Curse of Dimensionality, Dimensionality Reduction

Curse of Dimensionality: Overfitting

- If the number of features ***d*** is large, the number of samples ***n***, may be too small for accurate parameter estimation.
- For example, covariance matrix has d^2 parameters:

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \cdots & \sigma_{1d} \\ \vdots & \ddots & \vdots \\ \sigma_{d1} & \cdots & \sigma_d^2 \end{bmatrix}$$

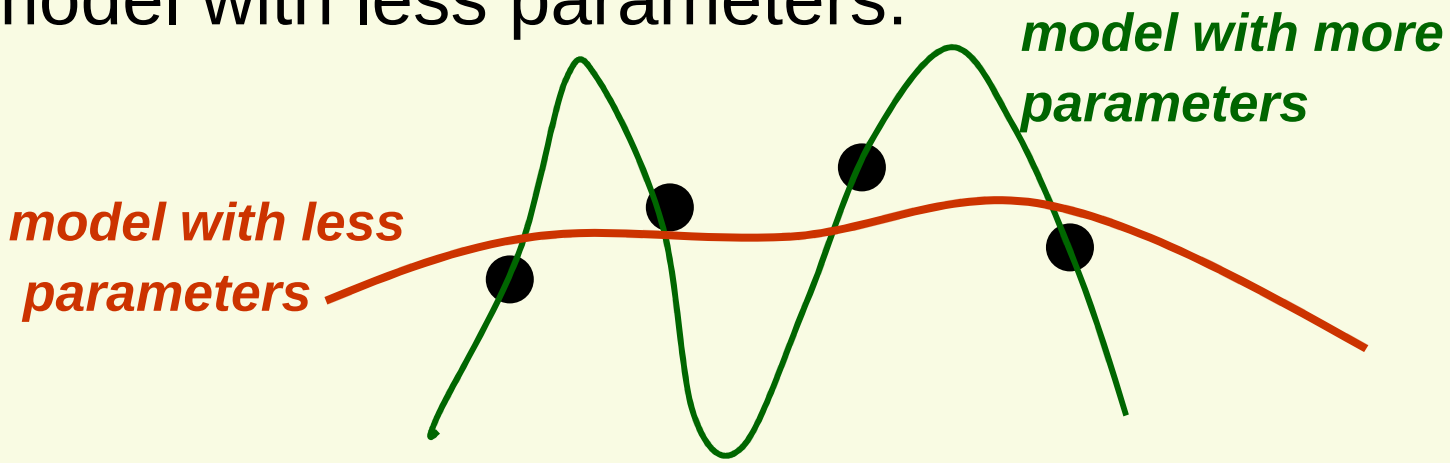
- For accurate estimation, ***n*** should be much bigger than d^2, otherwise model is too complicated for the data, ***overfitting***:

Curse of Dimensionality: Overfitting

- Paradox: If $n < d^2$ we are better off assuming that features are uncorrelated, even if we know this assumption is wrong
- In this case, the covariance matrix has only d parameters:

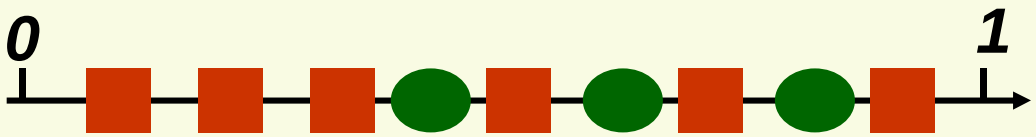
$$\Sigma = \begin{bmatrix} \sigma_1^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_d^2 \end{bmatrix}$$

- We are likely to avoid overfitting because we fit a model with less parameters:



Curse of Dimensionality: Number of Samples

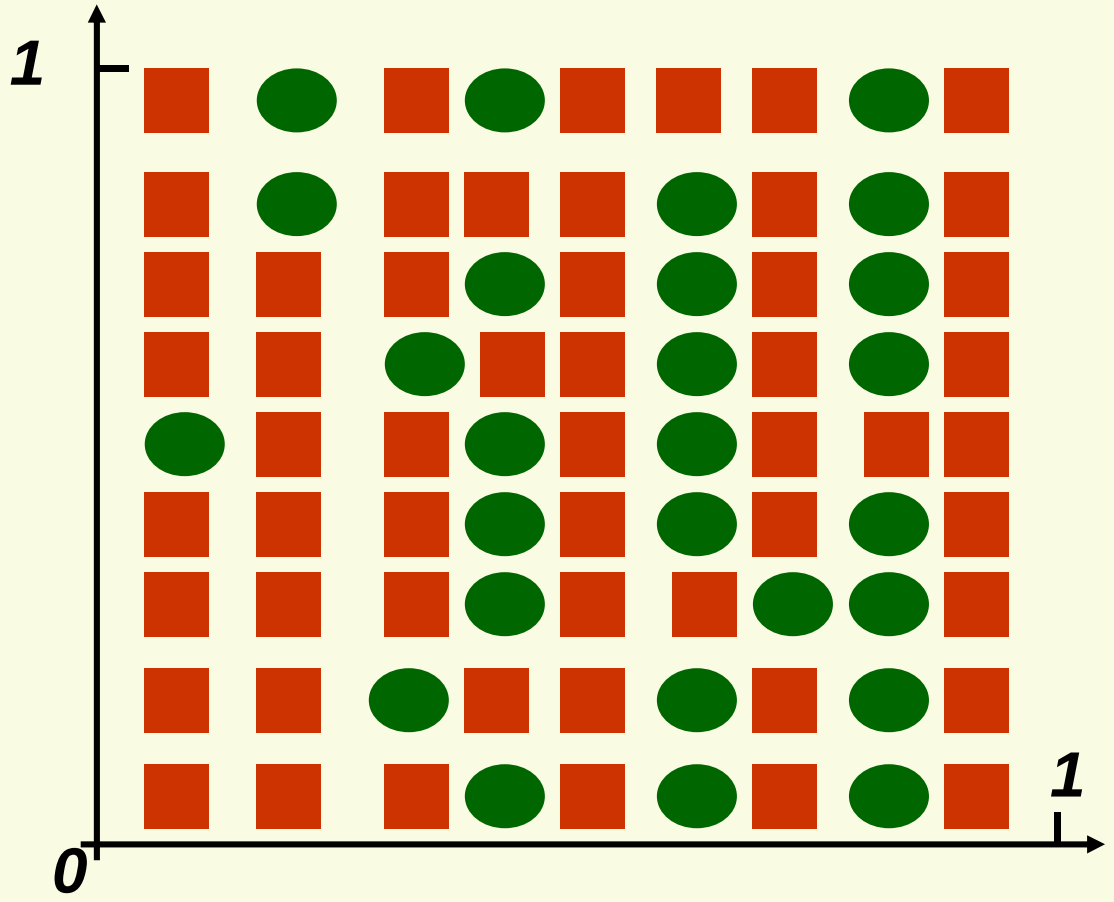
- Suppose we want to use the nearest neighbor approach with ***k*** = 1 (***1NN***)
- Suppose we start with only one feature



- This feature is not discriminative, i.e. it does not separate the classes well
- We decide to use 2 features. For the 1NN method to work well, need a lot of samples, i.e. samples have to be dense
- To maintain the same density as in 1D (9 samples per unit length), how many samples do we need?

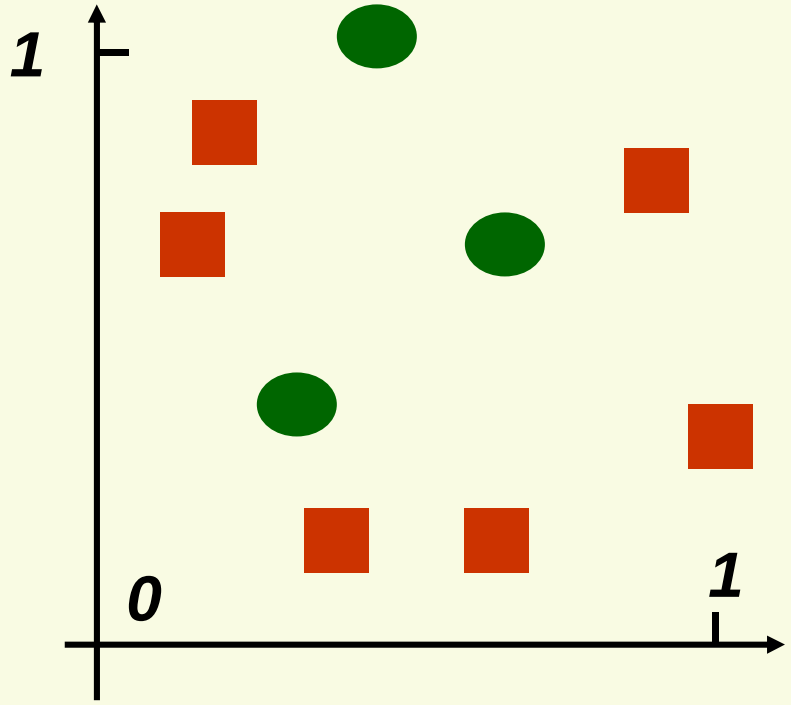
Curse of Dimensionality: Number of Samples

- We need 9^2 samples to maintain the same density as in ***1D***



Curse of Dimensionality: Number of Samples

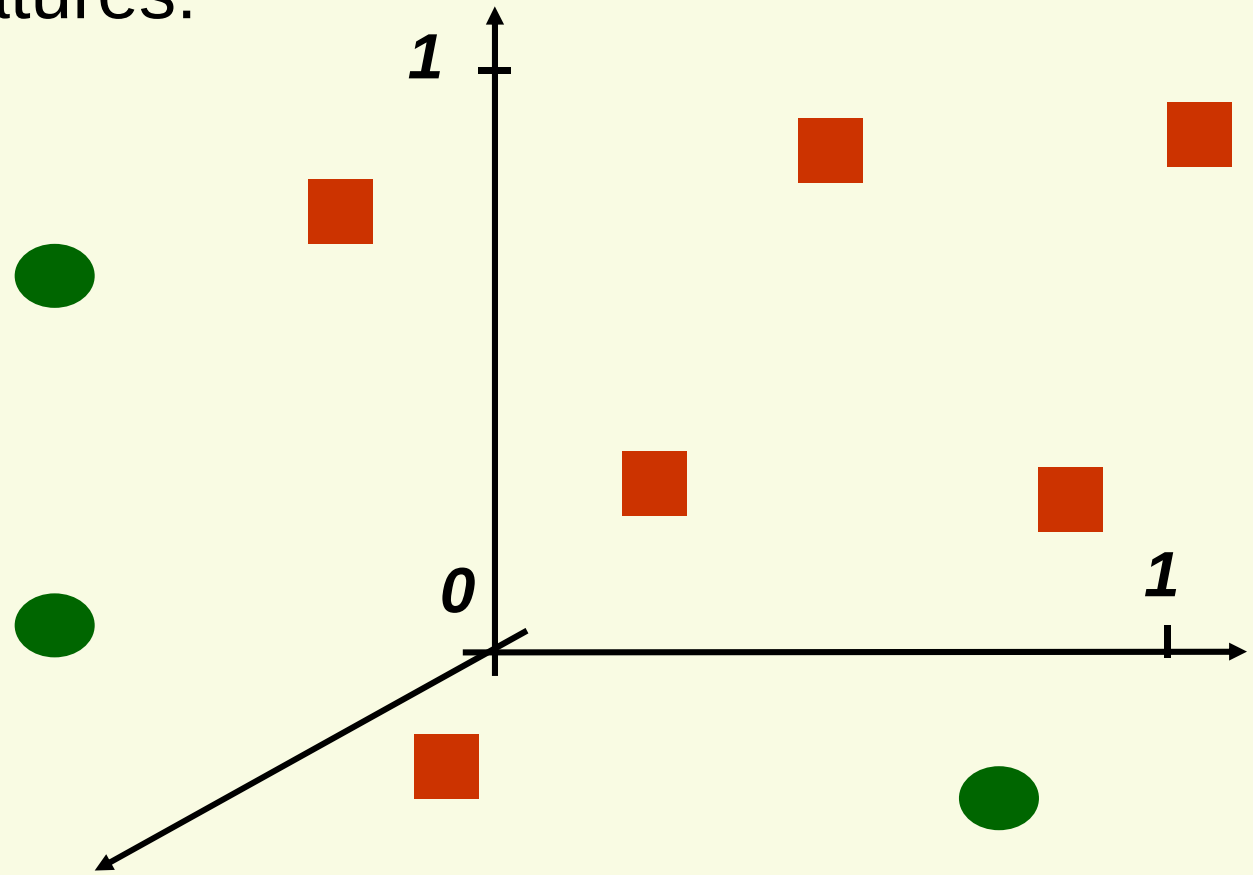
- Of course, when we go from 1 feature to 2, no one gives us more samples, we still have 9



- This is way too sparse for ***1NN*** to work well

Curse of Dimensionality: Number of Samples

- Things go from bad to worse if we decide to use 3 features:



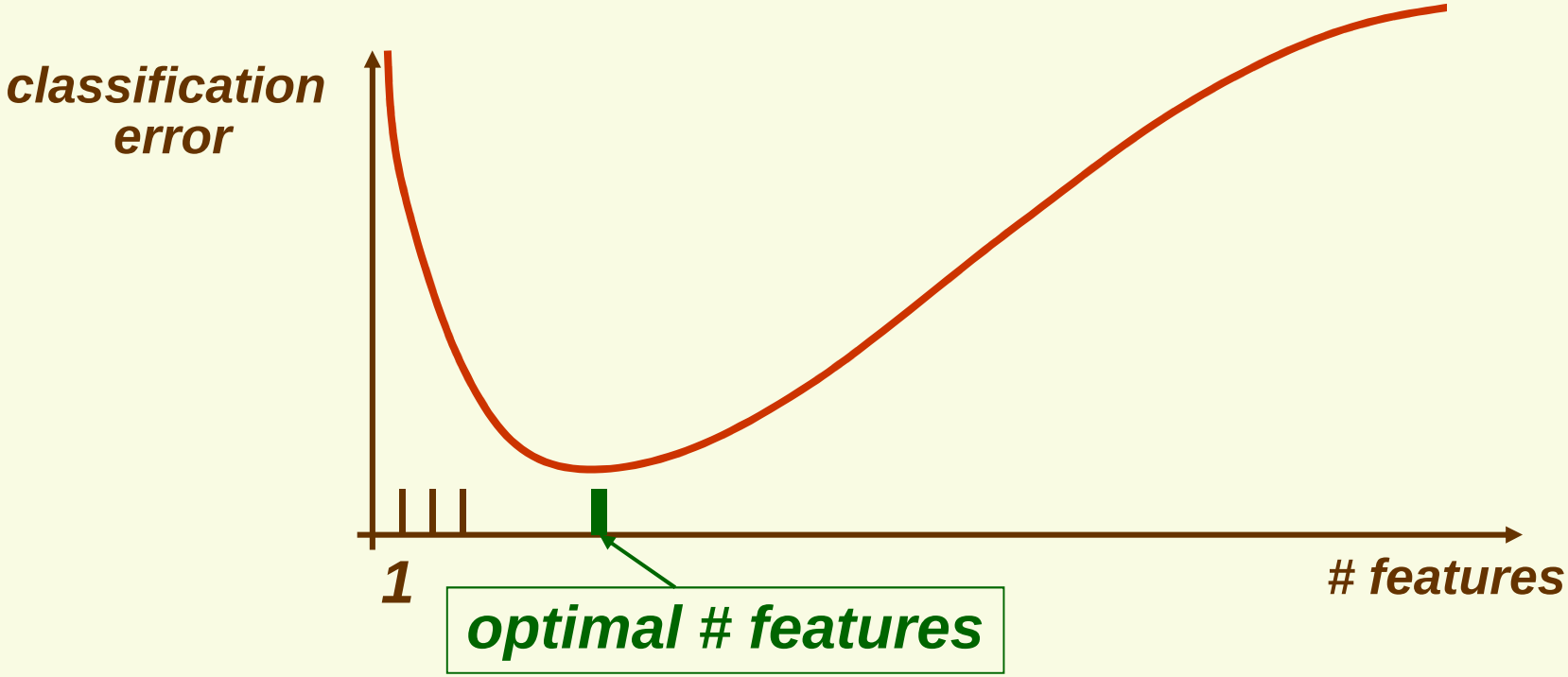
- If ***9*** was dense enough in 1D, in 3D we need $9^3=729$ samples!

Curse of Dimensionality: Number of Samples

- In general, if ***n*** samples is dense enough in ***1D***
- Then in ***d*** dimensions we need n^d samples!
- And n^d grows really really fast as a function of ***d***
- Common pitfall:
  - If we can't solve a problem with a few features, adding more features seems like a good idea
  - However the number of samples usually stays the same
  - The method with more features is likely to perform worse instead of expected better

Curse of Dimensionality: Number of Samples

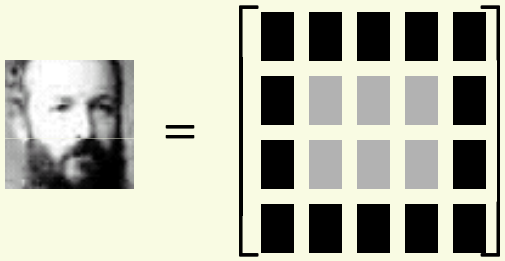
- For a fixed number of samples, as we add features, the graph of classification error:



- Thus for each fixed sample size ***n***, there is the optimal number of features to use

The Curse of Dimensionality

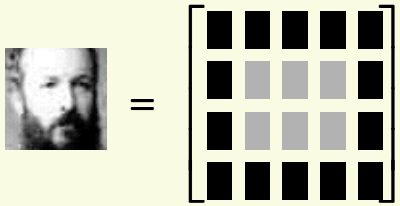
- We should try to avoid creating lot of features
- Often no choice, problem starts with many features
- Example: Face Detection
  - One sample point is ***k*** by ***m*** array of pixels



  - Feature extraction is not trivial, usually every pixel is taken as a feature
  - Typical dimension is 20 by 20 = 400
  - Suppose ***10*** samples are dense enough for 1 dimension. Need only $\boldsymbol{10^{400}}$ samples

The Curse of Dimensionality

- Face Detection, dimension of one sample point is ***km***



- The fact that we set up the problem with ***km*** dimensions (features) does not mean it is really a ***km***-dimensional problem
- Space of all ***k*** by ***m*** images has ***km*** dimensions
- Space of all ***k*** by ***m*** faces must be much smaller, since faces form a tiny fraction of all possible images
- Most likely we are not setting the problem up with the right features
- If we used better features, we are likely need much less than ***km***-dimensions

Dimensionality Reduction

- High dimensionality is challenging and redundant
- It is natural to try to reduce dimensionality
- Reduce dimensionality by feature combination: combine old features $\boldsymbol{x}$ to create new features $\boldsymbol{y}$

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \rightarrow \boldsymbol{f}\left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}\right) = \begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix} = \boldsymbol{y} \qquad \textit{with } k < d$$

- For example,

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \rightarrow \begin{bmatrix} x_1 + x_2 \\ x_3 + x_4 \end{bmatrix} = \boldsymbol{y}$$

- Ideally, the new vector $\boldsymbol{y}$ should retain from $\boldsymbol{x}$ all information important for classification

Dimensionality Reduction

- The best $\boldsymbol{f}(\boldsymbol{x})$ is most likely a non-linear function
- Linear functions are easier to find though
- For now, assume that $\boldsymbol{f}(\boldsymbol{x})$ is a linear mapping
- Thus it can be represented by a matrix $\boldsymbol{W}$:

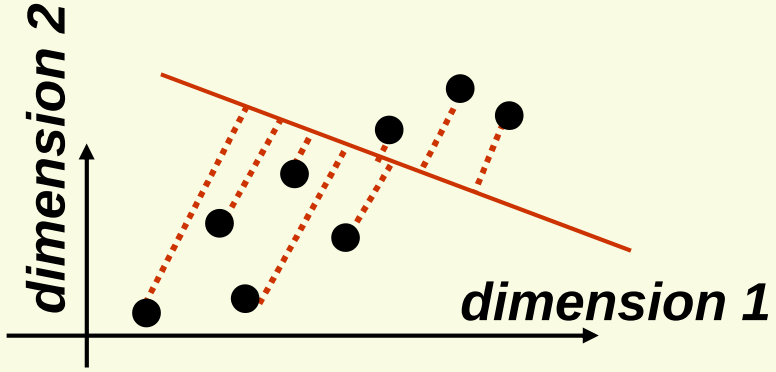
$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \Rightarrow W \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} = \begin{bmatrix} w_{11} & \cdots & w_{1d} \\ \vdots & & \vdots \\ w_{k1} & \cdots & w_{kd} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix} \quad \textit{with } k < d$$

Feature Combination

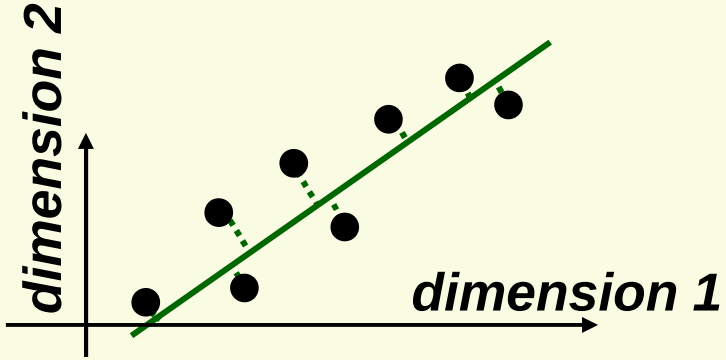
- We will look at 2 methods for feature combination
  - Principle Component Analysis (PCA)
  - Fischer Linear Discriminant (next lecture)

Principle Component Analysis (PCA)

- Main idea: seek most accurate data representation in a lower dimensional space
- Example in 2-D
  - Project data to 1-D subspace (a line) which minimize the projection error



large projection errors, bad line to project to

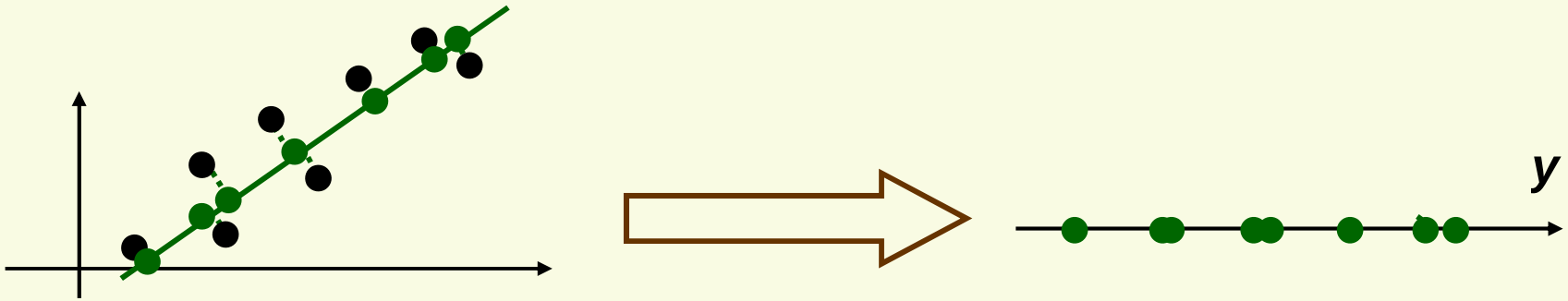


small projection errors, good line to project to

- Notice that the the good line to use for projection lies in the direction of largest variance

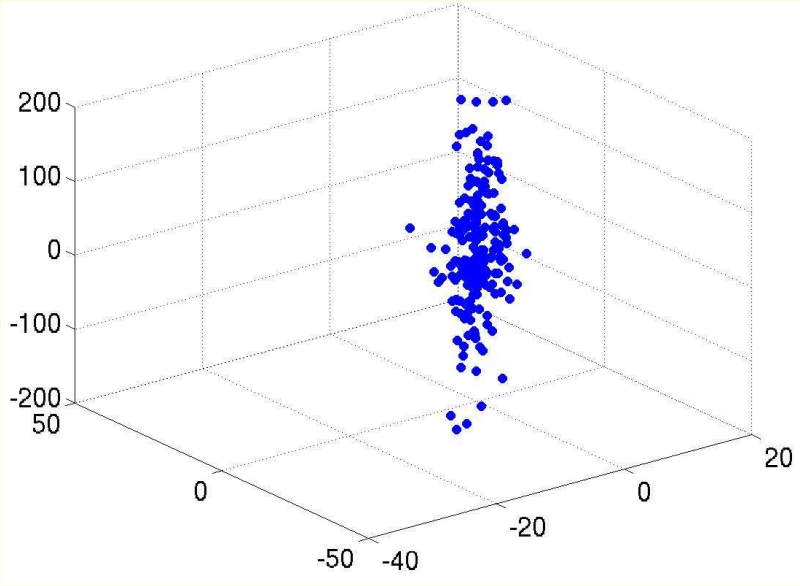
PCA

- After the data is projected on the best line, need to transform the coordinate system to get 1D representation for vector ***y***

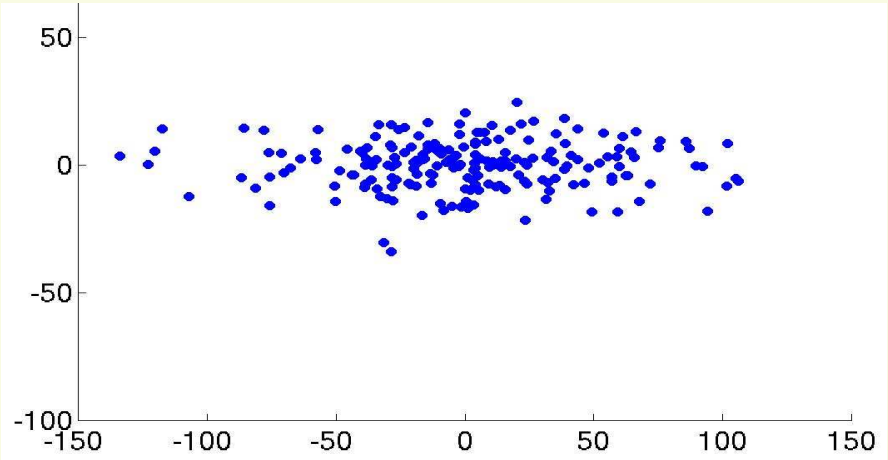


- Note that new data ***y*** has the same variance as old data ***x*** in the direction of the green line
- PCA preserves largest variances in the data. We will prove this statement, for now it is just an intuition of what PCA will do

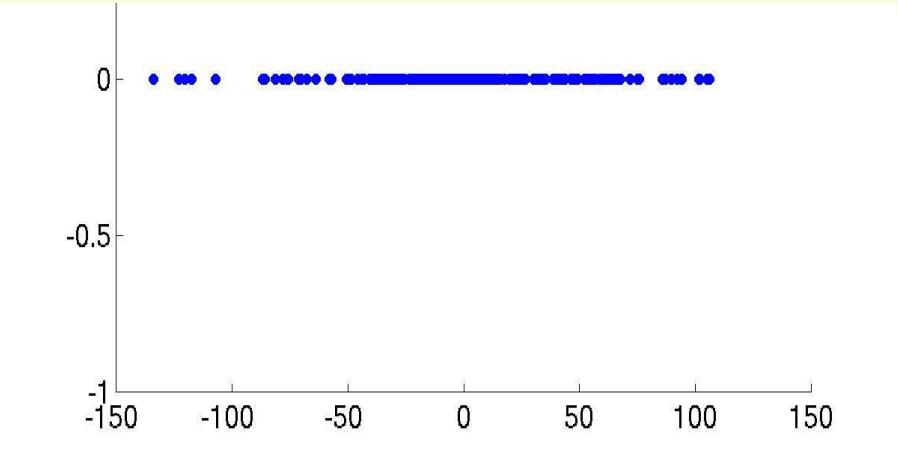
PCA: Approximation of Elliptical Cloud in 3D



best 2D approximation



best 1D approximation



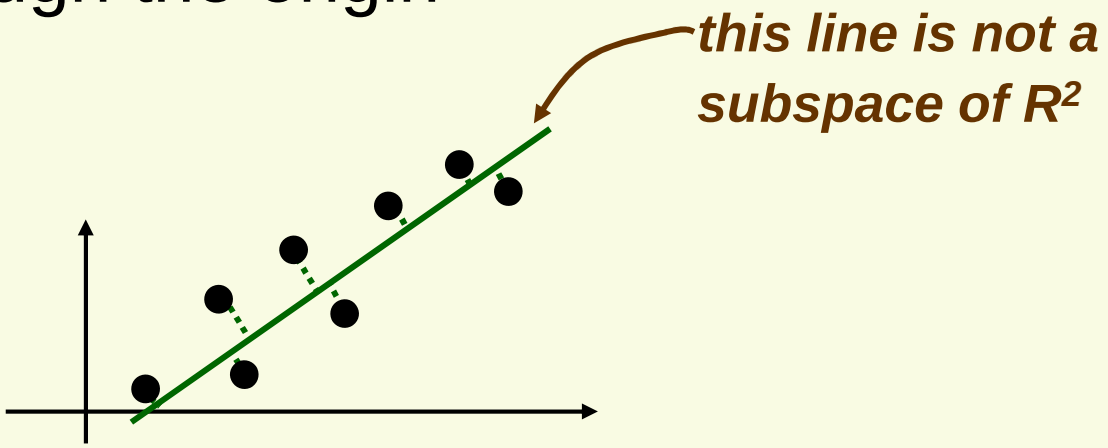
PCA: Linear Algebra for Derivation

- Let $\boldsymbol{V}$ be a $\boldsymbol{d}$ dimensional linear space, and $\boldsymbol{W}$ be a $\boldsymbol{k}$ dimensional linear subspace of $\boldsymbol{V}$
- We can always find a set of $\boldsymbol{d}$ dimensional vectors $\{\boldsymbol{e_1},\boldsymbol{e_2},\ldots,\boldsymbol{e_k}\}$ which forms an orthonormal basis for $\boldsymbol{W}$
  - $\langle \boldsymbol{e_i},\boldsymbol{e_j}\rangle = 0$ if $\boldsymbol{i}$ is not equal to $\boldsymbol{j}$ and $\langle \boldsymbol{e_i},\boldsymbol{e_i}\rangle = 1$
- Thus any vector in $\boldsymbol{W}$ can be written as

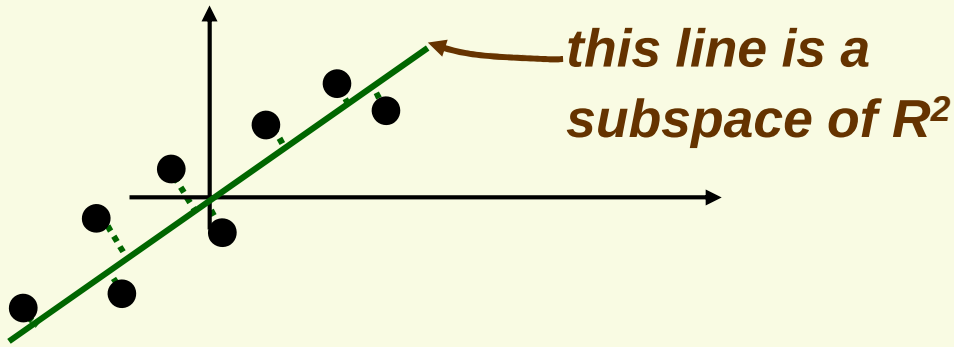
$$\alpha_1\boldsymbol{e}_1 + \alpha_2\boldsymbol{e}_2 + \ldots + \alpha_k\boldsymbol{e}_k = \sum_{i=1}^{k} \alpha_i\boldsymbol{e}_i \quad \textbf{for scalars} \quad \alpha_1,\ldots,\alpha_k$$

PCA: Linear Algebra for Derivation

- Recall that subspace ***W*** contains the zero vector, i.e. it goes through the origin



- For derivation, it will be convenient to project to subspace ***W***: thus we need to shift everything



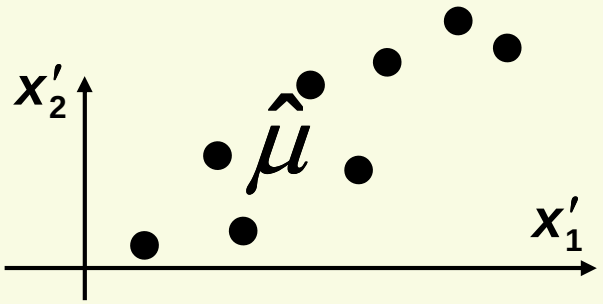
PCA Derivation: Shift by the Mean Vector

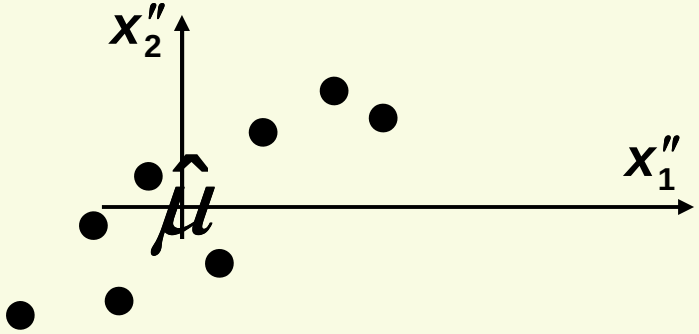
- Before PCA, subtract sample mean from the data

$$x - \frac{1}{n}\sum_{i=1}^{n} x_i = x - \hat{\mu}$$

- The new data has zero mean.

- All we did is change the coordinate system





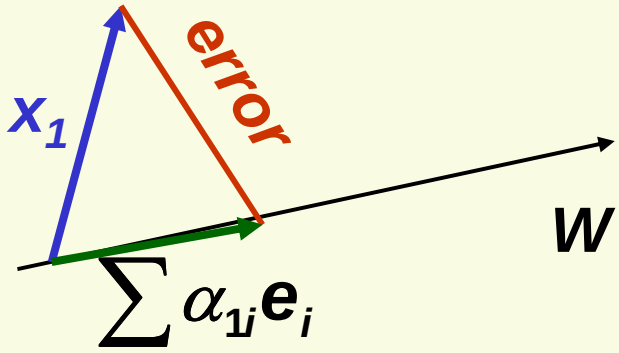
PCA: Derivation

- We want to find the most accurate representation of data $\boldsymbol{D}=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_n\}$ in some subspace $\boldsymbol{W}$ which has dimension $\boldsymbol{k} < \boldsymbol{d}$
- Let $\{\boldsymbol{e}_1,\boldsymbol{e}_2,\ldots,\boldsymbol{e}_k\}$ be the orthonormal basis for $\boldsymbol{W}$. Any vector in $\boldsymbol{W}$ can be written as $\sum_{i=1}^{k} \alpha_i \boldsymbol{e}_i$
- Thus $\boldsymbol{x}_1$ will be represented by some vector in $\boldsymbol{W}$

$$\sum_{i=1}^{k} \alpha_{1i} \boldsymbol{e}_i$$

- Error of this representation:

$$\boldsymbol{error} = \left\| \boldsymbol{x}_1 - \sum_{i=1}^{k} \alpha_{1i} \boldsymbol{e}_i \right\|^2$$



PCA: Derivation

- To find the total error, we need to sum over all $\boldsymbol{x}_j$'s
- Any $\boldsymbol{x}_j$ can be written as $\sum_{i=1}^{k} \alpha_{ji} \mathbf{e}_i$
- Thus the total error for representation of all data $\boldsymbol{D}$ is:

sum over all data points

$$J(\underbrace{\mathbf{e}_1,\ldots,\mathbf{e}_k,\alpha_{11},\ldots\alpha_{nk}}_{\textit{unknowns}}) = \sum_{j=1}^{n} \left\| \mathbf{x}_j - \sum_{i=1}^{k} \alpha_{ji}\mathbf{e}_i \right\|^2$$

error at one point

PCA: Derivation

- To minimize J, need to take partial derivatives and also enforce constraint that $\{e_1, e_2, \ldots, e_k\}$ are orthogonal

$$J(e_1,\ldots,e_k,\alpha_{11},\ldots\alpha_{nk}) = \sum_{j=1}^{n}\left\| x_j - \sum_{i=1}^{k}\alpha_{ji}e_i \right\|^2$$

- Let us simplify J first:

$$J(e_1,\ldots,e_k,\alpha_{11},\ldots\alpha_{nk}) = \sum_{j=1}^{n}\|x_j\|^2 - 2\sum_{j=1}^{n}\sum_{i=1}^{k}\alpha_{ji}x_j^t e_i + \sum_{j=1}^{n}\sum_{i=1}^{k}\alpha_{ji}^2$$

PCA: Derivation

$$J(\mathbf{e}_1,\ldots,\mathbf{e}_k,\alpha_{11},\ldots\alpha_{nk}) = \sum_{j=1}^{n}\left\|\mathbf{x}_j\right\|^2 - 2\sum_{j=1}^{n}\sum_{i=1}^{k}\alpha_{ji}\mathbf{x}_j^t\mathbf{e}_i + \sum_{j=1}^{n}\sum_{i=1}^{k}\alpha_{ji}^2$$

- First take partial derivatives with respect to α_{ml}

$$\frac{\partial}{\partial\alpha_{ml}}J(\mathbf{e}_1,\ldots,\mathbf{e}_k,\alpha_{11},\ldots\alpha_{nk}) = -2\mathbf{x}_m^t\mathbf{e}_l + 2\alpha_{ml}$$

- Thus the optimal value for α_{ml} is

$$-2\mathbf{x}_m^t\mathbf{e}_l + 2\alpha_{ml} = 0 \;\Rightarrow\; \alpha_{ml} = \mathbf{x}_m^t\mathbf{e}_l$$

PCA: Derivation

$$J(e_1,\ldots,e_k,\alpha_{11},\ldots\alpha_{nk}) = \sum_{j=1}^{n}\left\|x_j\right\|^2 - 2\sum_{j=1}^{n}\sum_{i=1}^{k}\alpha_{ji}x_j^t e_i + \sum_{j=1}^{n}\sum_{i=1}^{k}\alpha_{ji}^2$$

- Plug the optimal value for $\alpha_{ml} = x^t_m e_l$ back into J

$$J(e_1,\ldots,e_k) = \sum_{j=1}^{n}\left\|x_j\right\|^2 - 2\sum_{j=1}^{n}\sum_{i=1}^{k}\left(x_j^t e_i\right)x_j^t e_i + \sum_{j=1}^{n}\sum_{i=1}^{k}\left(x_j^t e_i\right)^2$$

- Can simplify J

$$J(e_1,\ldots,e_k) = \sum_{j=1}^{n}\left\|x_j\right\|^2 - \sum_{j=1}^{n}\sum_{i=1}^{k}\left(x_j^t e_i\right)^2$$

PCA: Derivation

$$J(e_1,...,e_k) = \sum_{j=1}^{n} \|x_j\|^2 - \sum_{j=1}^{n}\sum_{i=1}^{k} (x_j^t e_i)^2$$

- Rewrite J using $(a^t b)^2 = (a^t b)(a^t b) = (b^t a)(a^t b) = b^t(aa^t)b$

$$J(e_1,...,e_k) = \sum_{j=1}^{n} \|x_j\|^2 - \sum_{i=1}^{k} e_i^t \left(\sum_{j=1}^{n} (x_j x_j^t) \right) e_i$$

$$= \sum_{j=1}^{n} \|x_j\|^2 - \sum_{i=1}^{k} e_i^t S e_i$$

- Where $S = \sum_{j=1}^{n} x_j x_j^t$
- S is called the scatter matrix, it is just n-1 times the sample covariance matrix we have seen before

$$\hat{\Sigma} = \frac{1}{n-1} \sum_{j=1}^{n} (x_j - \hat{\mu})(x_j - \hat{\mu})^t$$

PCA: Derivation

$$J(e_1,...,e_k) = \sum_{j=1}^{n} \|x_j\|^2 - \sum_{i=1}^{k} e_i^t S e_i$$

constant

- Minimizing J is equivalent to maximizing $\sum_{i=1}^{k} e_i^t S e_i$
- We should also enforce constraints $e_i^t e_i = 1$ for all i
- Use the method of Lagrange multipliers, incorporate the constraints with undetermined λ_1 ,…, λ_k
- Need to maximize new function u

$$u(e_1,...,e_k) = \sum_{i=1}^{k} e_i^t S e_i - \sum_{j=1}^{k} \lambda_j (e_j^t e_j - 1)$$

PCA: Derivation

$$u(\mathbf{e}_1,\ldots,\mathbf{e}_k) = \sum_{i=1}^{k} \mathbf{e}_i^t \mathbf{S}\, \mathbf{e}_i - \sum_{j=1}^{k} \lambda_j \left(\mathbf{e}_j^t \mathbf{e}_j - 1\right)$$

- Compute the partial derivatives with respect to $\mathbf{e}_m$

$$\frac{\partial}{\partial \mathbf{e}_m} u(\mathbf{e}_1,\ldots,\mathbf{e}_k) = 2\mathbf{S}\mathbf{e}_m - 2\lambda_m \mathbf{e}_m = \mathbf{0}$$

Note: *$\mathbf{e}_m$ is a vector, what we are really doing here is taking partial derivatives with respect to each element of $\mathbf{e}_m$ and then arranging them up in a linear equation*

- Thus λ_m and $\mathbf{e}_m$ are eigenvalues and eigenvectors of scatter matrix $\mathbf{S}$

$$\mathbf{S}\mathbf{e}_m = \lambda_m \mathbf{e}_m$$

PCA: Derivation

$$J(e_1,...,e_k) = \sum_{j=1}^{n} \|x_j\|^2 - \sum_{i=1}^{k} e_i^t S e_i$$

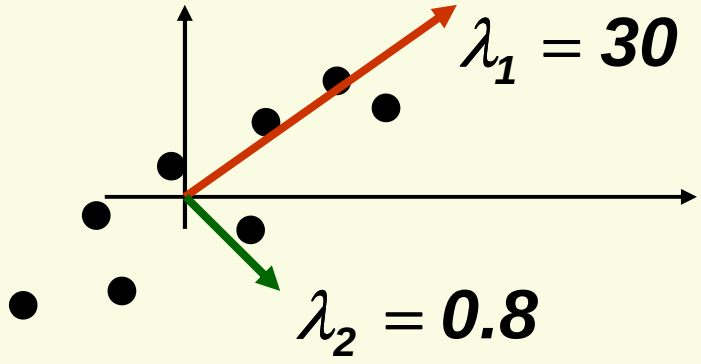
- Let's plug e_m back into J and use $Se_m = \lambda_m e_m$

$$J(e_1,...,e_k) = \sum_{j=1}^{n} \|x_j\|^2 - \sum_{i=1}^{k} \lambda_i \|e_i\|^2 = \underbrace{\sum_{j=1}^{n} \|x_j\|^2}_{\text{constant}} - \sum_{i=1}^{k} \lambda_i$$

- Thus to minimize J take for the basis of W the k eigenvectors of S corresponding to the k largest eigenvalues

PCA

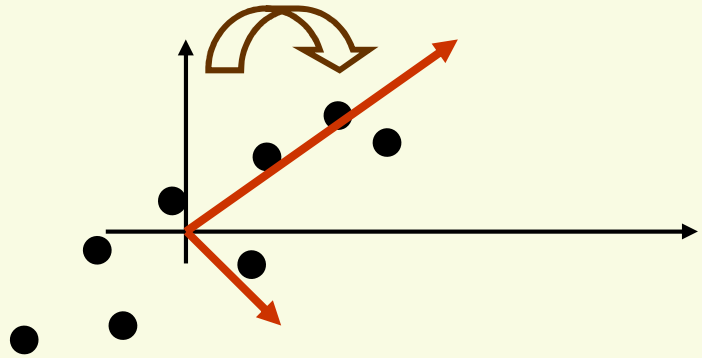
- The larger the eigenvalue of ***S***, the larger is the variance in the direction of corresponding eigenvector



- This result is exactly what we expected: project ***x*** into subspace of dimension ***k*** which has the largest variance
- This is very intuitive: restrict attention to directions where the scatter is the greatest

PCA

- Thus PCA can be thought of as finding new orthogonal basis by rotating the old axis until the directions of maximum variance are found



PCA as Data Approximation

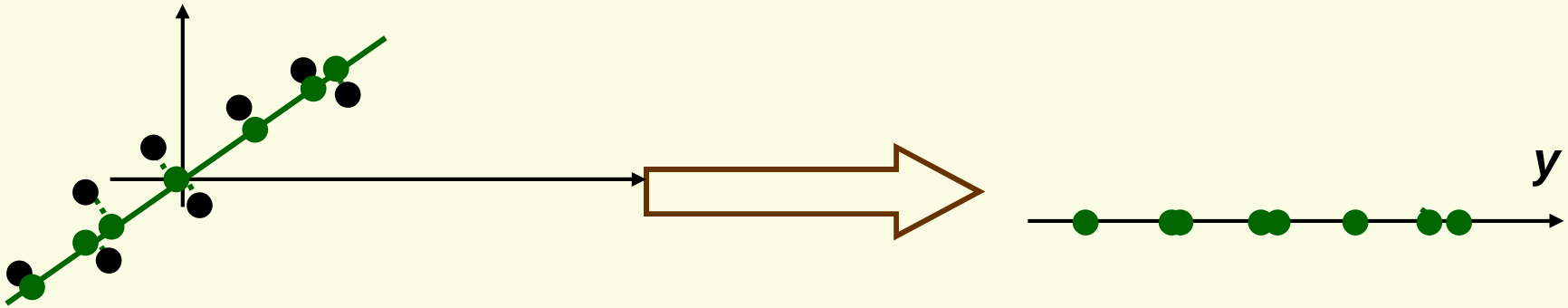
- Let $\{\boldsymbol{e_1}, \boldsymbol{e_2}, \ldots, \boldsymbol{e_d}\}$ be all $\boldsymbol{d}$ eigenvectors of the scatter matrix $\boldsymbol{S}$, sorted in order of decreasing corresponding eigenvalue
- Without any approximation, for any sample $\boldsymbol{x_i}$:

$$\boldsymbol{x}_i = \sum_{j=1}^{d} \alpha_j \boldsymbol{e}_j = \underbrace{\alpha_1 \boldsymbol{e}_1 + \ldots + \alpha_k \boldsymbol{e}_k}_{\text{approximation of } \boldsymbol{x}_i} + \overbrace{\alpha_{k+1} \boldsymbol{e}_{k+1} \ldots + \alpha_d \boldsymbol{e}_d}^{\text{error of approximation}}$$

- coefficients $\alpha_m = \boldsymbol{x}^t_i \boldsymbol{e}_m$ are called *principle components*
  - The larger $\boldsymbol{k}$, the better is the approximation
  - Components are arranged in order of importance, more important components come first
- Thus PCA takes the first $\boldsymbol{k}$ most important components of $\boldsymbol{x_i}$ as an approximation to $\boldsymbol{x_i}$

PCA: Last Step

- Now we know how to project the data
- Last step is to change the coordinates to get final ***k***-dimensional vector ***y***



- Let matrix $\boldsymbol{E} = [\boldsymbol{e}_1 \cdots \boldsymbol{e}_k]$
- Then the coordinate transformation is $\boldsymbol{y} = \boldsymbol{E}^t \boldsymbol{x}$
- Under $\boldsymbol{E}^t$, the eigenvectors become the standard basis:

$$\boldsymbol{E}^t \boldsymbol{e}_i = \begin{bmatrix} \boldsymbol{e}_1 \\ \vdots \\ \boldsymbol{e}_i \\ \vdots \\ \boldsymbol{e}_k \end{bmatrix} \boldsymbol{e}_i = \begin{bmatrix} \mathbf{0} \\ \vdots \\ \mathbf{1} \\ \vdots \\ \mathbf{0} \end{bmatrix}$$

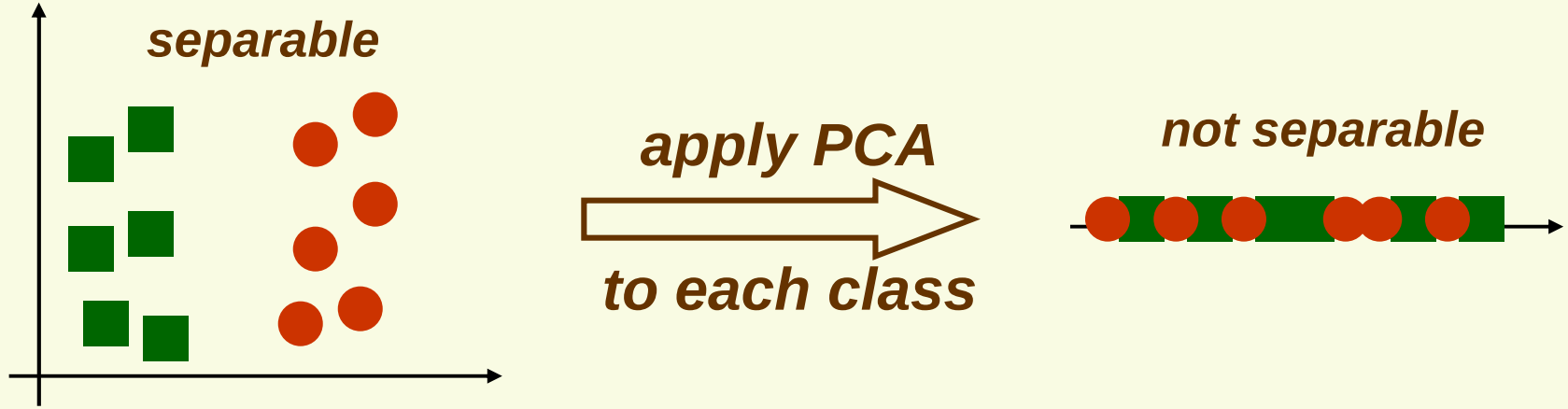
Recipe for Dimension Reduction with PCA

Data $\boldsymbol{D}=\{\boldsymbol{x_1},\boldsymbol{x_2},\ldots,\boldsymbol{x_n}\}$. Each $\boldsymbol{x_i}$ is a $\boldsymbol{d}$-dimensional vector. Wish to use PCA to reduce dimension to $\boldsymbol{k}$

1. Find the sample mean $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i$
2. Subtract sample mean from the data $\boldsymbol{z}_i = \boldsymbol{x}_i - \hat{\mu}$
3. Compute the scatter matrix $\boldsymbol{S} = \sum_{i=1}^{n} \boldsymbol{z}_i \boldsymbol{z}_i^t$
4. Compute eigenvectors $\boldsymbol{e_1},\boldsymbol{e_2},\ldots,\boldsymbol{e_k}$ corresponding to the $\boldsymbol{k}$ largest eigenvalues of $\boldsymbol{S}$
5. Let $\boldsymbol{e_1},\boldsymbol{e_2},\ldots,\boldsymbol{e_k}$ be the columns of matrix $\boldsymbol{E} = [\boldsymbol{e}_1 \cdots \boldsymbol{e}_k]$
6. The desired $\boldsymbol{y}$ which is the closest approximation to $\boldsymbol{x}$ is $\boldsymbol{y} = \boldsymbol{E}^t \boldsymbol{z}$

Data Representation vs. Data Classification

- PCA finds the most accurate *data representation* in a lower dimensional space
  - Project data in the directions of maximum variance
- However the directions of maximum variance may be useless for classification

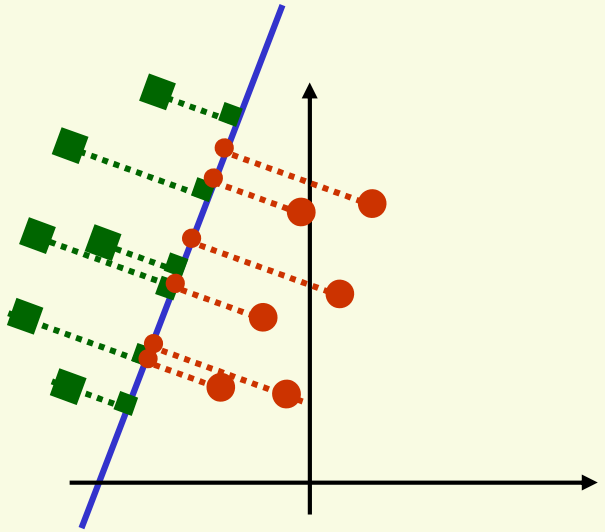


- Fisher Linear Discriminant projects to a line which preserves direction useful for *data classification*

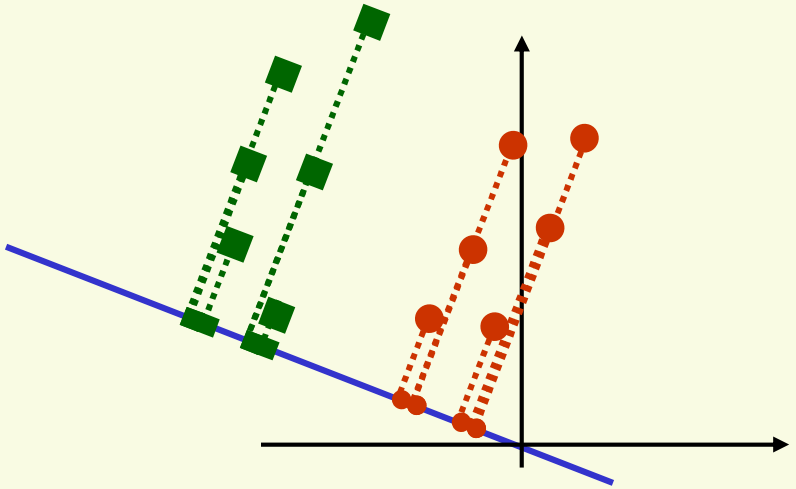
Fisher Linear Discriminant

- Main idea: find projection to a line s.t. samples from different classes are well separated

Example in 2D



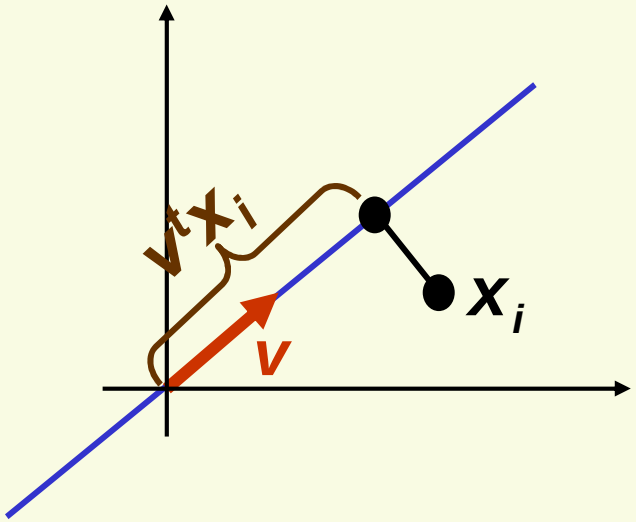
bad line to project to, classes are mixed up



good line to project to, classes are well separated

Fisher Linear Discriminant

- Suppose we have 2 classes and ***d***-dimensional samples $\boldsymbol{x_1},\ldots,\boldsymbol{x_n}$ where
  - $\boldsymbol{n_1}$ samples come from the first class
  - $\boldsymbol{n_2}$ samples come from the second class
- consider projection on a line
- Let the line direction be given by unit vector ***v***



- Thus the projection of sample $\boldsymbol{x_i}$ onto a line in direction ***v*** is given by $\boldsymbol{v^t x_i}$

Fisher Linear Discriminant

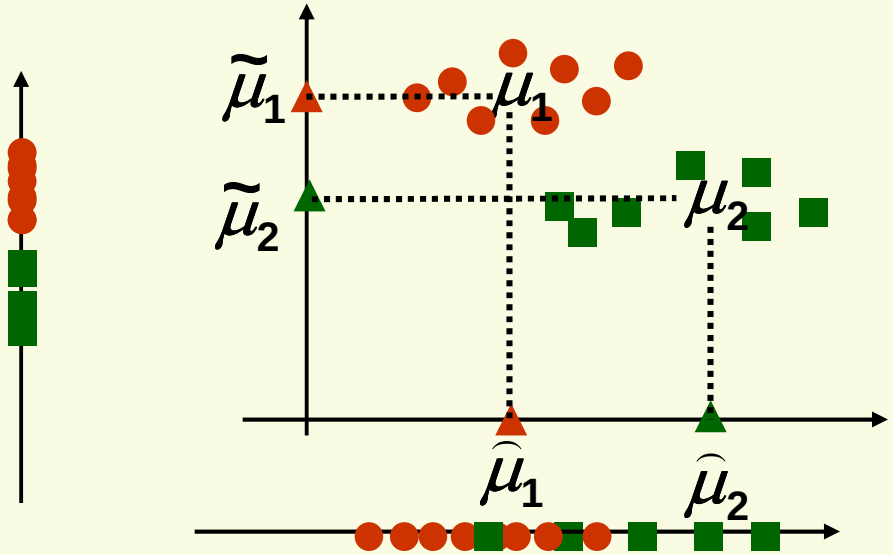
- How to measure separation between projections of different classes?
- Let $\tilde{\mu}_1$ and $\tilde{\mu}_2$ be the means of projections of classes 1 and 2
- Let μ_1 and μ_2 be the means of classes 1 and 2
- $|\tilde{\mu}_1 - \tilde{\mu}_2|$ seems like a good measure

$$\tilde{\mu}_1 = \frac{1}{n_1}\sum_{x_i \in C1}^{n_1} v^t x_i = v^t\left(\frac{1}{n_1}\sum_{x_i \in C1}^{n_1} x_i\right) = v^t \mu_1$$

***similarly*,** $\tilde{\mu}_2 = v^t \mu_2$

Fisher Linear Discriminant

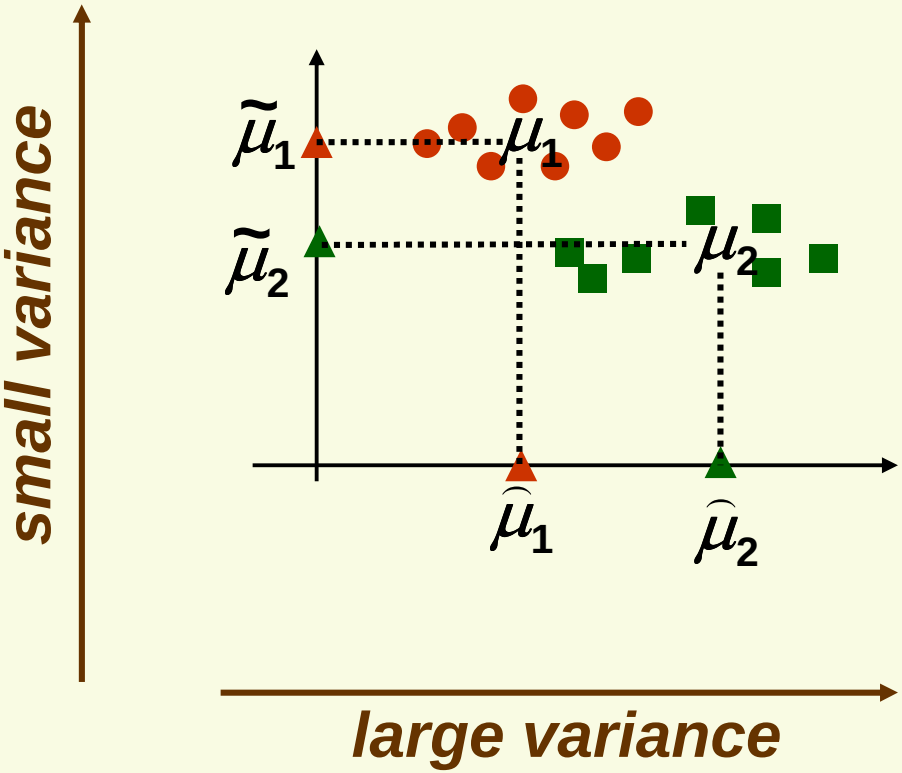
- How good is $|\tilde{\mu}_1 - \tilde{\mu}_2|$ as a measure of separation?
  - The larger $|\tilde{\mu}_1 - \tilde{\mu}_2|$, the better is the expected separation



- the vertical axes is a better line than the horizontal axes to project to for class separability
- however $|\hat{\mu}_1 - \hat{\mu}_2| > |\tilde{\mu}_1 - \tilde{\mu}_2|$

Fisher Linear Discriminant

- The problem with $\left|\tilde{\mu}_1 - \tilde{\mu}_2\right|$ is that it does not consider the variance of the classes

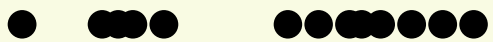


Fisher Linear Discriminant

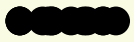
- We need to normalize $|\tilde{\mu}_1 - \tilde{\mu}_2|$ by a factor which is proportional to variance
- 1D samples $z_1,\ldots,z_n$. Sample mean is $\mu_z = \frac{1}{n}\sum_{i=1}^{n} z_i$
- Define their ***scatter*** as

$$s = \sum_{i=1}^{n} (z_i - \mu_z)^2$$

- Thus scatter is just sample variance multiplied by n
  - scatter measures the same thing as variance, the spread of data around the mean
  - scatter is just on different scale than variance



larger scatter



smaller scatter

Fisher Linear Discriminant

- Fisher Solution: normalize $|\tilde{\mu}_1 - \tilde{\mu}_2|$ by scatter
- Let $y_i = v^t x_i$, i.e. y_i 's are the projected samples
- Scatter for projected samples of class 1 is

$$\tilde{s}_1^2 = \sum_{y_i \in Class\ 1} (y_i - \tilde{\mu}_1)^2$$

- Scatter for projected samples of class 2 is

$$\tilde{s}_2^2 = \sum_{y_i \in Class\ 2} (y_i - \tilde{\mu}_2)^2$$

Fisher Linear Discriminant

- We need to normalize by both scatter of class 1 and scatter of class 2
- Thus Fisher linear discriminant is to project on line in the direction ***v*** which maximizes

want projected means are far from each other

$$J(\boldsymbol{v}) = \frac{(\tilde{\mu}_1 - \tilde{\mu}_2)^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$$

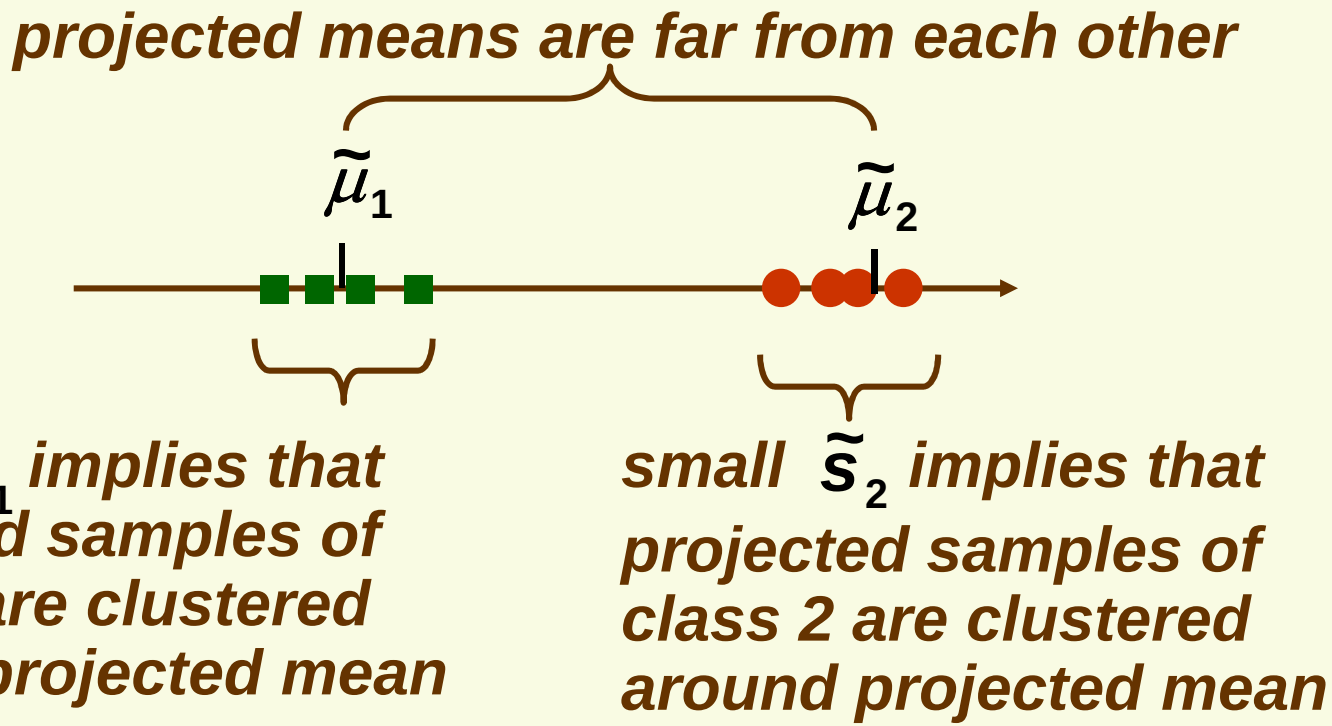
want scatter in class 1 to be as small as possible, i.e. samples of class 1 cluster around the projected mean $\tilde{\mu}_1$

want scatter in class 2 to be as small as possible, i.e. samples of class 2 cluster around the projected mean $\tilde{\mu}_2$

Fisher Linear Discriminant

$$J(\boldsymbol{v}) = \frac{(\tilde{\mu}_1 - \tilde{\mu}_2)^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$$

- If we find $\boldsymbol{v}$ which makes $J(\boldsymbol{v})$ large, we are guaranteed that the classes are well separated



Fisher Linear Discriminant Derivation

$$J(\boldsymbol{v}) = \frac{(\tilde{\mu}_1 - \tilde{\mu}_2)^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$$

- All we need to do now is to express $\boldsymbol{J}$ explicitly as a function of $\boldsymbol{v}$ and maximize it
  - straightforward but need linear algebra and Calculus (the derivation is shown in the next few slides.)
  - The solution is found by ***generalized eigenvalue problem*** $\Rightarrow \boxed{\boldsymbol{S}_B\boldsymbol{v} = \lambda \boldsymbol{S}_W\boldsymbol{v}}$

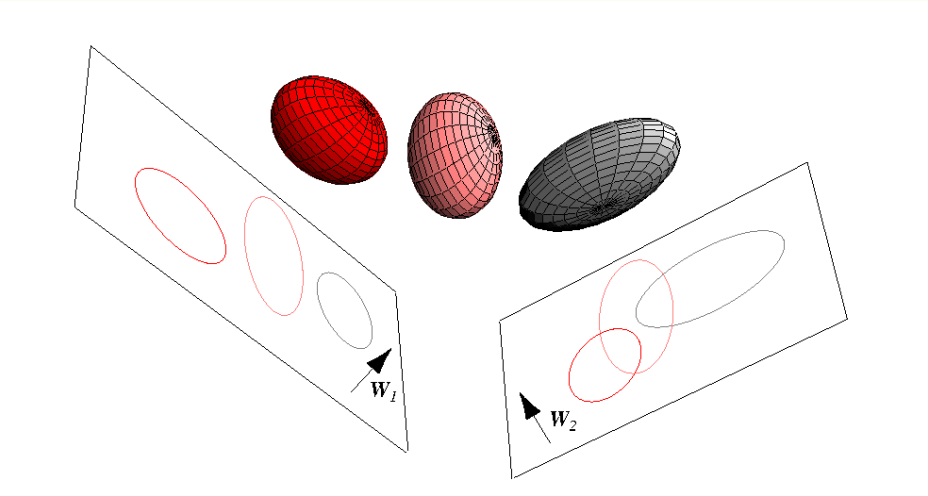
between class scatter matrix $\boldsymbol{S}_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^t$

within the class scatter matrix $\boldsymbol{S}_W = \boldsymbol{S}_1 + \boldsymbol{S}_2$

$$\boldsymbol{S}_1 = \sum_{x_i \in Class\ 1} (\boldsymbol{x}_i - \mu_1)(\boldsymbol{x}_i - \mu_1)^t \qquad \boldsymbol{S}_2 = \sum_{x_i \in Class\ 2} (\boldsymbol{x}_i - \mu_2)(\boldsymbol{x}_i - \mu_2)^t$$

Multiple Discriminant Analysis (MDA)

- Can generalize FLD to multiple classes
- In case of ***c*** classes, can reduce dimensionality to 1, 2, 3,…, ***c***-1 dimensions
- Project sample $\boldsymbol{x}_i$ to a linear subspace $\boldsymbol{y}_i = \boldsymbol{V}^t\boldsymbol{x}_i$
  - ***V*** is called projection matrix



Multiple Discriminant Analysis (MDA)

- Let
  - n_i by the number of samples of class i
  - and μ_i be the sample mean of class i
  - μ be the total mean of all samples

$$\mu_i = \frac{1}{n_i} \sum_{x \in class\ i} x \qquad \mu = \frac{1}{n} \sum_{x_i} x_i$$

- Objective function: $J(V) = \frac{det\left(V^t S_B V\right)}{det\left(V^t S_W V\right)}$
  - within the class scatter matrix S_W is

$$S_W = \sum_{i=1}^{c} S_i = \sum_{i=1}^{c} \sum_{x_k \in class\ i} (x_k - \mu_i)(x_k - \mu_i)^t$$

  - between the class scatter matrix S_B is

$$S_B = \sum_{i=1}^{c} n_i (\mu_i - \mu)(\mu_i - \mu)^t$$

maximum rank is c -1

Multiple Discriminant Analysis (MDA)

- Objective function:

$$J(V) = \frac{det\left(V^t S_B V\right)}{det\left(V^t S_W V\right)}$$

- It can be shown that "scatter" of the samples is directly proportional to the determinant of the scatter matrix
  - the larger ***det***(S), the more scattered samples are
  - ***det***(S) is the product of eigenvalues of S
- Thus we are seeking transformation ***V*** which maximizes the between class scatter and minimizes the within-class scatter

Multiple Discriminant Analysis (MDA)

$$J(V) = \frac{det\left(V^t S_B V\right)}{det\left(V^t S_W V\right)}$$

- First solve the ***generalized eigenvalue*** problem:

$$S_B v = \lambda S_W v$$

- At most ***c-1*** distinct solution eigenvalues
- Let $v_1, v_2, \ldots, v_{c-1}$ be the corresponding eigenvectors
- The optimal projection matrix ***V*** to a subspace of dimension ***k*** is given by the eigenvectors corresponding to the largest ***k*** eigenvalues
- Thus can project to a subspace of dimension at most ***c-1***