INF 5300
Feature selection and principal component analysis

Anne Solberg (anne@ifi.uio.no)

Today:

- Feature normalization
- Feature selection
- Feature transformation through principal component analysis

Next lecture:

- Fisher's linear discriminant function

Curriculum

- The lecture is based on the following sections from "Pattern Recognition" by S. Theodoridis and K. Koutroumbas:
- 5.1
- 5.2.2 Feature normalization
- 5.5.3 Scatter matrices
- 5.6 Feature subset selection
- 5.7 Fisher's linear discriminant function (next lecture)
- 6.1-6.3 Principal component analysis

Reminder - Basic classification principles

Classification task:

- Classify object $x = \{x_1, \ldots, x_n\}$ to one of the R classes $\omega_1, \ldots \omega_R$
- *Decision rule* d(**x**)=ω_r divides the feature space into R disjoint subsets K_r, r=1,…R.
- The borders between subsets K_r, r=1,…R are defined by R scalar *discrimination functions* $g_1(\mathbf{x}), \ldots, g_R(\mathbf{x})$
- The discrimination functions must satisfy:

  $g_r(\mathbf{x}) \geq g_s(\mathbf{x})$, $s \neq r$, for all $\mathbf{x} \in K_r$
- Discrimination hypersurfaces are thus defined by

  $g_r(\mathbf{x}) - g_s(\mathbf{x}) = 0$
- The pattern x will be classified to the class whose discrimination function gives a maximum:

$$d(\mathbf{x}) = \omega_r \Leftrightarrow g_r(\mathbf{x}) = \max_{s=1,\ldots R} g_s(\mathbf{x})$$

Reminder - Bayesian classification

- Prior probabilities $P(\omega_r)$ for each class
- Bayes classification rule: classify a pattern **x** to the class with the highest posterior probability $P(\omega_r|\mathbf{x})$

$$P(\omega_r|\mathbf{x}) = \max_{s=1,\ldots R} P(\omega_s|\mathbf{x})$$

- $P(\omega_s|\mathbf{x})$ is computed using Bayes formula

$$P(\omega_s \mid x) = \frac{p(x \mid \omega_s) P(\omega_s)}{p(x)}$$

$$p(x) = \sum_{s=1}^{R} p(x \mid \omega_s) P(\omega_s)$$

- $p(\mathbf{x}|\ \omega_s)$ is the class-conditional probability density for a given class.

Reminder - Classification with Gaussian distributions

- Probability distribution for n-dimensional Gaussian vector:

$$p(x \mid \omega_s) = \frac{1}{(2\pi)^{d/2} \left|\Sigma_s\right|^{1/2}} \exp\left[-\frac{1}{2}(x-\mu_s)^t \Sigma_s^{-1} (x-\mu_s)\right]$$

$$\hat{\mu}_s = \frac{1}{M_s} \sum_{m=1}^{M_s} x_m,$$

$$\hat{\Sigma}_s = \frac{1}{M_s} \sum_{m=1}^{M_s} (x_m - \hat{\mu}_s)(x_m - \hat{\mu}_s)^t$$

where the sum is over all training samples belonging to class s

- μ_s and Σ_s are not known, but they are estimated from M training samples as the Maximum Likelihood estimates

The curse of dimensionality

- Assume we have S classes and a n-dimensional feature vector.
- With a fully multivariate Gaussian model, we must estimate S different mean vectors and S different covariance matrices from training samples.

  $\hat{\mu}_s$ has n elements

  $\hat{\Sigma}_s$ has n(n-1)/2 elements

- Assume that we have M_s training samples from each class
- Given M_s, there is a maximum of the achieved classification performance for a certain value of n (increasing n beyond this limit will lead to worse performance after a certain).
- Adding more features is not always a good idea!
- If we have limited training data, we can use diagonal covariance matrices or regularization.

How do we beat the ”curse of dimensionality”?

- Use regularized estimates for the Gaussian case
  - Use diagonal covariance matrices
  - Apply regularized covariance estimation
- Generate few, but informative features
  - Careful feature design given the application
- Reducing the dimensionality
  - Feature selection
  - Feature transforms

Regularized covariance matrix estimation

- Let the covariance matrix be a weighted combination of a class-specific covariance matrix Σ_k and a common covariance matrix Σ:

$$\Sigma_k(\alpha) = \frac{(1-\alpha)n_k\Sigma_k + \alpha n\Sigma}{(1-\alpha)n_k + \alpha n}$$

  der $0 \leq \alpha \leq 1$ must be determined, and n_k and n is the number of training samples for class k and overall.

- Alternatively:

$$\Sigma_k(\beta) = (1-\beta)\Sigma_k + \beta I$$

  where the parameter $0 \leq \beta \leq 1$ must be determined.

Feature selection

- Given a large set of N features, how do we select the best subset of m features?
  - How do we select m?
  - Finding the best combination of m features out a N possible is a large optimization problem.
  - Full search is normally not possible.
  - Suboptimal approaches are often used.
  - How many features are needed?
- Alternative: compute lower-dimensional projections of the N-dimensional space
  - PCA
  - Fisher's linear discriminant
  - Projection pursuit and other non-linear approaches

Preprocessing - data normalization

- Features may have different ranges
  - Feature 1 has range $f1_{min}$-$f1_{max}$
  - Feature n has range fn_{min}-fn_{max}
  - This does not reflect their significance in classification performance!
  - Example: minimum distance classifier uses Euclidean distance
    - Features with large absolute values will dominate the classifier

Feature normalization

- Normalize all features to have the same mean and variance.
- Data set with N objects and K features
- Features x_{ik}, i=1…N, k=1,…K

Zero mean, unit variance:

$$\bar{x}_k = \frac{1}{N}\sum_{i=i}^{N} x_{ik}$$

$$\sigma_k^2 = \frac{1}{N-1}\sum_{i=i}^{N}(x_{ik}-\bar{x}_k)^2$$

$$\hat{x}_{ik} = \frac{x_{ik}-\bar{x}_k}{\sigma_k}$$

Softmax (non-linear)

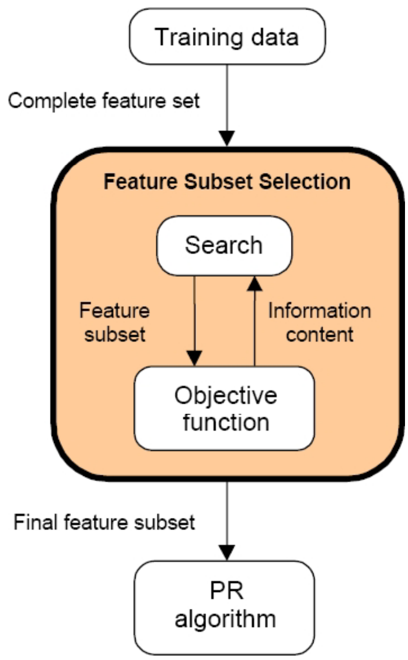
$$y = \frac{x_{ik}-\bar{x}_k}{r\sigma_k}$$

$$\hat{x}_{ik} = \frac{1}{1+\exp(-y)}$$

Remark: normalization may destroy important discrimination information

Feature selection

- How do we find the best subset of m out of n features.
- Search strategy
  - Exhaustive search implies $\binom{n}{m}$ if we fix *m* and 2^n if we need to search all possible *m* as well.
  - Choosing *10* out of *100* will result in 10^{13} queries to *J*
  - Obviously we need to guide the search!
- Objective function (J)
  - "Predict" classifier performance
  - Decides how good a subset if



Note that $\binom{m}{l} = \frac{m!}{l!(m-l)!}$

Distance measures (to specify J)

- Between two classes:
  - Distance between the closest two points?
  - Maximum distance between two points?
  - Distance between the class means?
  - Average distance between points in the two classes?
  - Which distance measure?
- Between K classes:
  - How do we generalize to more than two classes?
  - Average distance between the classes?
  - Smallest distance between a pair of classes?

Note: Often performance should be evalued in terms of classification error rate (e.g. on the training set or on a validation set)

Class separability measures

- How do we get an indication of the separability between two classes?
  - Euclidean distance $|\mu_r - \mu_s|$
  - Bhattacharyya distance
    - Can be defined for different distributions
    - For Gaussian data, it is

$$B = \frac{1}{8}(\mu_r - \mu_s)^T \left(\frac{\Sigma_r + \Sigma_s}{2}\right)^{-1} (\mu_r - \mu_s) + \frac{1}{2}\ln \frac{\left|\frac{1}{2}(\Sigma_r + \Sigma_s)\right|}{\sqrt{|\Sigma_r||\Sigma_s|}}$$

  - Mahalanobis distance between two classes:

$$\Delta = (\mu_1 - \mu_2)^T \Sigma^{-1} (\mu_1 - \mu_2)$$
$$\Sigma = N_1\Sigma_1 + N_2\Sigma_2$$

Divergence

- Divergence (see 5.5 in Theodoridis and Koutroumbas) is a measure of distance between probability density functions.
- Mahalanobis distance is a form of divergence measure.
- The Bhattacharrya distance is related to the Chernoff bound for the lowest classification error.
- If two classes have equal variance $\Sigma_1=\Sigma_2$, then the Bhattacharrya distance is proportional to the Mahalanobis distance.

Selecting individual features

- Each feature is treated individually (no correlation between features)
- Select a criteria, e.g. a distance measure
- Rank the feature according to the value of the criteria C(k)
- Select the set of features with the best individual criteria value
- Multiclass situations:
  - Average class separability or
  - C(k) = min distance(i,j) - worst case ← Often used

- Advantage with individual selection: computation time
- Disadvantage: no correlation is utilized.

Individual feature selection cont.

- We can also include a simple measure of feature correlation.
- Cross-Correlation between feature i and j: ($|\rho_{ij}| \leq 1$)

$$\rho_{ij} = \frac{\sum_{n=1}^{N} x_{ni} x_{nj}}{\sum_{n=1}^{N} x_{ni}^2 \sum_{n=1}^{N} x_{nj}^2}$$

- Simple algorithm:
  – Select C(k) and compute for all x_k, k=1,...m. Rank in descending order and select the one with best value. Call this x_{i1}.
  – Compute the cross-correlation between x_{i1} and all other features. Choose the feature x_{i2} for which

$$i_2 = \arg\max_j \left\{ \alpha_1 C(j) - \alpha_2 \left| \rho_{i1j} \right| \right\} \text{for all } j \neq i_1$$

  – Select x_{ik}, k=3,...l so that

$$i_k = \arg\max_j \left\{ \alpha_1 C(j) - \frac{\alpha_2}{k-1} \sum_{r=1}^{k-1} \left| \rho_{i1j} \right| \right\} \text{for all } j \neq i_1$$

Sequential backward selection

- Example: 4 features x_1,x_2,x_3,x_4
- Choose a criterion C and compute it for the vector $[x_1,x_2,x_3,x_4]^T$
- Eliminate one feature at a time by computing $[x_1,x_2,x_3]^T$, $[x_1,x_2,x_4]_T$, $[x_1,x_3,x_4]^T$ and $[x_2,x_3,x_4]^T$
- Select the best combination, say $[x_1,x_2,x_3]^T$.
- From the selected 3-dimensional feature vector eliminate one more feature, and evaluate the criterion for $[x_1,x_2]^T$, $[x_1,x_3]_T$, $[x_2,x_3]^T$ and select the one with the best value.
- Number of combinations searched:
  1+1/2((m+1)m-l(l+1))

Sequential forward selection

- Compute the criterion value for each feature. Select the feature with the best value, say x_1.
- Form all possible combinations of features x1 (the winner at the previous step) and a new feature, e.g. $[x_1,x_2]^T$, $[x_1,x_3]^T$, $[x_1,x_4]^T$, etc. Compute the criterion and select the best one, say $[x_1,x_3]^T$.
- Continue with adding a new feature.
- Number of combinations searched: lm-l(l-1)/2.
  - Backwards selection is faster if l is closer to m than to 1.

Plus-L Minus-R Selection (LRS)

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 additions

Algorithm

1. If $L > R$ then start with the empty set $Y = \emptyset$ else start with the full set $Y = X$ goto step 3
2. Repeat SFS step L times
3. Repeat SBS step R times
4. Goto step 2

LRS attempts to compensate for weaknesses in SFS and SBS by backtracking

Bidirectional Search (BDS)

- Bidirectional Search is a parallel implementation of SFS and SBS
  - SFS is performed from the empty set
  - SBS is performed from the full set
- To guarantee that SFS and SBS converge to the same solution, we must ensure that
  - Features already selected by SFS are not removed by SBS
  - Features already removed by SBS are not selected by SFS
  - For example, before SFS attempts to add a new feature, it checks if it has been removed by SBS and, if it has, attempts to add the second best feature, and so on. SBS operates in a similar fashion

Floating search methods

- Problem with backward selection: if one feature is excluded, it cannot be considered again.
- Floating methods can reconsider features previously discarded.
- Floating search can be defined both for forward and backward selection, here we study forward selection.
- Let $X_k=\{x_1,x_2,...,x_k\}$ be the best combination of the k features and Y_{m-k} the remaining m-k features.
- At the next step the k+1 best subset $X_{k+1}$is formed by 'borrowing' an element from Y_{m-k}.
- Then, return to previously selected lower dimension subset to check whether the inclusion of this new element improves the criterion.
- If so, let the new element replace one of the previously selected features.

Algorithm for floating search

- Step I: Inclusion

  x_{k+1}=argmax$_{y \in Ym-k}$C({X_k,y}) (choose the element from Y_{m-k} that has best effect of C when combined with X_k).

  Set X_{k+1}= {X_k, x_{k+1}}.

- Step II: Test

  1. x_r= argmax$_{y \in X_{k+1}}$C({X_{k+1}-y}) (Find the feature with the least effect on C when removed from X_{k+1})
  2. If r=k+1, change k=k+1 and go to step I.
  3. If r$\neq$k+1 AND C({X_{k+1}-x_r})<C(X_k), goto step I. (If removing x_k did not improve the cost, no further backwards selection)
  4. If k=2 put X_k= X_{k+1}- x_r and C(X_k)=C(X_{k+1}- x_r). Goto step I.

Algorithm cont.

- Step III: Exclusion

  1.X_k'=X_{k+1}-x_r (remove x_r)

  2.x_s= argmax$_{y \in X_k'}$C({X_k'-y}) (find the least significant feature in the new set.)

  3.If C(X_k'- x_s)<C(X_{k-1}) then X_k= X_k' and goto step I.

  4.Put X_{k-1}'=X_k'-x_s and k=k-1.

  5.If k=2, put X_k=X_k' and C(X_k)=C(X_k') and goto step I.

  6.Goto step III.

Floating search often yields better performance than sequential search, but at the cost of increased computational time.

Optimal searches and randomized methods

- If the criterion increases monotonically $J(x_{i1}) \leq J(x_{i1}, x_{i2}) \leq J(x_{i1}, x_{i2}, \ldots, x_{in})$, one can use graph-theoretic methods to perform effective subset searches. (I.e. branch and bound or dynamic programming)
- Randomized methods are also popular, examples would be sequential searching with random starting subsets, simulated annealing (a random subset permutation where the randomness cools off) or genetic algorithms.

Sequential Floating Search (SFFS and SFBS)

- Extension to the LRS algorithms with flexible backtracking capabilities
- Rather than fixing the values of L and R, these floating methods allow those values to be determined from the data: The size of the subset during the search can be thought to be " ‘floating" ’
- Sequential Floating Forward Selection (SFFS) starts from the empty set
  - After each forward step, SFFS performs backward steps as long as the objective function increases
- Sequential Floating Backward Selection (SFBS) starts from the full set
  - After each backward step, SFBS performs forward steps as long as the objective function increases

Feature transforms

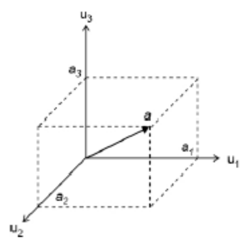
- We now consider computing new features as linear combinations of the existing features.
- From the original feature vector x, we compute a new vector y of transformed features

  $y=A^Tx$

  y is l-dimensional, x is m-dimensional, A is a l×m matrix.
- y is normally defined in such a way that it has lower dimension than x.

Vector spaces

- A set of vectors $u_1, u_2, \ldots u_n$ is said to form a basis for a vector space if any arbitrary vector x can be represented by a linear combination $x = a_1 u_1 + a_2 u_2 + \ldots a_n u_n$
  - The coefficients $a_1, a_2, \ldots a_n$ are called the components of vector x with respect to the basis u_i
  - In order to form a basis, it is necessary and sufficient that the u_i vectors be linearly independent
  - A basis u_i is said to be orthogonal if $u_i^T u_j = \begin{cases} \neq 0 & i = j \\ = 0 & i \neq j \end{cases}$
  - A basis u_i is said to be orthonormal if $u_i^T u_j = \begin{cases} = 1 & i = j \\ = 0 & i \neq j \end{cases}$



Linear transformation

- A linear transformation is a mapping from a vector space X^N onto a vector space Y^M, and is represented by a matrix
  - Given a vector $x \in X^N$, the corresponding vector y on Y^M is

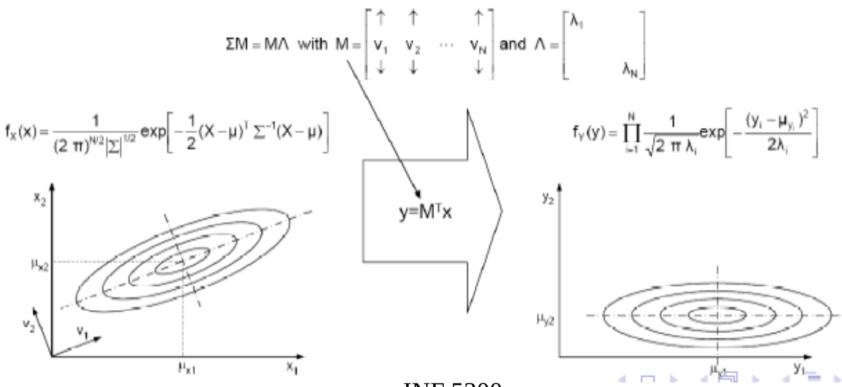
$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} a_{11} & a_{11} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m1} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix}$$

Eigenvalues and eigenvectors

- Given a matrix $A_{N \times N}$, we say that v is an eigenvector if there exists a scalar λ (the eigenvalue) such that $Av = \lambda v \Leftrightarrow v$ is an eigenvector with corresponding eigenvalue λ
- $Av = \lambda v \Rightarrow (A - \lambda I)v = 0 \Rightarrow$
  $|(A - \lambda I)| = 0 \Rightarrow \underbrace{\lambda^N + a_1 \lambda^{N-1} + \ldots a_{N-1}\lambda + a_0 = 0}_{\text{Characteristic equation}}$
- Zeroes of the characteristic equation are the eigenvalues of A
- A is non-singular $\Leftrightarrow$ all eigenvalues are non-zero
- A is real and symmetric $\Leftrightarrow$ all eigenvalues are real, and eigenvectors are orthogonal

Interpretation of eigenvectors and eigenvalues

- The eigenvectors of the covariance matrix Σ correspond to the *principal axes* of equiprobability ellipses!
- The linear transformation defined by the eigenvectors of Σ leads to vectors that are uncorrelated *regardless* of the form of the distribution
  - If the distribution happens to be Gaussian, then the transformed vectors will be statistically independent



Linear feature transforms

- Feature extraction can be stated as
  - Given a feature space $x_i \in \mathbb{R}_n$ find an optimal mapping $y = f(x) : \mathbb{R}_n \rightarrow \mathbb{R}_m$ with $m < n$.
  - An optimal mapping in classification :the transformed feature vector y yield the same classification rate as x.
- The optimal mapping may be a non-linear function
  - Difficult to generate/optimize non-linear transforms
  - Feature extraction is therefore usually limited to linear transforms $y = A^T x$

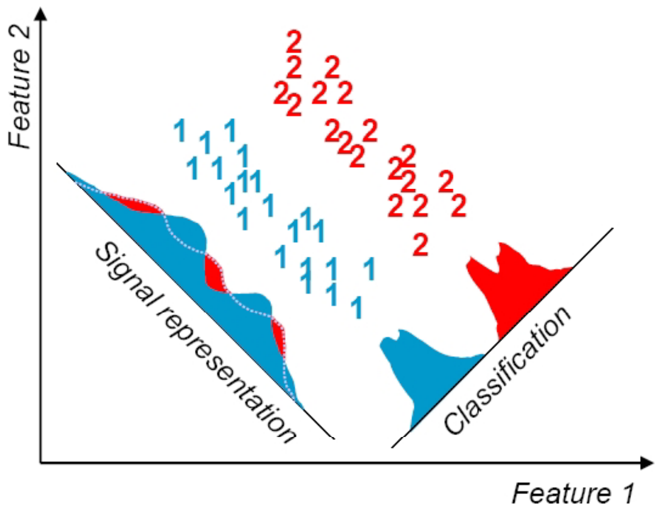
$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} a_{11} & a_{11} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m1} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix}$$

Signal representation vs classification

- The search for the feature extraction mapping $y = f(x)$ is guided by an objective function we want to maximize.
- In general we have two categories of objectives in feature extraction:
  - Signal representation: Accurately approximate the samples in a lower-dimensional space by minimizing the mean square error between the original feature vector and the low-dimensional projection.
  - Classification: Keep (or enhance) class-discriminatory information in a lower-dimensional space.

Signal representation vs classification

- Principal components analysis (PCA)
  - - signal representation, unsupervised
  - Minimize the mean square representation error
- Linear discriminant analysis (LDA)
  - -classification, supervised
  - Maximize the distance between the classes



Correlation matrix vs. covariance matrix

- Σ_x is the covariance matrix of x

$$\Sigma_x = E\left[(x-\mu)(x-\mu)^T\right]$$

- R_x is the correlation matrix of x

$$R_x = E\left[(x)(x)^T\right]$$

- $R_x=\Sigma_x$ if $\mu_x=0$.

Principal component or Karhunen-Loeve transform

- Let x be a feature vector.
- Features are often correlated, which might lead to redundancies.
- We now derive a transform which yields uncorrelated features.
- We seek a linear transform $y=A^Tx$, and the y_is should be uncorrelated.
- The y_is are uncorrelated if $E[y(i)y(j)]=0$, $i \neq j$.
- If we can express the information in x using uncorrelated features, we might need **fewer** coefficients.

Principal component transform

- The correlation of Y is described by the correlation matrix

  $R_Y = E[yy^T] = E[A^T x x^T A] = A^T R_x A$ R_x is the correlation matrix of X

  R_x is symmetric, thus all eigenvectors are orthogonal.

- We seek uncorrelated components of Y, thus R_y should be diagonal.

From linear algebra:

- R_y will be diagonal if A is formed by the orthogonal eigenvectors a_i, i=0,...,N-1 of R_x: $R_y = A^T R_x A = \Lambda$, where Λ is diagonal with the eigenvalues of R_x, λ_i, on the diagonal.
- We find A by solving the equation $A^T R_x A = \Lambda$ (using Singular Value Decomposition (SVD)).
- A is formed by computing the eigenvectors of $R_{x.}$ Each eigenvector will be a column of A.

Mean square error approximation

- x can be expressed as a combination of all N basis vectors:

$$x = \sum_{i=0}^{N-1} y(i) a_i, \text{ where } y(i) = a_i^T x$$

- An approximation to x is found by using only m of the basis vectors:

$$\hat{x} = \sum_{i=0}^{m-1} y(i) a_i$$

a projection into the m-dimensional subspace spanned by m eigenvectors

- The PC-transform is based on minimizing the mean square error associated with this approximation.
- The mean square error associated with this approximation is

$$E\left[\|x-\hat{x}\|^2\right] = E\left[\left\|\sum_{i=m}^{N-1} y(i) a_i\right\|^2\right] = E\left[\sum_i \sum_j \left(y(i) a_i^T\right)\left(y(j) a_j\right)\right] = \sum_{i=m}^{N-1} E\left[y^2(i)\right] = \sum_{i=m}^{N-1} a_i^T E\left[xx^T\right] a_i$$

- Furthermore, we can find that

$$E\left[\|x-\hat{x}\|^2\right] = \sum_{i=m}^{N-1} a_i^T \lambda_i a_i = \sum_{i=m}^{N-1} \lambda_i$$

- The mean square error is thus

$$E\left[\|x-\hat{x}\|^2\right] = \sum_{i=1}^{N-1} \lambda_i - \sum_{i=1}^{m} \lambda_i = \sum_{i=m}^{N-1} \lambda_i$$

- The error is minimized if we select the eigenvectors corresponding to the *m* largest eigenvales of the correlation matrix R_x.
- The transformed vector y is called the principal components of x. The transform is called the principal component transform or Karhunen-Loeve-transform.

Principal component of the covariance matrix

- Alternatively, we can find the principal components of the covariance matrix Σ_x.
- If we have software for computing principal components of R_x, we can compute principal components from Σ_x by first setting $z=x-\mu_x$ and compute PC(z).

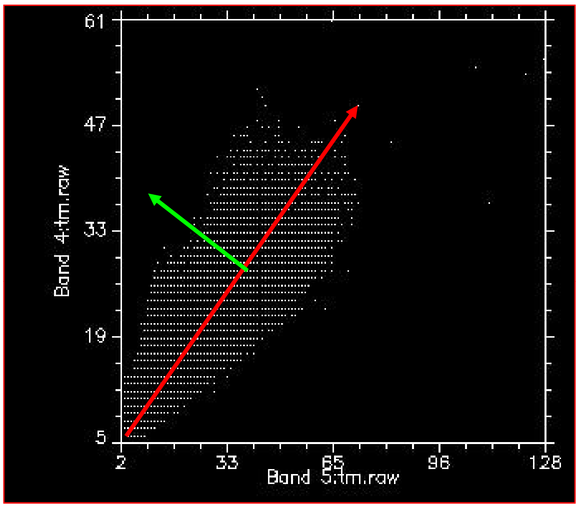
- The principal component transform is not scale invariant, because the eigenvectors are not invariant. Often, normalization to data with zero mean and unit variance is done prior to applying the PC-transform.

Principal components and total variance

- Assume that E[x]=0.
- Let y=PC(x).
- From R_y we know that the variance of component y_j is λ_j.
- The eigenvalues λ_j of the correlation matrix R_x is thus equal to the variance of the transformed features.
- By selecting the *m* eigenvectors with the largest eigenvalues, we select the *m* dimensions with the largest variance.
- The first principal component will be along the direction of the input space which has largest variance.

Geometrical interpretation of principal components

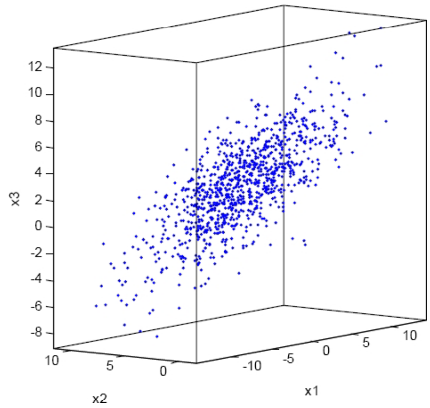
- The eigenvector corresponding to the largest eigenvalue is the direction in n-dimensional space with highest variance.
- The next principal component is orthogonal to the first, and along the direction with the second largest variance.



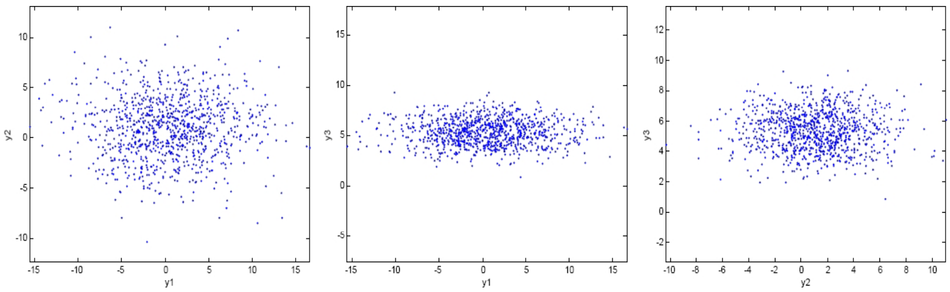
Note that the direction with the highest variance is NOT related to separability between classes.

PCA example

3d Gaussian with parameters



$$\mu = [0\ 5\ 2]^T,\ \Sigma = \begin{bmatrix} 25 & -1 & 7 \\ -1 & 4 & -4 \\ 7 & -4 & 10 \end{bmatrix}$$



Principal component images

- For an image with n bands, we can compute the principal component transform of the entire image X.
- Y=PC(X) will then be a new image with n bands, but most of the variance is in the bands with the lowest index (corresponding to the largest eigenvalues).

PC and compression

- PC-transform is optimal transform with respect to preserving the energy in the original image.
- For compression purposes, PC-transform is theoretically optimal with respect to maximizing the entropy (from information theory). Entropy is related to randomness and thus to variance.
- The basis vectors are the eigenvectors and vary from image to image. For transmission, both the transform coefficients and the eigenvectors must be transmitted.
- PC-transform can be reasonably well approximated by the Cosinus-transform or Sinus-transform. These use constant basis vectors and are better suited for transmission, since only the coefficients must be transmitted (or stored).