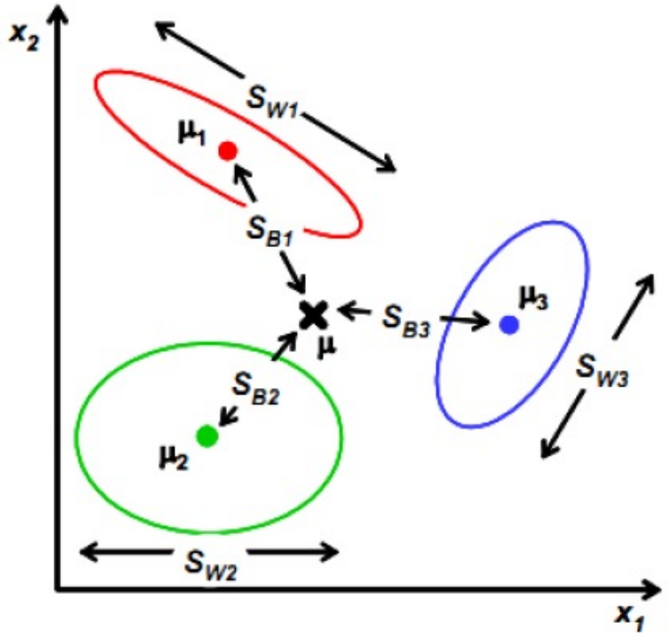
  - The generalization of the between-class scatter matrix is:

$$S_B = \sum_{k=1}^{C} N_k (\boldsymbol{\mu_k} - \boldsymbol{\mu})(\boldsymbol{\mu_k} - \boldsymbol{\mu})^T$$

    - Where $\boldsymbol{\mu} = \frac{1}{N} \sum_{\forall x} x$ and $\boldsymbol{\mu_k} = \frac{1}{N_k} \sum_{x \in C_k} x$
  - Then, the Fisher's LDA objective function is generalized as:

$$\boldsymbol{W^*} = \underset{\boldsymbol{W}}{\operatorname{argmax}} \left[\frac{|\boldsymbol{w^T S_B w}|}{|\boldsymbol{w^T} S_W \boldsymbol{w}|} \right]$$

    - Note that since the projection is no longer a scalar, the determinant of the scatter matrices |.| is used to obtain a scalar objective function:



LDA (Multi class)

- It can be shown that the optimal projection matrix W* is the one whose columns are eigenvectors corresponding to the largest eigenvalues of $S_W^{-1}S_B$:

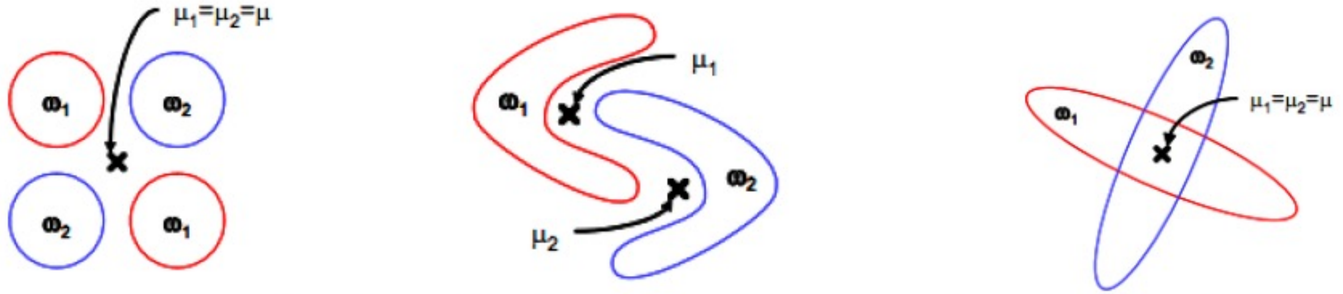
$$\boldsymbol{W}^* = [w_1^*|w_2^*|\dots|w_{C-1}^*] = \underset{\boldsymbol{W}}{\text{argmax}}\left[\frac{|\boldsymbol{w}^T\boldsymbol{S_B}\boldsymbol{w}|}{|\boldsymbol{w}^T\boldsymbol{S_W}\boldsymbol{w}|}\right] \quad \Rightarrow \quad S_W^{-1}S_B\boldsymbol{w_k} = \lambda_k\boldsymbol{w_k}$$

- Note that: at most (C-1) of eigenvalues λ_i will be non-zero

LDA Limitation

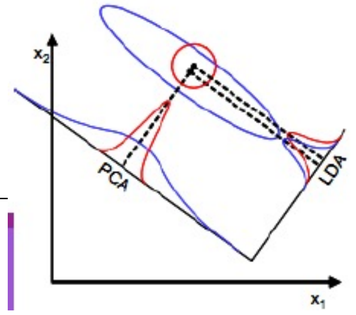
- LDA provides at most $(C-1)$ feature dimension

- LDA can be seen as a parametric method because it assumes unimodal Gaussian likelihood
  - It is not suitable when the distributions are multi-modal or significantly non-Gaussian



- LDA fails when the discriminatory information is not in the mean (e.g. $\mu_1 = \mu_2$) but rather in the variance of the data (in this condition J(w) =0)

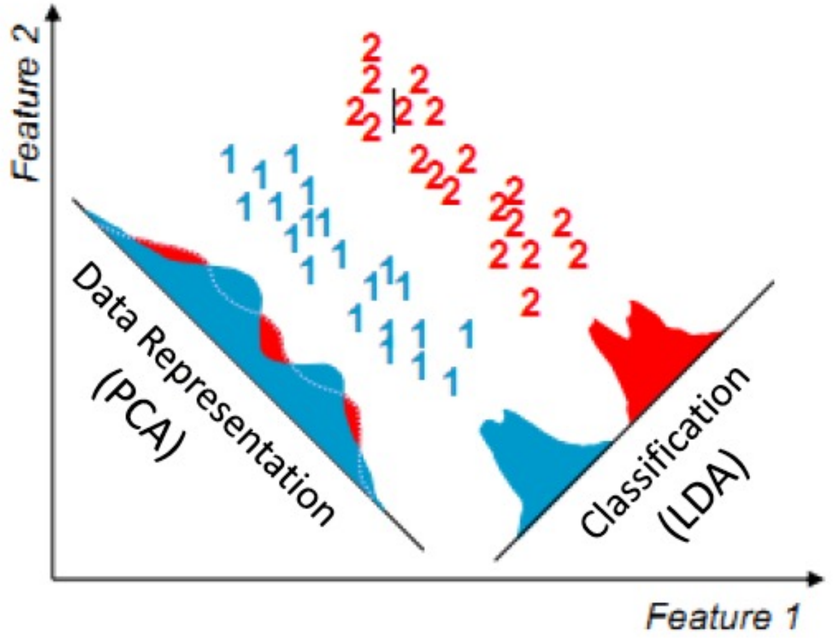
- LDA fails If classes have large overlap when projected to **any** line



PCA vs. LDA

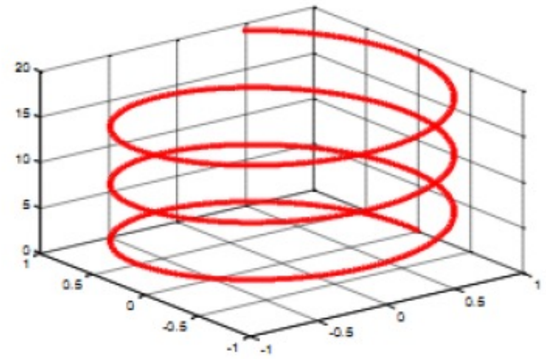
- PCA
  - Linear & unsupervised
  - Performs dimensionality reduction while **preserving largest variances in the data**
  - At most find d<<D Eigenvector

- LDA
  - Linear & supervised
  - Performs dimensionality reduction while maximize between-class scatter matrix and at the same time minimize the within-scatter matrix in the projected subspace
  - At most find (C-1) Eigenvector



Linear Methods Drawback

- Data may not be best summarized by linear combination of features and many nonlinear structures might be invisible to PCA and LDA



Does a linear methods can discover a perfect 1D structure? What about non-linear methods?

- Solution: nonlinear dimensionality reduction approach (Manifold learning) such as:
  - (is not discussed here!)
  - Kernel PCA
  - Kernel LDA
  - ISOMAP, or
  - Locally Linear Embedding

3 ways to build classifiers

(1) Generative models (e.g., LDA) [We'll learn about LDA next lecture.]
- Assume sample points come from probability distributions, different for each class.
- Guess form of distributions
- For each class C, fit distribution parameters to class C points, giving $f(X|Y = \mathrm{C})$
- For each C, estimate $P(Y = \mathrm{C})$
- Bayes' Theorem gives $P(Y|X)$
- If 0-1 loss, pick class C that maximizes $P(Y = \mathrm{C}|X = x)$ [posterior probability]
  equivalently, maximizes $f(X = x|Y = \mathrm{C})\, P(Y = \mathrm{C})$

(2) Discriminative models (e.g., logistic regression) [We'll learn about logistic regression in a few weeks.]
- Model $P(Y|X)$ directly

(3) Find decision boundary (e.g., SVM)
- Model $r(x)$ directly (no posterior)

3 ways to build classifiers

Advantage of (1 & 2): $P(Y|X)$ tells you probability your guess is wrong
[This is something SVMs don't do.]
Advantage of (1): you can diagnose outliers: $P(X)$ is very small
Disadvantages of (1): often hard to estimate distributions accurately;
real distributions rarely match standard ones.

Revisit: Gaussian Discrimination Analysis (QDA, LDA)

Gaussian Discrimination Analysis

Fundamental assumption: each class comes from normal distribution (Gaussian).

$$X \sim \mathcal{N}(\mu, \sigma^2) : f(x) = \frac{1}{(\sqrt{2\pi}\sigma)^d} \exp\left(-\frac{\|x-\mu\|^2}{2\sigma^2}\right) \qquad [\mu \ \& \ x = \text{vectors}; \ \sigma = \text{scalar}; \ d = \text{dimension}]$$

For each class C, suppose we estimate mean μ_C, variance σ_C^2, and prior $\pi_C = P(Y = C)$.

Given x, Bayes decision rule $r^*(x)$ predicts class C that maximizes $f(X = x|Y = C)\,\pi_C$.

$\ln \omega$ is monotonically increasing for $\omega > 0$, so it is equivalent to maximize

$$Q_C(x) = \ln\left((\sqrt{2\pi})^d f_C(x)\,\pi_C\right) = -\frac{\|x-\mu_C\|^2}{2\sigma_C^2} - d\ln\sigma_C + \ln\pi_C$$

↑ quadratic in x. ↑ normal PDF, estimates $f(X = x|Y = C)$

Quadratic Discrimination Analysis (QDA)

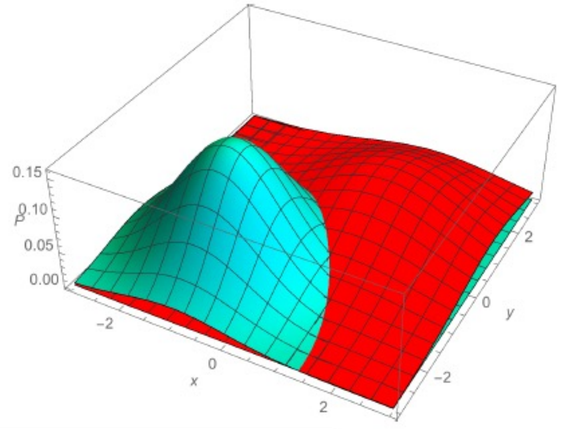
Suppose only 2 classes C, D. Then

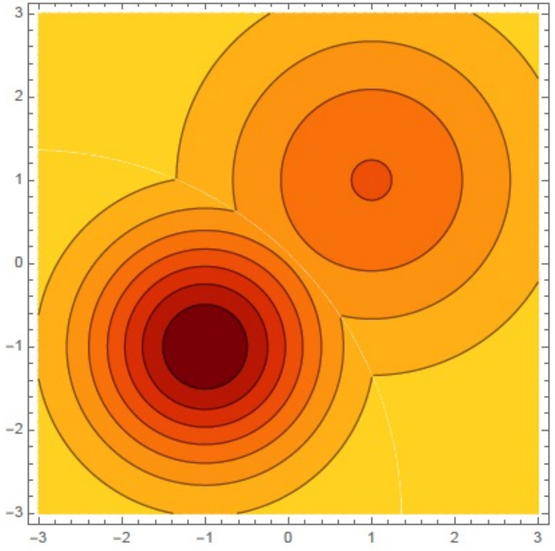
$$r^*(x) = \begin{cases} \text{C} & \text{if } Q_{\text{C}}(x) - Q_{\text{D}}(x) > 0, \\ \text{D} & \text{otherwise.} \end{cases}$$

[Pick the class with the biggest posterior probability]

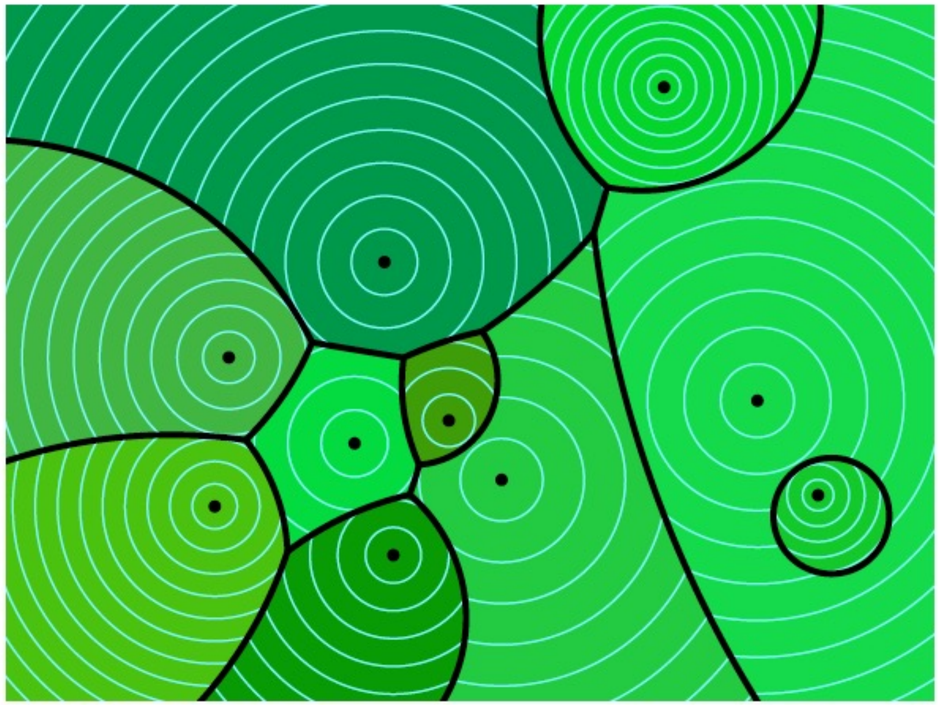
Decision fn is quadratic in x. Bayes decision boundary is $Q_{\text{C}}(x) - Q_{\text{D}}(x) = 0$.

- In 1D, B.d.b. may have 1 or 2 points. [Solutions to a quadratic equation]
- In d-D, B.d.b. is a quadric. [In 2D, that's a conic section]





Quadratic Discrimination Analysis (QDA)



multiplicative.pdf [The feature space gets partitioned into regions. In two or more dimensions, you typically wind up with multiple decision boundaries that adjoin each other at joints. It looks like a sort of Voronoi diagram. In fact, it's a special kind of Voronoi diagram called a multiplicatively, additively weighted Voronoi diagram.]

Quadratic Discrimination Analysis (QDA)

To recover posterior probabilities in 2-class case, use Bayes:

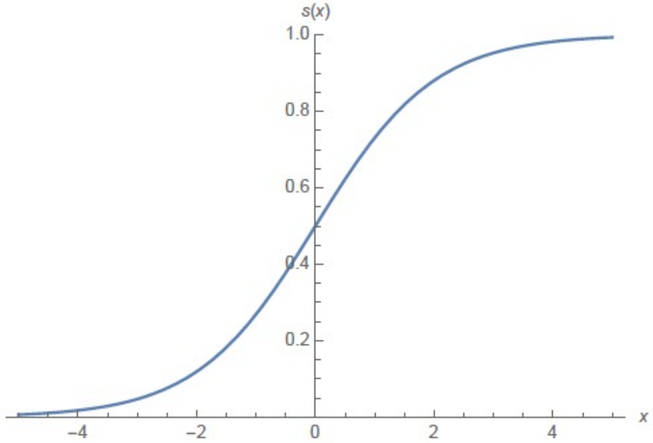
$$P(Y = \mathrm{C}|X) = \frac{f(X|Y = \mathrm{C})\,\pi_{\mathrm{C}}}{f(X|Y = \mathrm{C})\,\pi_{\mathrm{C}} + f(X|Y = \mathrm{D})\,\pi_{\mathrm{D}}}$$

recall $e^{Q_{\mathrm{C}}(x)} = (\sqrt{2\pi})^d\, f_{\mathrm{C}}(x)\,\pi_{\mathrm{C}}$ [by definition of Q_{C}]

$$\begin{aligned} P(Y = \mathrm{C}|X = x) &= \frac{e^{Q_{\mathrm{C}}(x)}}{e^{Q_{\mathrm{C}}(x)} + e^{Q_{\mathrm{D}}(x)}} = \frac{1}{1 + e^{Q_{\mathrm{D}}(x) - Q_{\mathrm{C}}(x)}} \\ &= s(Q_{\mathrm{C}}(x) - Q_{\mathrm{D}}(x)), \quad \text{where} \end{aligned}$$

$$s(\gamma) = \frac{1}{1 + e^{-\gamma}} \quad \Leftarrow \underline{\text{logistic fn}} \text{ aka } \underline{\text{sigmoid fn}}$$

[recall $Q_C - Q_D$ is the decision fn]



logistic.pdf [The logistic function. Write beside it:] $s(0) = \frac{1}{2}$, $s(\infty) \to 1$, $s(-\infty) \to 0$, monotonically increasing.

[We interpret $s(0) = \frac{1}{2}$ as saying that on the decision boundary, there's a 50% chance of class C and a 50% chance of class D.]

Linear Discrimination Analysis (LDA)

[LDA is a variant of QDA with linear decision boundaries. It's less likely to overfit than QDA.]

Fundamental assumption: all the Gaussians have same variance σ.
[The equations simplify nicely in this case.]

$$Q_C(x) - Q_D(x) = \underbrace{\frac{(\mu_C - \mu_D) \cdot x}{\sigma^2}}_{w \cdot x} \underbrace{- \frac{\|\mu_C\|^2 - \|\mu_D\|^2}{2\sigma^2} + \ln \pi_C - \ln \pi_D}_{+\alpha}$$

[The quadratic terms in Q_C and Q_D canceled each other out!]

Now it's a linear classifier! Choose C that maximizes linear discriminant fn

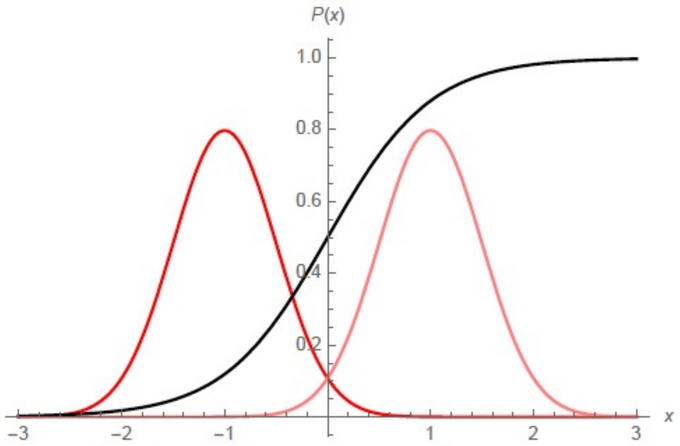
$$\frac{\mu_C \cdot x}{\sigma^2} - \frac{\|\mu_C\|^2}{2\sigma^2} + \ln \pi_C$$

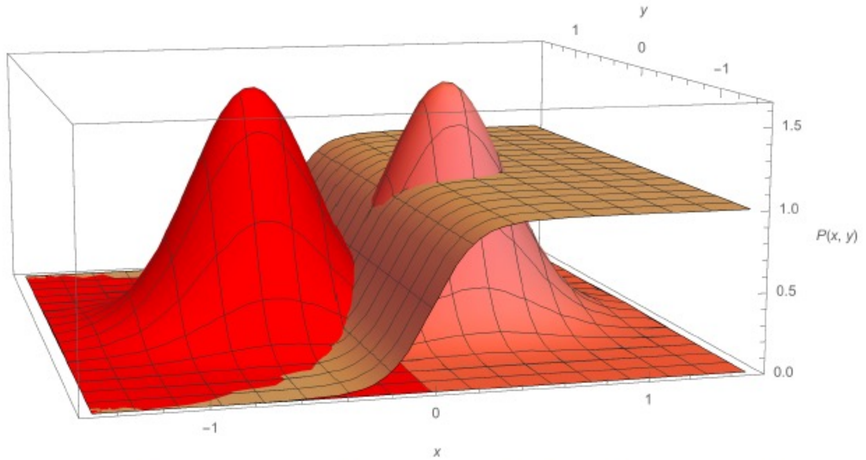
[this works for any number of classes]

In 2-class case: decision boundary is $w \cdot x + \alpha = 0$
posterior is $P(Y = C|X = x) = s(w \cdot x + \alpha)$

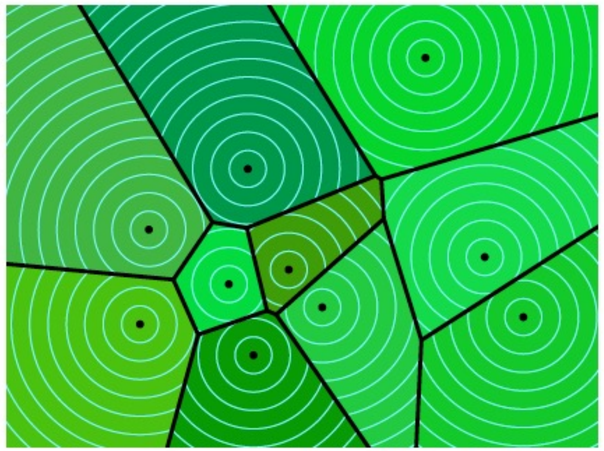
[The effect of "$w \cdot x + \alpha$" is to scale and translate the logistic fn in x-space. It's a linear transformation.]

Linear Discrimination Analysis (LDA)





lda1d.pdf, lda2d.pdf [Two Gaussians (red) and the logistic function (black). The logistic function is the right Gaussian divided by the sum of the Gaussians. Observe that even when the Gaussians are 2D, the logistic function still looks 1D.]



voronoi.pdf [When you have many classes, their LDA decision boundaries form a classical Voronoi diagram if the priors π_C are equal. All the Gaussians have the same width.]

$$\text{If } \pi_C = \pi_D = \frac{1}{2} \quad \Rightarrow \quad (\mu_C - \mu_D) \cdot x - (\mu_C - \mu_D) \cdot \left(\frac{\mu_C + \mu_D}{2} \right) = 0$$

This is the centroid method!

Maximum Likelihood Estimation of Parameters

[To use Gaussian discriminant analysis, we must first fit Gaussians to the sample points and estimate the class prior probabilities. We'll do priors first—they're easier, because they involve a discrete distribution. Then we'll fit the Gaussians—they're less intuitive, because they're continuous distributions.]

Let's flip biased coins! Heads with probability p; tails w/prob. $1 - p$.

10 flips, 8 heads, 2 tails. [Let me ask you a weird question.] What is the most likely value of p?

# of heads is $X \sim \mathcal{B}(n, p)$, binomial distribution:

$$P[X = x] = \binom{n}{x} p^x (1-p)^{n-x}$$ [this is the probability of getting exactly x heads in n coin flips]

Our example: $n = 10$,

$$P[X = 8] = 45 p^8 (1-p)^2 \quad \stackrel{\text{def}}{=} \mathcal{L}(p)$$

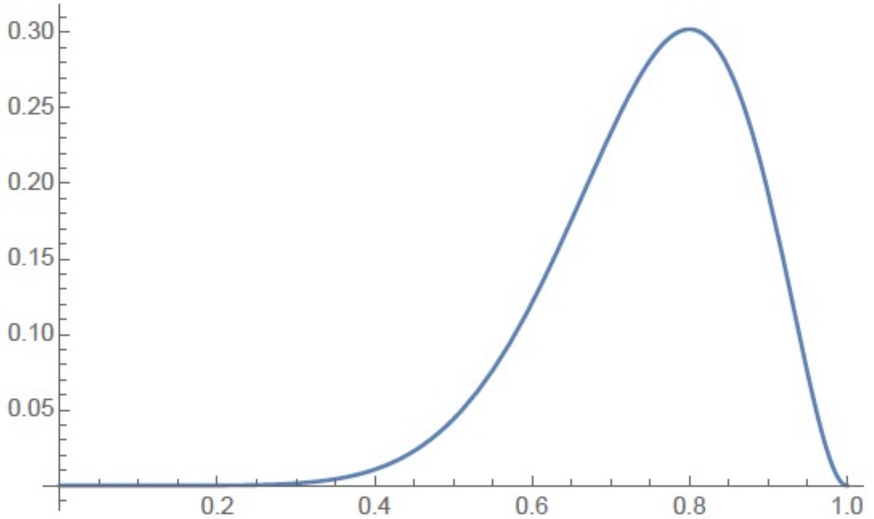
Probability of 8 heads in 10 flips:
written as a fn $\mathcal{L}(p)$ of distribution parameter(s), this is the likelihood fn.

Maximum Likelihood Estimation of Parameters

Maximum likelihood estimation (MLE): A method of estimating the parameters of a statistical model by picking the params that maximize [the likelihood function] $\mathcal{L}$.
. . . is one method of density estimation: estimating a PDF [probability density function] from data.

[Let's phrase it as an optimization problem.]

Find p that maximizes $\mathcal{L}(p)$.



binomlikelihood.pdf [Graph of $\mathcal{L}(p)$ for this example.]

Maximum Likelihood Estimation of Parameters

Solve this example by setting derivative = 0:

$$\frac{d\mathcal{L}}{dp} = 360p^7(1-p)^2 - 90p^8(1-p) = 0$$

$$\Rightarrow \quad 4(1-p) - p = 0 \quad \Rightarrow \quad p = 0.8$$

[It shouldn't seem surprising that a coin that comes up heads 80% of the time is the coin most likely to produce 8 heads in 10 flips.]

[Note: $\frac{d^2\mathcal{L}}{dp^2} \doteq -18.9 < 0$ at $p = 0.8$, confirming it's a maximum.]

[Here's how this applies to prior probabilities. Suppose our data set is 10 sample points, and 8 of them are of class C and 2 are not. Then our estimated prior for class C will be $\pi_C = 0.8$.]

Likelihood of a Gaussian

Given sample points $X_1, X_2, \ldots, X_n$, find best-fit Gaussian.

[Now we want a normal distribution instead of a binomial distribution. If you generate a random point from a normal distribution, what is the probability that it will be exactly at X_1?]

[Zero. So it might seem like we have a problem here. With a continuous distribution, the probability of generating any particular point is zero. But we're just going to ignore that and do "likelihood" anyway.]

Likelihood of generating these points is

$$\mathcal{L}(\mu, \sigma; X_1, \ldots, X_n) = f(X_1)\, f(X_2) \cdots f(X_n).$$

[How do we maximize this?]

Likelihood of a Gaussian

The log likelihood $\ell(\cdot)$ is the ln of the likelihood $\mathcal{L}(\cdot)$.
Maximizing likelihood $\Leftrightarrow$ maximizing log likelihood.

$$\begin{aligned}\ell(\mu, \sigma; X_1, ..., X_n) &= \ln f(X_1) + \ln f(X_2) + ... + \ln f(X_n) \\ &= \sum_{i=1}^{n} \underbrace{\left(-\frac{\|X_i - \mu\|^2}{2\sigma^2} - d \ln \sqrt{2\pi} - d \ln \sigma \right)}_{\text{ln of normal PDF}}\end{aligned}$$

Want to set $\nabla_\mu \ell = 0, \frac{\partial \ell}{\partial \sigma} = 0$

$$\nabla_\mu \ell = \sum_{i=1}^{n} \frac{X_i - \mu}{\sigma^2} = 0 \quad \Rightarrow \quad \hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} X_i$$

[The hats ˆ mean "estimated"]

$$\frac{\partial \ell}{\partial \sigma} = \sum_{i=1}^{n} \frac{\|X_i - \mu\|^2 - d\sigma^2}{\sigma^3} = 0 \quad \Rightarrow \quad \hat{\sigma}^2 = \frac{1}{dn} \sum_{i=1}^{n} \|X_i - \mu\|^2$$

We don't know μ exactly, so substitute $\hat{\mu}$ for μ to compute $\hat{\sigma}$.

I.e., we use sample mean & variance of pts in class C to estimate mean & variance of Gaussian for class C.

Likelihood of a Gaussian

For QDA: estimate conditional mean $\hat{\mu}_C$ & conditional variance $\hat{\sigma}^2_C$ of **each class C separately** [as above] & estimate the priors:

$$\hat{\pi}_C = \frac{n_C}{\sum_D n_D} \quad \Leftarrow \quad \text{total sample points in all classes}$$

[$\hat{\pi}_C$ is the coin flip parameter]

For LDA: same means & priors; one variance for all classes:

$$\hat{\sigma}^2 = \frac{1}{dn} \sum_C \sum_{\{i:y_i=C\}} \|X_i - \hat{\mu}_C\|^2 \quad \Leftarrow \text{ pooled within-class variance}$$

[Notice that although LDA is computing one variance for all the data, each sample point contributes with respect to *its own class's mean*. This gives a very different result than if you simply use the global mean! It's usually smaller than the global variance. We say "within-class" because we use each point's distance from its class's mean, but "pooled" because we then pool all the classes together.]

Feature Selection

Dimensionality Reduction Methods

- **Feature selection:**
  - These methods find a new feature set of d ($d < D$) by selecting d features from existing features

- **Feature extraction:**
  - These methods select d ($d < D$) features out of D dimensions and discard $D - d$ dimensions.
    - Given a feature space $\boldsymbol{x} \in \mathbb{R}^D$ find a mapping $\boldsymbol{y} = f(\boldsymbol{x}) : \mathbb{R}^D \rightarrow \mathbb{R}^d$ with $d < D$ such that transformed feature vector $\boldsymbol{y} \in \mathbb{R}^d$ optimises an objective function

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \xrightarrow{\text{feature selection}} \begin{bmatrix} x_{i_1} \\ x_{i_2} \\ \vdots \\ x_{i_d} \end{bmatrix}$$

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \xrightarrow{\text{feature extraction}} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_d \end{bmatrix} = f\left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix}\right)$$

Why Feature Selection

- Feature selection is not necessarily required as a pre-processing step for classification/regression algorithms to perform well
  - Several algorithms employ regularization techniques to handle overfitting problem
- Why Feature Selection?
  - Removing **irrelevant** and **redundant** variables
    - Same performance could possibly be achieved with a smaller subset of complementary variables that does not contain redundant features.
  - Decreasing the time/space complexity
  - Decreasing the cost of collecting/extracting unnecessary features
  - Having a **simpler model** for the problem
    - A simpler model is less prone to overfitting problem (and has less variance/ is more robust).
    - A simpler model has more generalization on unseen samples.
    - A simpler model needs smaller training samples to be learned.
    - A simpler model is less sensitive to noise and outliers.

Feature Selection Methods

Filter Approach

- Use statistical properties of features to filter out poorly informative features.
- Ranks features independent of classifier/regressors

Wrapper Approach

- Use classifier/regressor to evaluate the score of feature subset
- They achieve better performance than filter methods

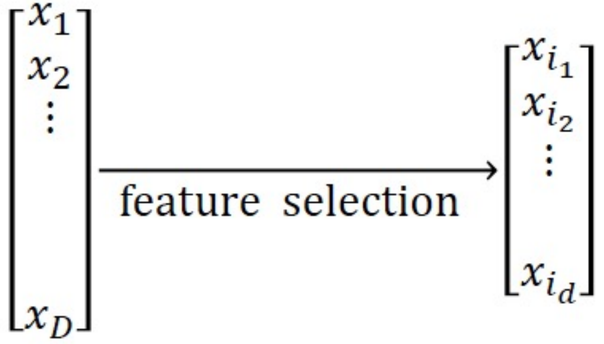
Embedded Approach

- Feature selection is done during the training of a classifier/regressor
  - E.g., Adding a regularization term in the cost function of linear classifiers

Feature Selection

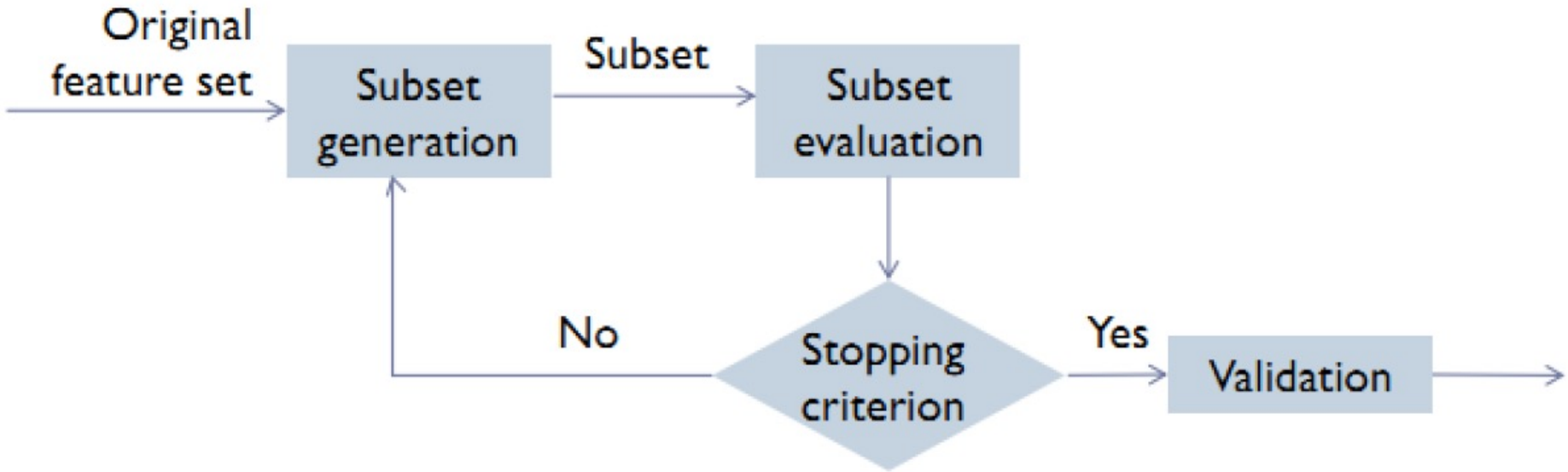
- Large number of feature is not informative, either **irrelevant** or **redundant** features
  - **Irrelevant** features are those features that **don't contribute to** a classification or regression rule.
  - **Redundant** features are those features that are **strongly correlated**.

- Feature Selection Goal: Given a feature set $X = [x_1, x_2, \dots, x_D]^T$, find a new subset $Y = [x_{i_1}, x_{i_2}, \dots, x_{i_d}]^T$ (d < D) that optimizes an objective function $J(Y)$, ideally the classification accuracy



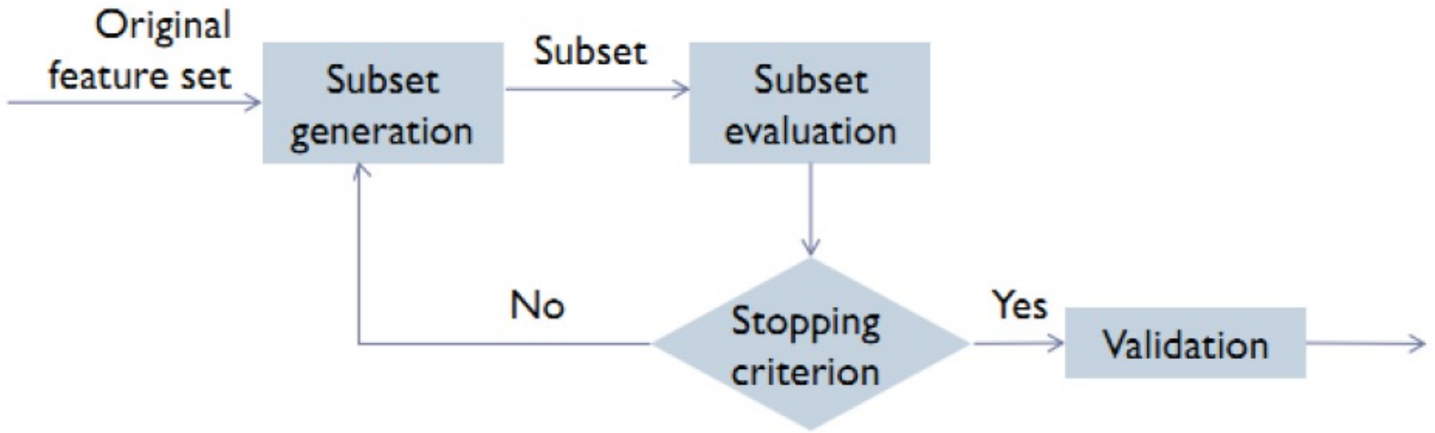
Feature Selection

- Two key steps in feature selection process:
  - **Subset generation**: A subset generation method to generate a features subset for evaluation
  - **Subset evaluation:** An evaluation measure to evaluate a candidate feature subset.



Stopping Criteria

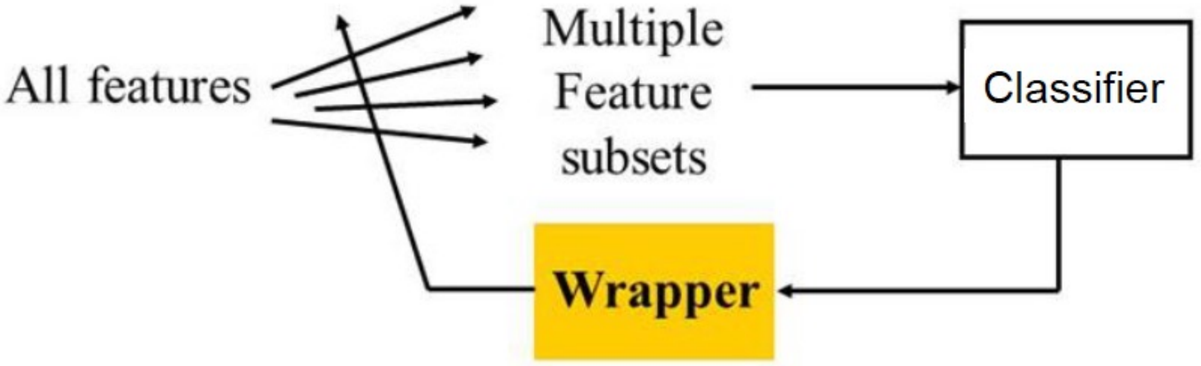
- Predefined number of features is selected
- Predefined number of iterations is reached
- Addition (or deletion) of any feature does not result in a better subset
- An optimal subset (according to the evaluation criterion) is obtained.



Subset Evaluation

- Filters use statistical measures to rank features and select a good feature subset (independent of the final classifier)

- Wrappers use classifier to evaluate the score of feature subset, e.g. error rate



Wrapper Methods: Evaluation Criteria

- Usually classification accuracy (or error rate) is used to evaluate a feature subset Y_m

- To do this, the classifier is trained on training data using the subset of features Y_m and its performance is evaluated via cross-validation

Filtering Methods: Evaluation Criteria

- Use statistical properties of features to filter out poorly informative features.
  - Ranks features independent of classifier/regressor

- What should we consider for the measure that we need ?
  - Generally, we are interested that the selected feature subset has less correlation to each other and large correlation with the class labels

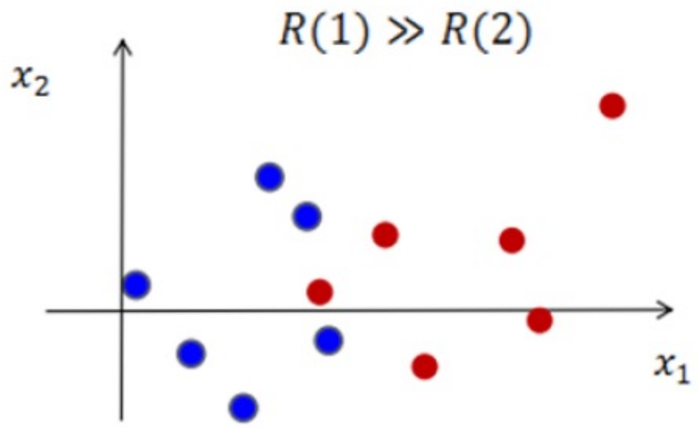
- Several well-known evaluation metrics used in filtering methods:
  - Pearson Correlation Criteria
  - Interclass distance
  - Mutual Information Criteria
  - Information Gain Criteria
  - ...

Filtering Methods: Evaluation Criteria

- Interclass distance: Use distance metrics to measure class separability
  - A measure of distance between classes is defined based on distances between mean of each class/ or between members of each class.
  - Example of these metrics is: Euclidean distance, Mahalanobis distance, etc.

- Pearson Correlation Criteria
  - It is also known as “correlation coefficient”
  - Measure linear dependency between a feature and class labels

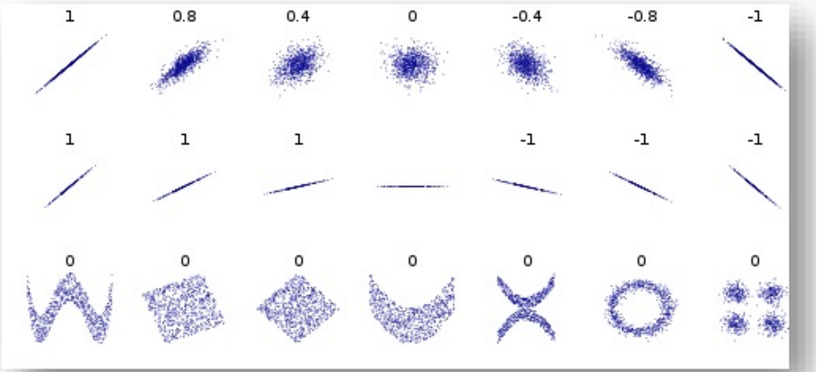
$$R(k) = \rho_{kc} = \frac{cov[x_k, c]}{\sigma_x \sigma_c}$$



Filtering Methods: Evaluation Criteria

- Good feature subsets contain features highly correlated with (predictive of) the class, yet uncorrelated with (not predictive of) each other
  - If a feature is heavily dependent on another, then it is redundant
  - Measure linear dependency

$$J(Y_M) = \frac{\sum_{i=1}^{M} \rho_{ic}}{\sum_{i=1}^{M} \sum_{j=i+1}^{M} \rho_{ij}}$$



- Where ρ_{ic} is the correlation coefficient between feature 'i' and the class label and ρ_{ij} is the correlation coefficient between features 'i' and 'j'

Filtering Methods: Evaluation Criteria

- Mutual Information Criteria
  - Measure non-linear dependency between a feature and class labels
  - The mutual information between the feature vector and the class label measures the amount by which the uncertainty in the class is decreased by knowledge of the feature vector

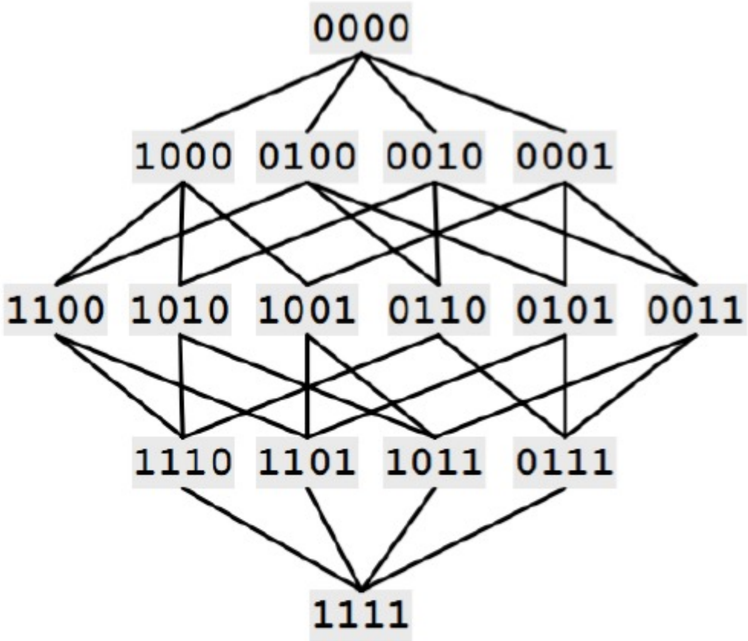
$$MI(X,Y) = E_{X,Y}\left[\log \frac{P(X,Y)}{P(X)P(Y)}\right]$$

Subset Generation or Selection

- Subset Generation can be performed via a search strategies:
  - **Exhaustive (Complete) search:** These algorithms evaluate all possible subsets
  - **Sequential search:** Search direction is guided heuristically. They may be trapped in local extrema
    - Best individual N
    - Sequential Forward Selection
    - Sequential Backward Selection
    - Plus-l Minus-r Selection
    - Bidirectional Search
    - Sequential Floating Selection
  - **Randomized search:** These algorithms incorporating randomness into their search procedure to escape local extrema
    - Simulated Annealing
    - Genetic Algorithms
    - Beam search
    - ...

Exhaustive Complete Search

- Evaluate all possible feature set and select the best one
  - For D features, we have 2^D possible subsets
  - For D features and a subset with predefined size M, we have $\binom{D}{M}$ possible subsets
- Optimal subset can be found
- Too expensive if feature dimension is large
- "Branch & Bound" or "Best first" search can also be used to prune the search space

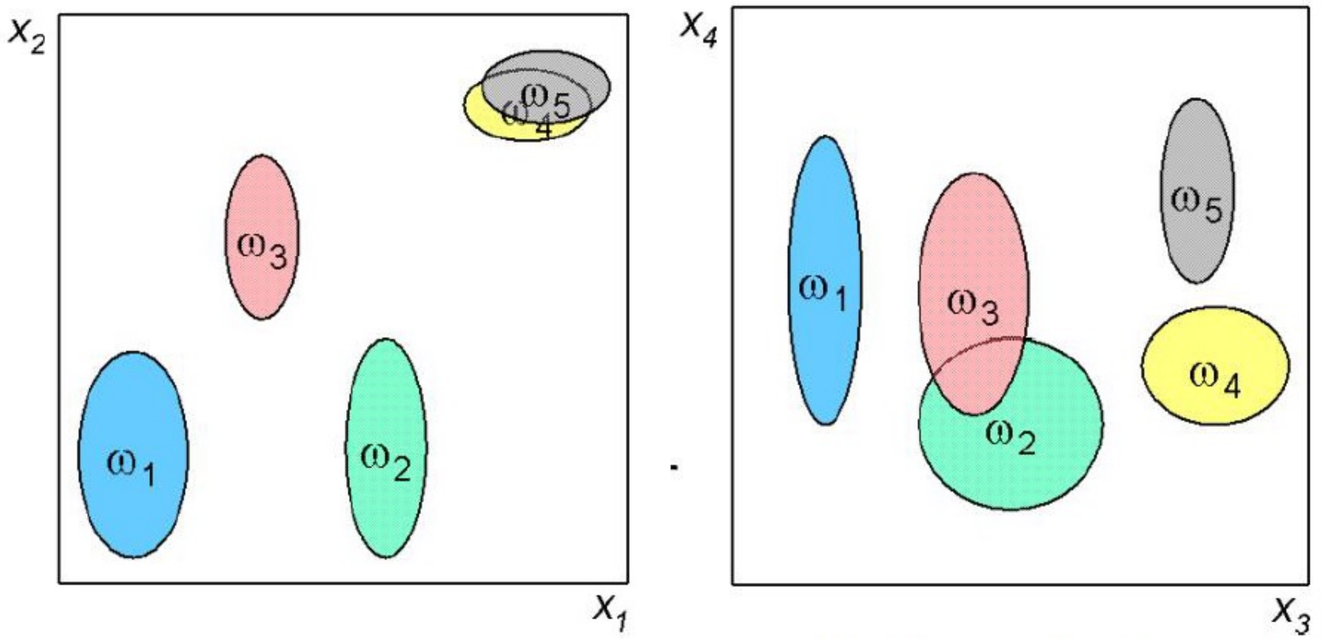


Best Individual N

- The simplest method is to assign a score to each feature and then select N top ranks features.
- 'Best Individual N' is widely used by Filters
- This strategy dose not consider co-operation between features.
- For example, suppose we want to select two features from $\{x_1, x_2, x_3\}$ and $J(x_1) > J(x_2) > J(x_3)$
  - Best Individual N: selects x_1 and x_2
  - Evidently, it is possible that for example $\{x_1, x_3\}$ be better than $\{x_1, x_2\}$, i.e. $J(\{x_1, x_3\}) > J(\{x_1, x_2\})$

Best Individual Approach Drawback

- What is the best two features if we select based on best individual approach?
- What is the best two features to be selected?

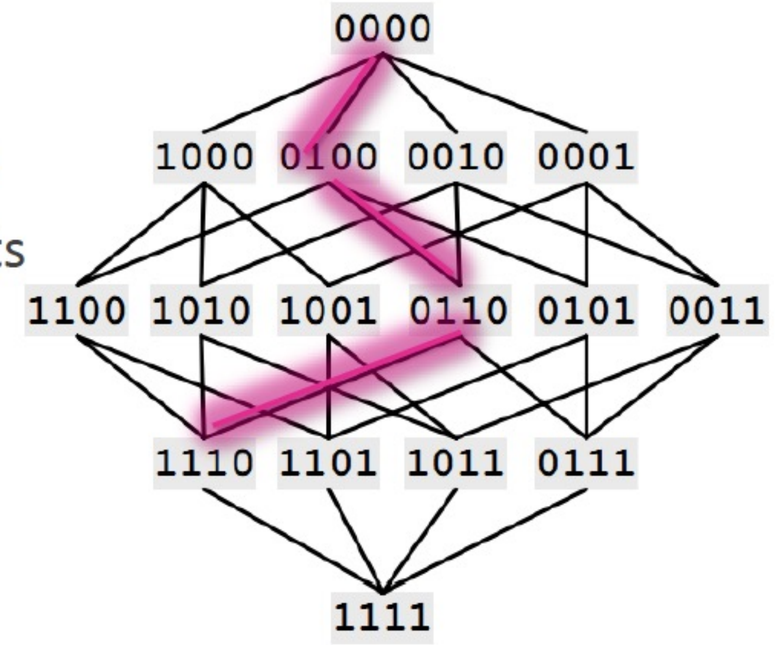


http://research.cs.tamu.edu/

Sequential Forward Selection (SFS)

- Sequential Forward Selection is the simplest greedy search algorithm

- Starting from the empty set, sequentially add the feature x^+ to the current selected features, $Y_{k+1} = Y_k \cup \{x+\}$ that results in the **highest** objective function $J(Y_{k+1})$



Notice that the number of states is larger in the middle of the search tree

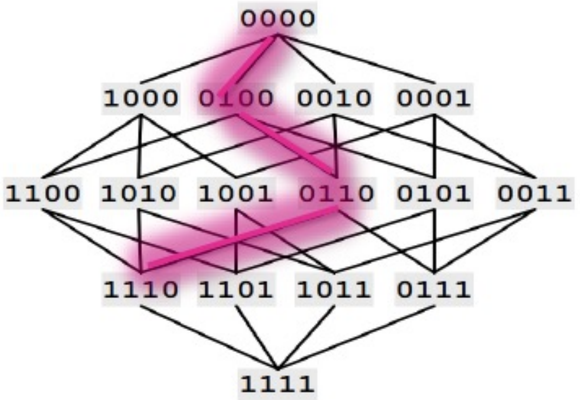
SFS Algorithm

- **Input:** Data $X_{D\times N}$ (each column a D-dim sample)

1. Start with the empty set $Y_0 = \{\emptyset\}$ and $k = 0$

2. Select the next best feature
   - $x^+ = argmax \ [J[Y_k \cup \{x\}]] \qquad \forall x \notin Y_k$
   - Update $Y_{k+1} = Y_k \cup \{x^+\}$
   - $k = k + 1$

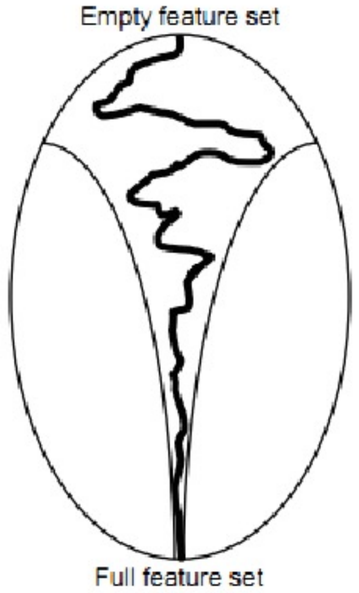
3. If stopping condition is not reached, Go back to step 2



- **Output:** Data $Y_{d\times N}$ (each column a d-dim sample)

Sequential Forward Selection (SFS)

- Search space is drawn like an ellipse to emphasize:
  - When the search is near the empty set, a large number of states can be potentially evaluated
  - Towards the full set, the region examined by SFS is narrower since most of the features have already been selected



- SFS performs best when the optimal subset has a small number of features

- The main disadvantage of SFS is that it has **no backtrack**! As a feature is added to the set, it is unable to remove the feature in the next steps

Example: Sequential Forward Selection (SFS)

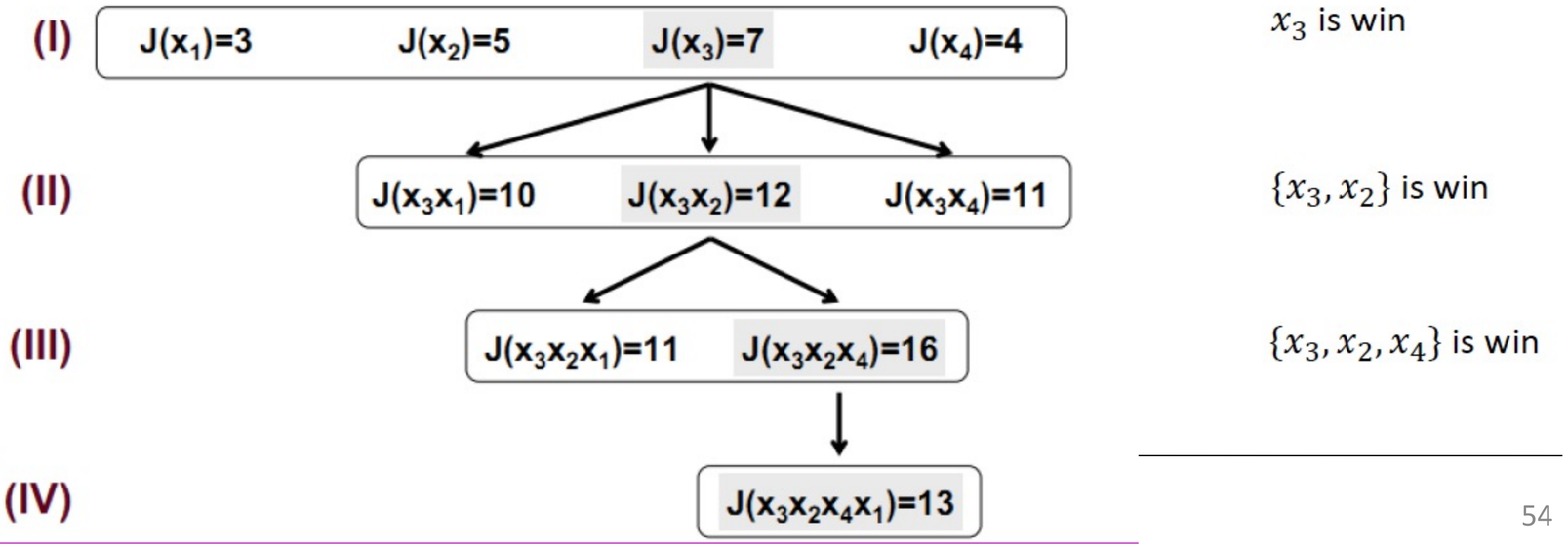
- Assume we have four features and let the following objective function

$$J(X) = -2x_1x_2 + 3x_1 + 5x_2 - 2x_1x_2x_3 + 7x_3 + 4x_4 - 2x_1x_2x_3x_4$$

  - where x_k are **indicator** variables that determine if the k-th feature has been selected (x_k=1) otherwise $x_k = 0$.

- Perform a Sequential Forward Selection to completion

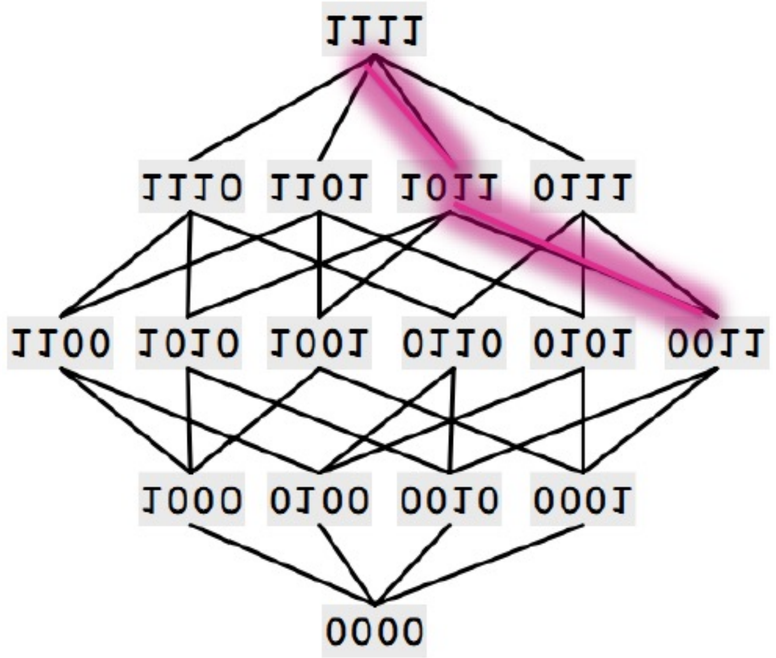
- Solution:



Sequential Backward Selection (SBS)

- Sequential Backward Selection works in the opposite direction of SFS

- Starting from the full set, sequentially remove the feature x^- that results in the **smallest decrease** in the value of the objective function $J[Y_k - \{x^-\}]$



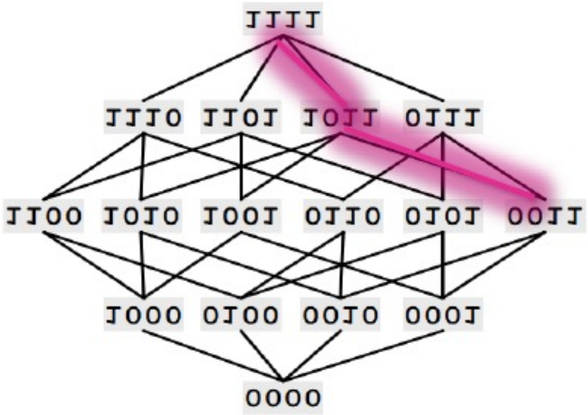
SBS Algorithm

Input: Data $X_{D\times N}$ (each column a D-dim sample)

1. Start with the Full set $Y_0 = X$ and $k = 0$

2. Remove the worst feature
   - $x^- = argmin \left[J[Y_k] - J[Y_k - \{x\}]\right] \quad \forall x \in Y_k$
   - Update $Y_{k+1} = Y_k - \{x^-\}$
   - $k = k + 1$

3. If stopping condition is not reached, go back to step 2

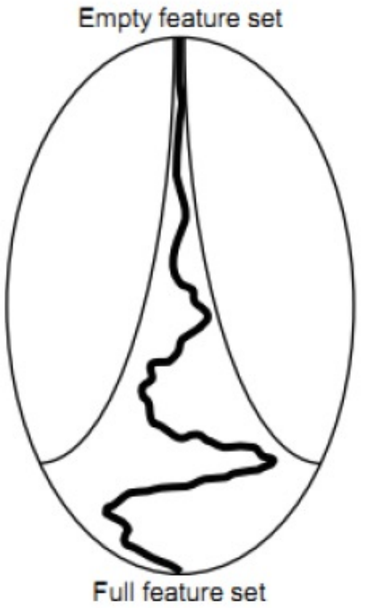


Output: Data $Y_{d\times N}$ (each column a d-dim sample)

Sequential Backward Selection (SBS)

- SBS performs best when the optimal subset has a large number of features

- Similar to SFS, the main disadvantage of SBS is that it has **no backtrack**! As a feature is deleted from the set, it is unable to add it in the next steps



Plus-L Minus-R Selection

- Plus-L Minus-R selection incorporates both SFS and SBS strategies to compensate the weaknesses of backtracking capabilities of SFS and SBS in some extent

- Plus-L Minus-R
  - If L>R, LRS starts from the empty set and repeatedly adds ‘L’ features and removes ‘R’ features
  - If L<R, LRS starts from the full set and repeatedly removes ‘R’ features followed by ‘L’ feature

- The main limitation is determining the optimal values of L and R.
- Besides, a fixed value of L and R might be not appropriate during search

Plus-L Minus-R Selection

Input: Data $X_{D\times N}$ (each column a D-dim sample)

1. If L<R start with the full set $Y_0 = X$ and go to step 3
   else start with the empty set $Y_0 = \emptyset$
2. Plus L features same as SFS
   - $x^+ = argmax \ \left[J[Y_k \cup \{x\}]\right] \qquad \forall\, x \notin Y_k$
   - Update $Y_{k+1} = Y_k \cup \{x^+\}$
3. Remove R features same as SBS
   - $x^- = argmin \ \left[J[Y_k] - J[Y_k - \{x\}]\right] \quad \forall \in Y_k$
   - Update $Y_{k+1} = Y_k - \{x^-\}$
4. $k = k + 1$. If stopping condition is not reached, go back to step 3

Output: Data $Y_{d\times N}$ (each column a d-dim sample)

Bi-directional Search

- Bidirectional Search is a parallel implementation of SFS and SBS

- To guarantee the convergence of the algorithm
  - Before SFS attempts to add a new feature, it checks that the feature was not been removed by SBS.
  - Before SBS attempts to remove a feature, it checks that the feature was not been added by SFS.

Bi-directional Search

- **Input:** Data $X_{D\times N}$ (each column a D-dim sample)

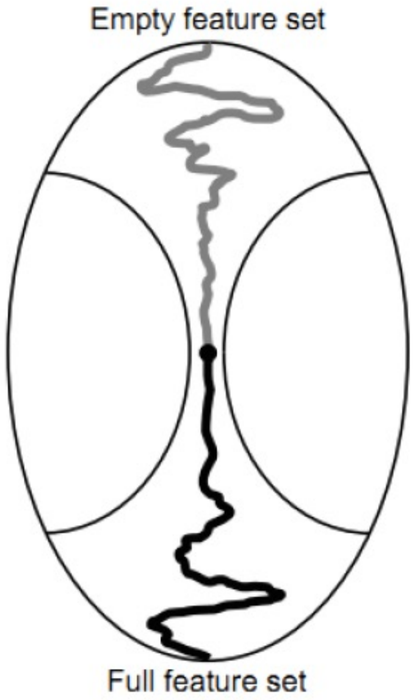
1. Start SFS with the empty set $Y_{F_0} = \{\emptyset\}$
2. Start SBS with the full set $Y_{B_0} = X$ and $k = 0$

3. Select the best feature
   - $x^+ = argmax \ [J[Y_k \cup \{x\}]]$ $\forall x \notin Y_{F_k}$ and $x \in Y_{B_k}$
   - Update $Y_{F_{k+1}} = Y_{F_k} \cup \{x^+\}$

4. Remove worst features
   - $x^- = argmax \ [J[Y_k - \{x\}]]$ $\forall x \in Y_{B_k}$ and $x \notin Y_{F_{k+1}}$
   - Update $Y_{B_{k+1}} = Y_{B_k} - \{x^-\}$

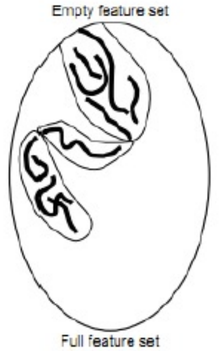
5. $k = k + 1$. If stopping condition is not reached, go back to step 3

- **Output:** Data $Y_{d\times N}$ (each column a d-dim sample)

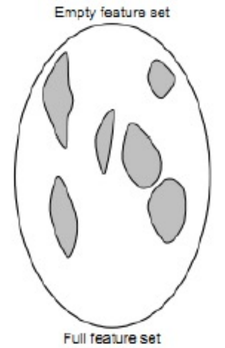


Randomized Search

- These algorithms incorporating randomness into their search procedure
  - They try to escape local extrema
  - Different Evolutionary Strategies can be used for randomized search
- Simulated Annealing:
  - Instead of always selecting the best feature, it selects a feature randomly and tries to select the best feature most of time (not always)
  - Features with higher score are more probable to be selected



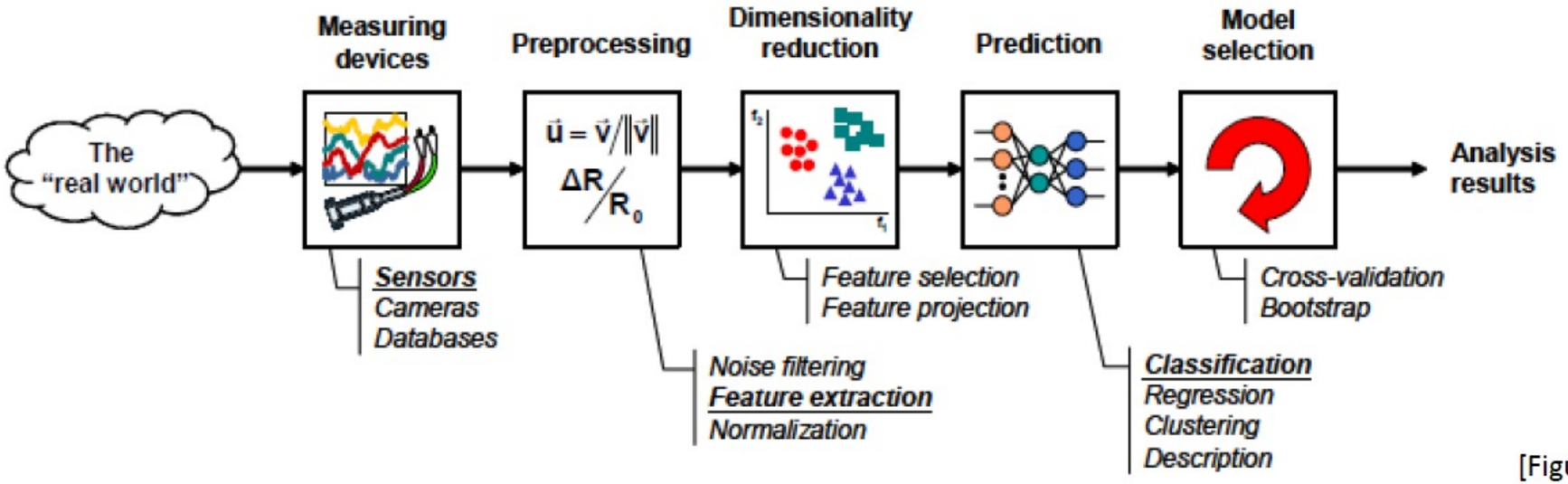
- Genetic algorithm
  - Starting with an initial random population of solutions (feature sets), evolve new populations by crossover pairs of solutions and mutating solutions according to their objective function
  - The better solutions are more likely to be selected for cross validation and mutation



Model Selection and Cross Validation

A Typical Pattern Recognition System

- A typical pattern recognition system can have:
  - A sensor
  - A preprocessing mechanism
  - A feature extraction/selection mechanism (manual or automated)
  - A regression/classification/ clustering algorithm
  - A set of examples (training set) already classified or described



[Figure by R. Gutierrez-Osuna]

Intro

- Suppose, you developed a learning method to solve a problem (classification, clustering, etc.)
- How to evaluate the learning method?
- If the learning method doesn't work well:
  - Is the selected method appropriate for your problem?
    - Suppose a case that the method assumes a Gaussian distribution. Is it reasonable for your problem?!
  - Is the training data sufficient?
  - If the method has one or more free parameters, do you select suitable values for them?
    - The number of neighbors in KNN classifier, and …
- How to select the optimal parameters?

Intro

- How do we evaluate the learning method?
- A naïve approach is training and testing over entire available training data
- If we have infinite number of samples:

the **error rate on the training** samples is the **true error** rate

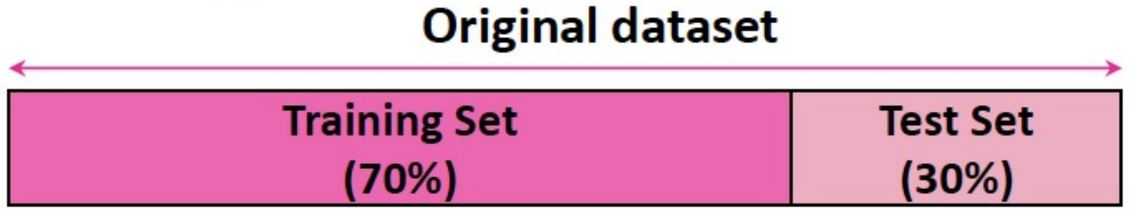
- However, in practice we have a finite set of samples!
  - The model might be overfitted on the training data.
  - The error rate on the training samples is lower than the true error rate
- How to use the limited training data to train and how to evaluate and select the best model

Evaluate the Learning Method

- Suppose you are going to implement a classifier for your classification problem
  - Using neural network, or
  - Bayesian classifier, or
  - KNN classifier
  - …
- Cancer classification (classify to malignant or benign)
- How to evaluate the performance of the developed system?

Training/Testing Procedure

- Split the data into training set and test set



1. Train your classifier using the **training set** (using 70% of training data)

2. Compute the classification error using the **test set** (using 30% of training data)

$$Er = \frac{\#miss\ classified\ samples}{\#all\ samples}$$

3. Report Er as the performance of your network

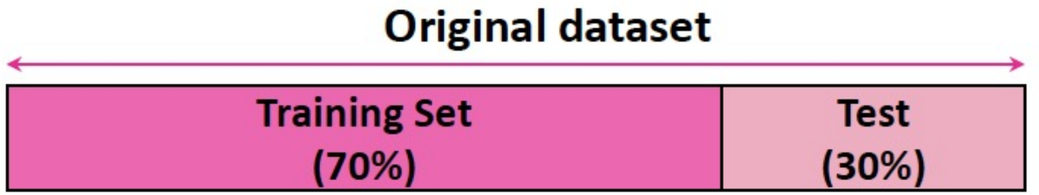
Model Selection

- Model selection: If the classifier have one or more free parameters θ, how select the best values for the parameters. For example:
  - The best k in KNN classifier
  - Select the number of hidden layers in a neural network
  - ...

- Let $h^{\theta}(x)$ show a classifier with parameter θ which is used to classify an known sample x

Model Selection: Train/Test

- Let $h^{\theta}(x)$ show a classifier with parameter θ which is used to classify an known sample x
- Suppose we have 10 different candidate values for the parameter, $\theta_1, \theta_2, \ldots, \theta_{10}$

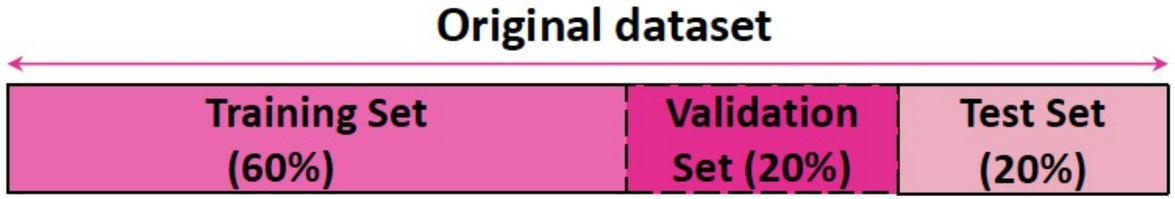


- How to select the best parameter (best model):
  1. Split the training data into training (70%) and test (30%) sets
  2. Train $h^{(\theta_1)}(x)$, $h^{(\theta_2)}(x)$, …, $h^{(\theta_{10})}(x)$ using training set (70%)
  3. Select the model with the lowest test error rate evaluated on the test set (30%)
     - Compute $Er_{test}^{(\theta_1)}$, $Er_{test}^{(\theta_2)}$, …, $Er_{test}^{(\theta_{10})}$ and select the lowest one

- Does this method have a good generalization?
  - Not exactly! It might be an optimistic estimate of generalization error
  - Because, the parameters may fit to **that test set**.

Model Selection: Train/Validation/Test

1. Split the training data into three sets:



2. Train the models $h^{(\theta_1)}(x)$, $h^{(\theta_2)}(x)$, …, $h^{(\theta_{10})}(x)$ using the training set (60%)

3. Select the model with lowest cross validation error
   - Compute $Er_V^{(\theta_1)}$, $Er_V^{(\theta_2)}$, …, $Er_V^{(\theta_{10})}$ using validation set (20%)

4. On the selected model, report the test error rate as the generalization error
   - Compute $Er_{test}^{(\theta^*)}$ for the model on the test set (20%)

Model Selection: Train/Validation/Test

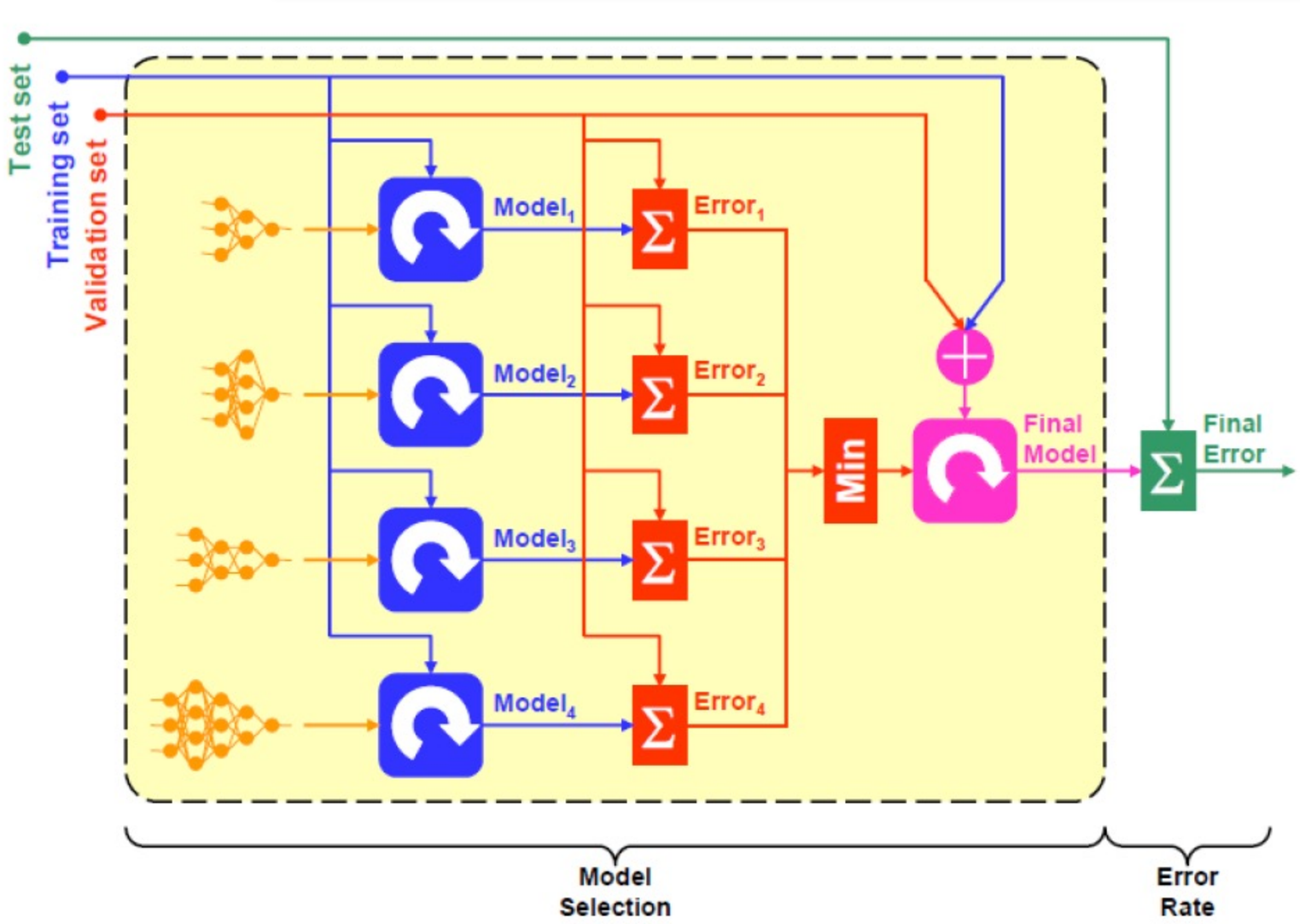
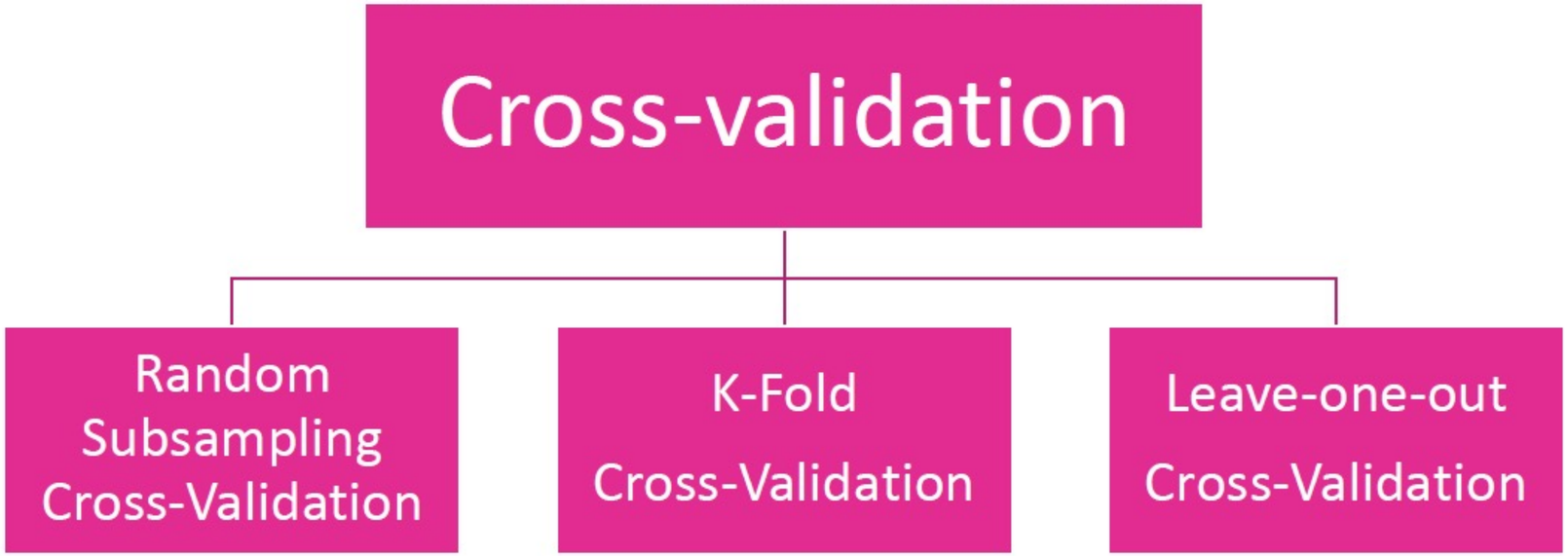


Figure by [R. Gutierrez-Osuna]

Train/Validation/Test

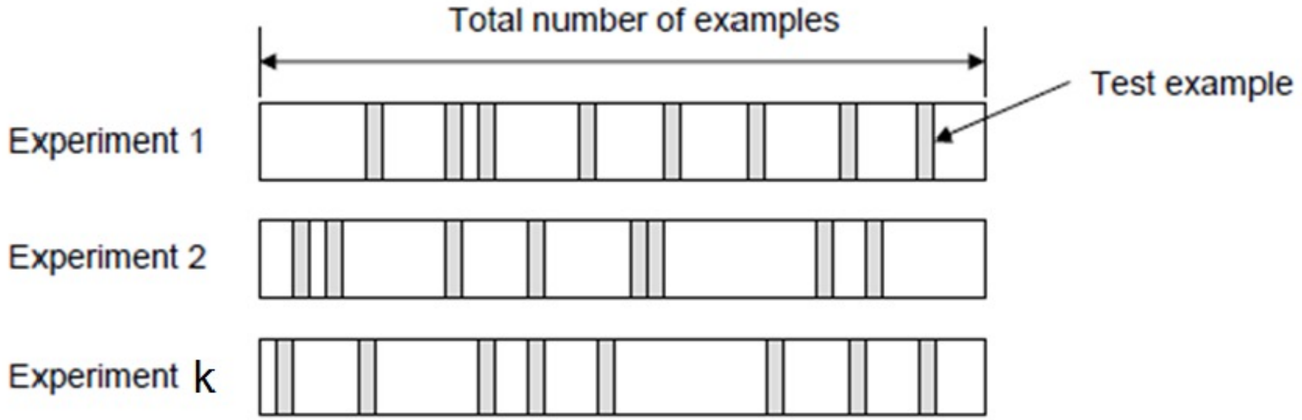
- Train/Validation/Test might have some issues:
  - If we have a **sparse training** dataset, we may not be able to select a portion of data only for validation/test
  - The test error may be an optimistic estimate of the true error due to an **fortunate/ unfortunate** split

To address these limitations:



Random Subsampling Cross-Validation

- Random Subsampling performs K data splits of the entire dataset
  - Each data split randomly selects a (fixed) number of examples without replacement
  - For each data split we retrain the classifier from scratch with the training examples and then estimate with the test examples



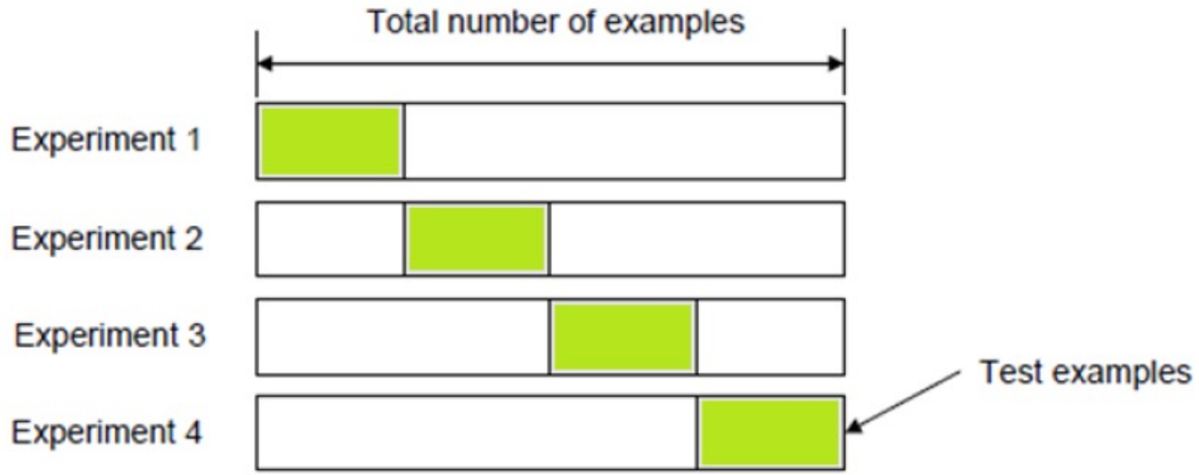
- The true error estimate is obtained as the average of the separate experiments
- Variance!!!

$$Er = \frac{1}{K}\sum_{i=1}^{K} Er_i$$

Figure by [R. Gutierrez-Osuna]

K-Fold Cross-Validation

- Create a K-fold partition of the dataset
  - For each of K experiments, use K-1 folds for training and a different fold for testing
  - For K=4



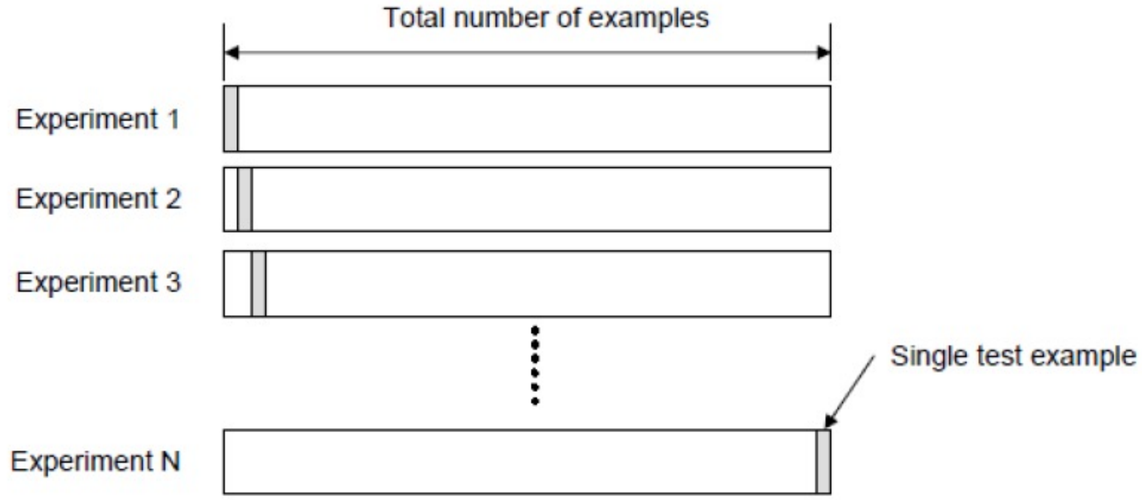
$$Er = \frac{1}{4}\sum_{i=1}^{4} Er_i$$

- The advantage of K-Fold Cross validation is that all the examples in the dataset are eventually used for both training and testing
- As before, the true error is estimated as the average error rate on test examples

Figure by [R. Gutierrez-Osuna]

Leave-One-Out Cross-Validation

- Leave-one-out is a K-Fold Cross Validation, where K is chosen as the total number of examples
  - For a dataset with N examples, perform N experiments
  - For each experiment use N-1 examples for training and the remaining example for testing



$$Er = \frac{1}{N}\sum_{i=1}^{N} Er_i$$

- As usual, the true error is estimated as the average error rate on test examples

Figure by [R. Gutierrez-Osuna]

Number of Folds

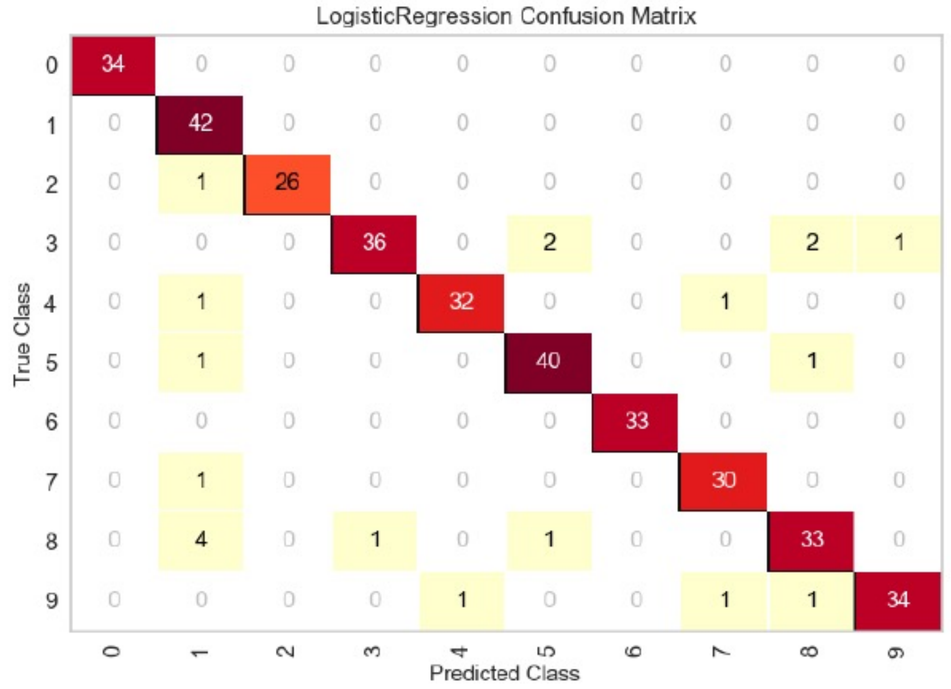
- In practice, the choice of the number of folds depends on the size of the dataset
  - For large datasets, even 3-Fold Cross Validation will be quite accurate
  - For very sparse datasets, we may have to use leave-one-out in order to train on as many examples as possible
- A common choice for K-Fold Cross Validation is K=10
- For better evaluation, **10-times-10-fold** can be used to have better estimation of the true error rate.
  - Repeat 10-fold cross validation 10 times (permute training data at each run)

Confusion Matrix

Two-Class Problem

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Multi-Class Problem



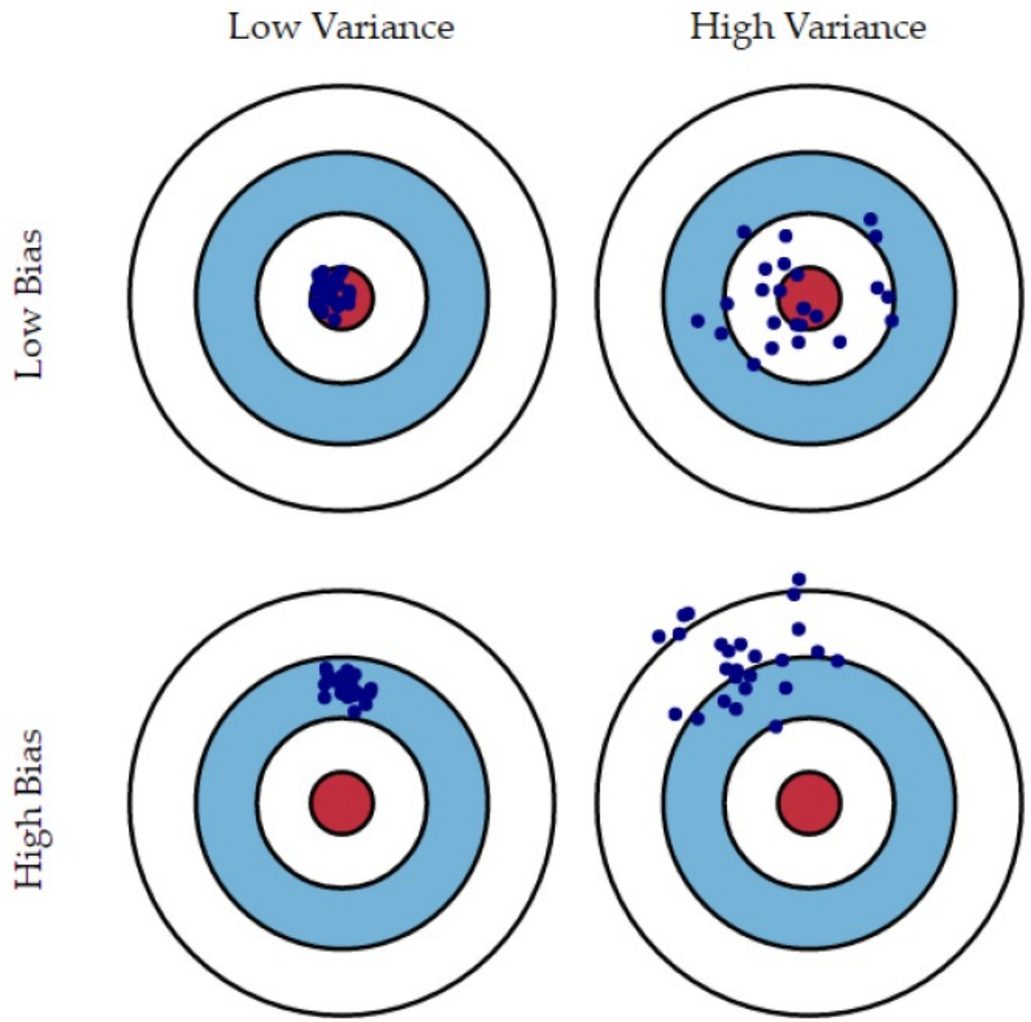
http://www.scikit-yb.org/en/latest/api/classifier/confusion_matrix.html

Bias/Variance

- How good an estimate is?
- Two measures of “goodness” are used for statistical estimates
  - BIAS: how close is the estimate to the true value?
    - Difference between the estimator's expected value and the true value of the parameter being estimated.
    - This is due to the classifier being "biased" due to a particular kind of solution (e.g. linear classifier) (underfitting)
    - An estimator with zero bias is called **unbiased**
  - VARIANCE: how much does the estimate change if you train on a different training set?
    - Error from sensitivity to small fluctuations in the training set.
    - This is due to the classifier being overfitted to a particular training set (overfitting)?

[R. Gutierrez-Osuna]

Bias/Variance



[Scott Formann-Roe]

Bias/Variance Trade-off

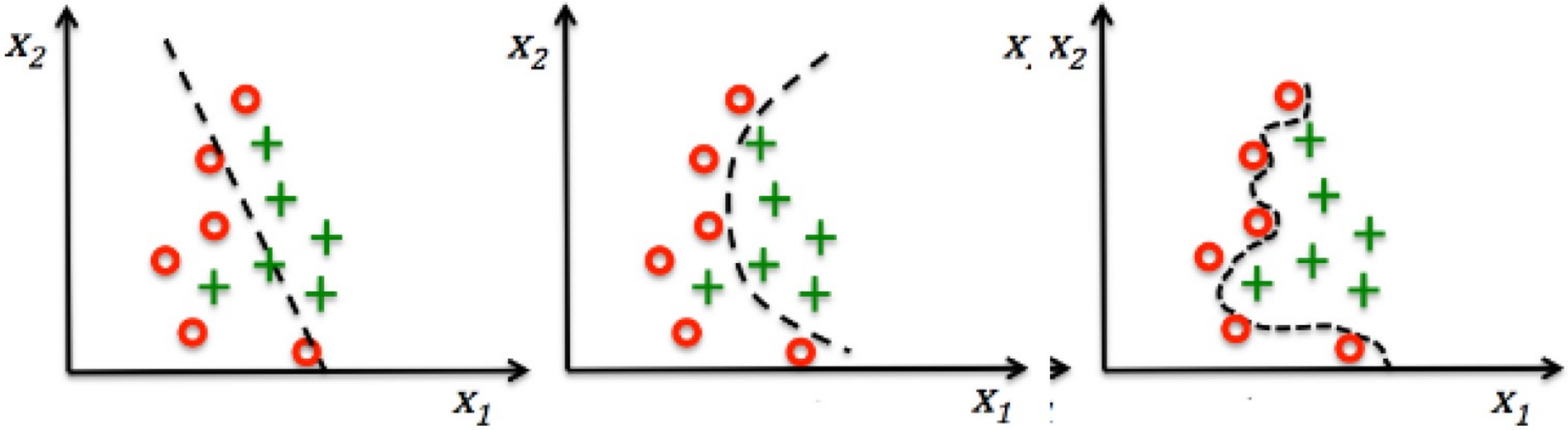
- The bias-variance tradeoff
  - Mostly, you can only decrease one of them at the expense of the other

- If you get bad results usually because of one of:
  - High bias - underfitting problem
  - High variance - overfitting problem

- Important to figure out which is the problem?
  - Knowing which will help let you improve the algorithm

Bias/Variance Classification

- Overfitting, Underfitting, Good Fit

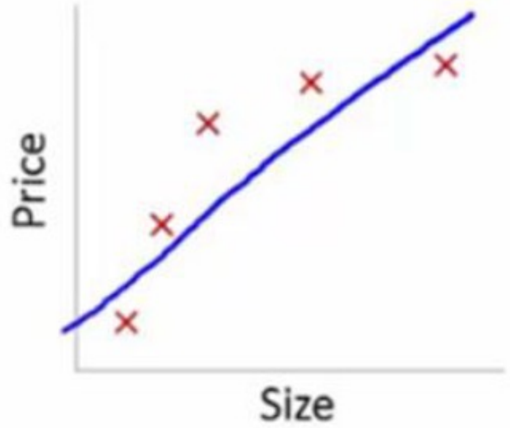


https://www.bogotobogo.com/python/scikit-learn/scikit_machine_learning_Bias-variance-Tradeoff.php

Bias/Variance for Regression

How select a good polynomial order?

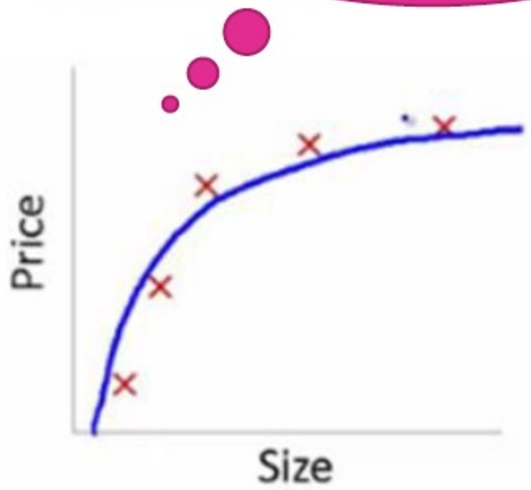
Model is too simple



$\theta_0 + \theta_1 x$

Underfitting

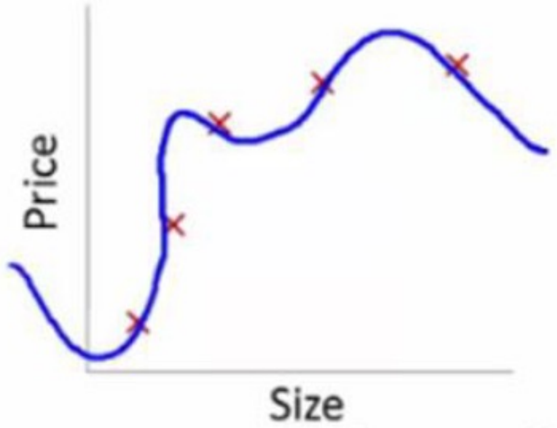
(High bias)



$\theta_0 + \theta_1 x + \theta_2 x^2$

Good fit

Model is too complex



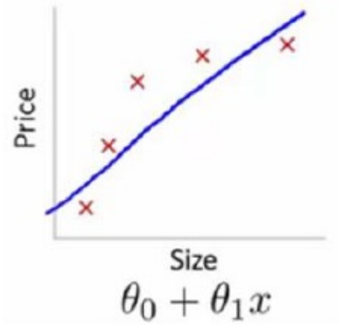
$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

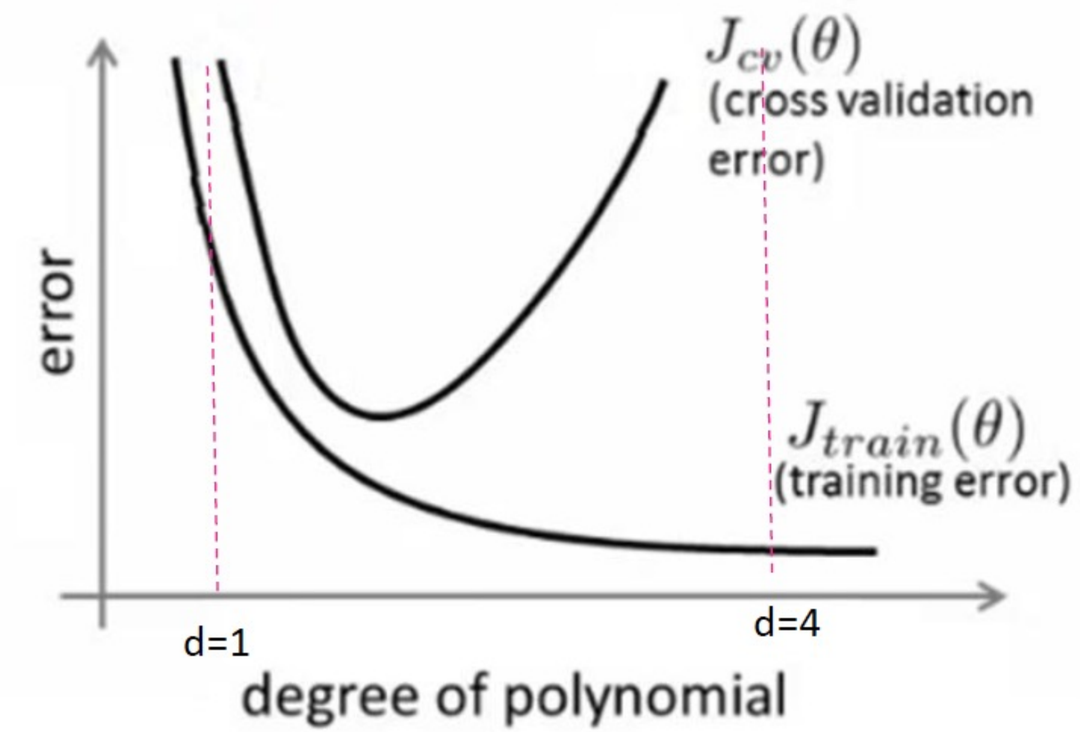
Overfitting

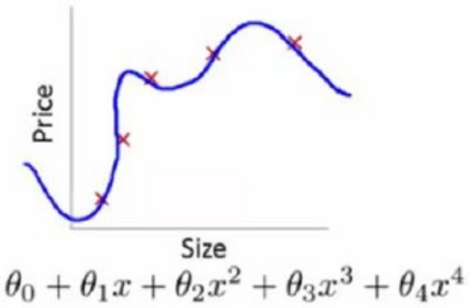
(High variance)

Diagnosis: bias vs variance

- Suppose your learning algorithm is performing less well than you were hoping. Is it a bias problem or a variance problem?



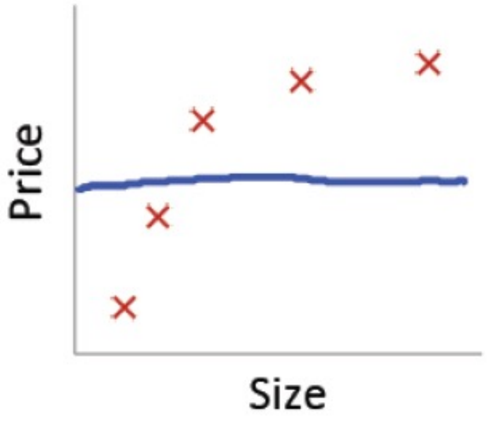




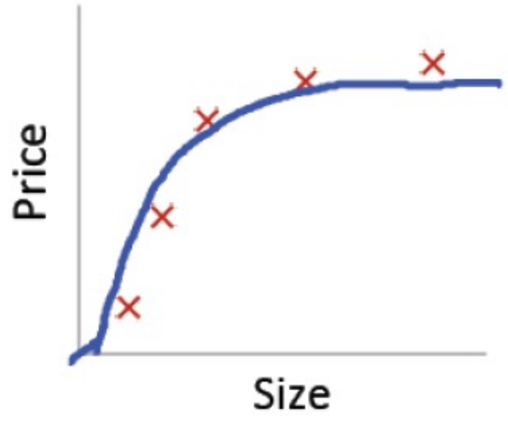
How is bias & variance effected by regularization?

Model $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + + \frac{\lambda}{2m}\sum_{j=0}^{n}\theta_j^2$$



Large λ
High bias (underfit)



Intermediate λ
"Just right"

Small λ
High variance (overfit)

Diagnosis: bias vs variance

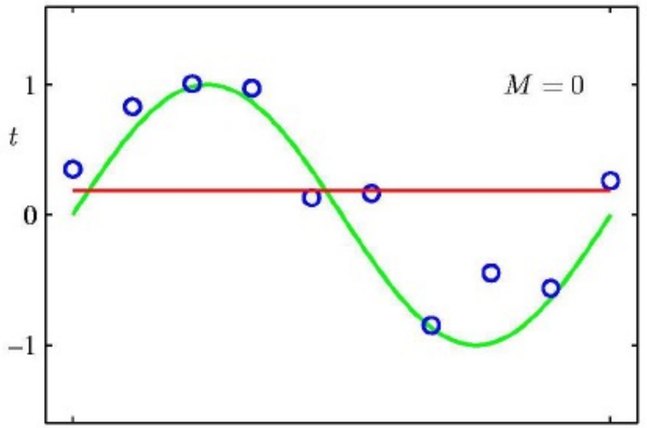
- if d is too small --> this probably corresponds to a high bias problem
- if d is too large --> this probably corresponds to a high variance problem

- For the high bias case, both cross validation and training error are high
  - Doesn't fit training data well
  - Doesn't generalize either

- For high variance, the cross validation error is high but training error is low
  - So we suffer from overfitting (training is low, cross validation is high)
  - i.e. training set fits well
  - But generalizes poorly

Addressing Underfit

- Underfitting may cause because:
  - The number of training data is not enough to learn the model
  - In iterative algorithm, the number of iteration on existing training data is not enough
  - More often, **the model is too simple** (the input features are not expressive enough) to describe the target well



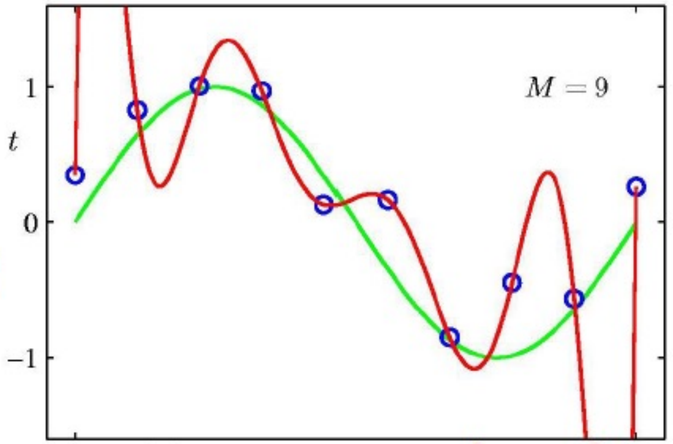
$h_\theta(x) = \theta_0$

- Addressing underfitting might achieved by:
  - Use more training data / Increase the number of iteration
  - Use more complex model (given that the number of training data are enough)
  - Add new features or change the feature types
  - Decrease the amount of regularization, if regularization is performed on the model (consider in next slides ….)

Addressing Overfit

- Overfitting may cause because:
  - the training is continued very much to achieve excellent accuracy on training data
  - the model is too complex

- Addressing Overfitting might achieved by:
  - Early stopping
  - Reduce the number of features (Manually or, Using feature selection algorithr
  - Reduce the model complexity (consider in next slides ….)
  - Use Regularization
    - Keep all features but reduce magnitude/values of parameters
    - Works well when we have a lot of features, each of which contributes a bit to predict y



$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$

So, conclusions

- So report the mean of the cross-validation accuracy along with the variance !!!

sources

Previous syde675 classes (Mohammad Javad Shafiee
-Octavia Camps, PSU
-www.cs.rit.edu/~rsg/BIIS_05lecture7.pp tby R.S.Gaborski Professor
-hebb.mit.edu/courses/9.641/lectures/pca.ppt by Sebastian Seung.
Cs189
Cs5785
Concise Machine Learning - Jonathan Richard Shewchuk
Russ Salakhutdinov – U. Toronto
B. Bernstein, NYU