IN 5520
28.10.20
Pratical guidelines for classification
Evaluation
Feature selection
Principal component transform

Anne Solberg (anne@ifi.uio.no)

Literature

- Practical guidelines of classification – lecture foils
- Chap 5.6-5.7 Class separability and feature selection (see undervisningsmateriale/lecturenotes)
- Chap 6.1-6.2 PCA (see undervisningsmateriale/lecturenotes)
  Recommended reading about PCA, text from Chapter 17 Principal Component Analysis (see undervisningsmateriale/lecturenotes)

Approaching a classification problem

- Collect and label data
- Get to know the data: exploratory data analysis
- Choose features
- Consider preprocessing/normalization
- Choose classifier
  - Estimate classifier parameters on training data

Is the accuracy in the range I want?
NOT?
Analyze errors to understand why not
Improve algorithm, repeat

Approaching a classification problem

- Collect and label data
- Get to know the data: exploratory data analysis
- Choose features
- Consider preprocessing/normalization
- Choose classifier
  - Estimate classifier parameters on training data
- Estimate hyperparameters on validation data
  - Alternative: cross-validation on the training data set
- Compute the accuracy on test data

Model fit and bias,variance

- f(x) unknown function
- Training samples $x_1,\ldots.x_n$
- Measurements $y_i = f(x_i) + \varepsilon_i$ (ε_i :noise with zero mean and variance σ_ε^2)
- Estimate $\widehat{f(x)}$
- Want to minimize the error $E[(y-\widehat{f(x)})^2]$
- Bias : Difference between f(x) and $\widehat{f(x)}$ - how well does the model fit
- Variance of the estimate:$E[(\hat{f}(x) - E[\hat{f}(x)])^2]$

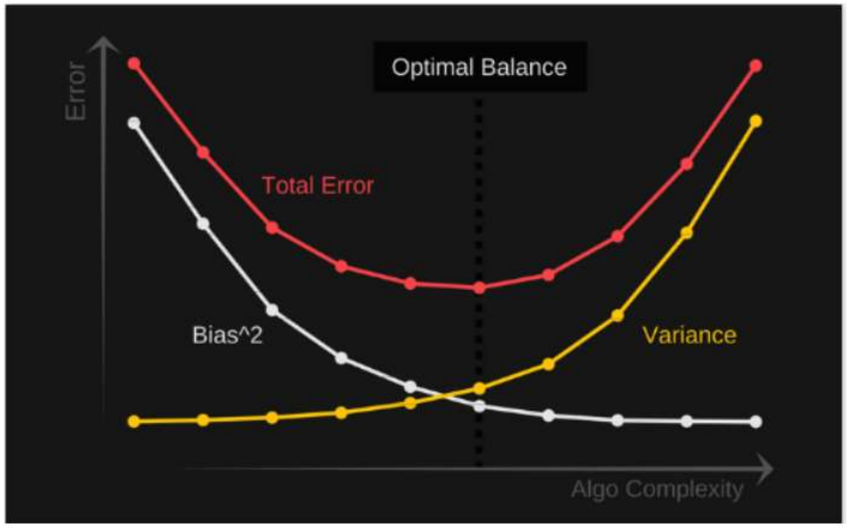
Balancing the total error

- Total squared error $Err(X) = E\left[\left(X - \widehat{f(x)}\right)^2\right]$

  The Err(x) can be further decomposed as

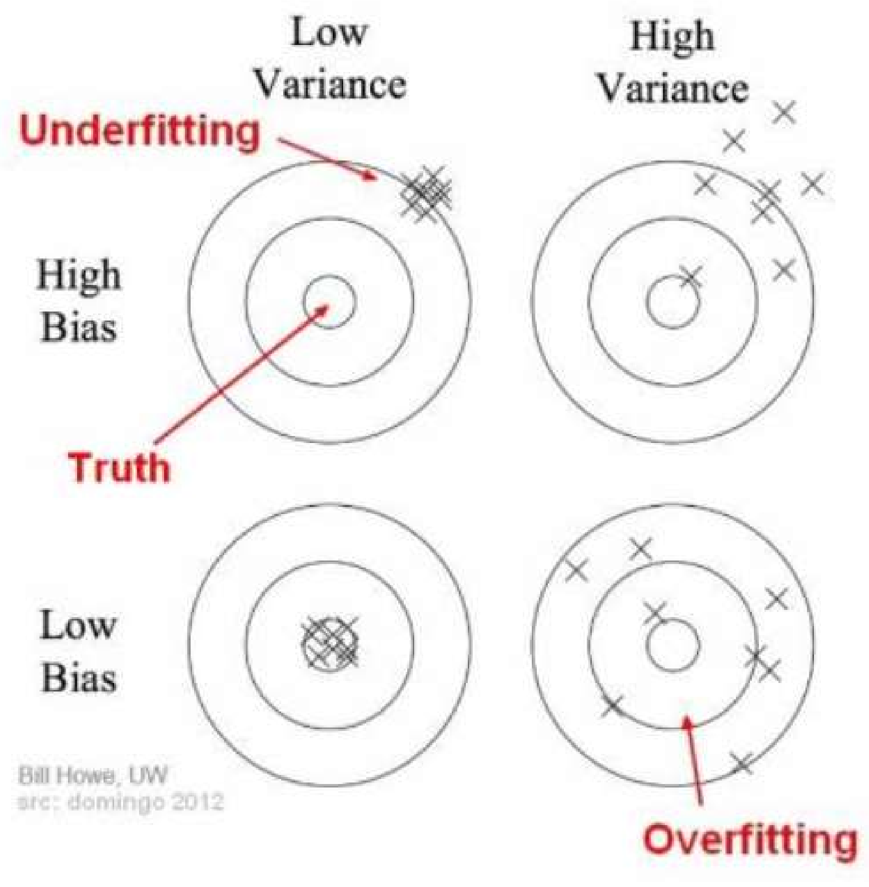
$$Err(x) = \left(E[\hat{f}(x)] - f(x)\right)^2 + E\left[\left(\hat{f}(x) - E[\hat{f}(x)]\right)^2\right] + \sigma_e^2$$

$$Err(x) = \mathrm{Bias}^2 + \mathrm{Variance} + \mathrm{Irreducible\ Error}$$



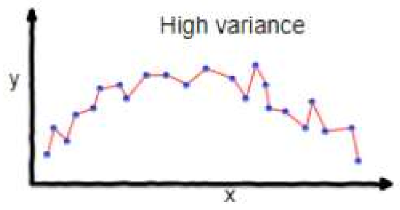
Balance the complexity of the model by the training error and validation error

Bias-Variance tradeoff

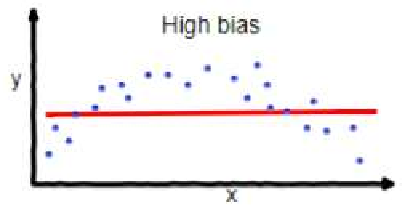


Underfitting and overfitting

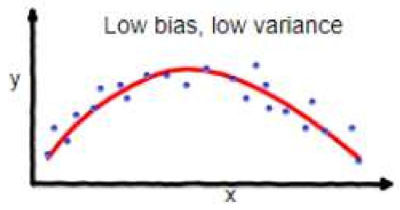
- Underfitting: the model is not able to capture the underlying pattern of data
  - High bias, low variance
  - In our setting: high error rate on training data and validation data
- Overfitting: the model captures the noise with the underlying pattern of data
  - Low bias, high variance
  - In our setting: low error rate on training data, high error on validation data



overfitting



underfitting



Good balance

Measures of classification accuary

- Average error rate
- Confusion matrices
- True/false positive/negatives
- Precision/recall and sensitivity/specificity
- ROC-curves

Confusion matrices

- A matrix with the true class label versus the estimated class labels for each class

Estimated class labels

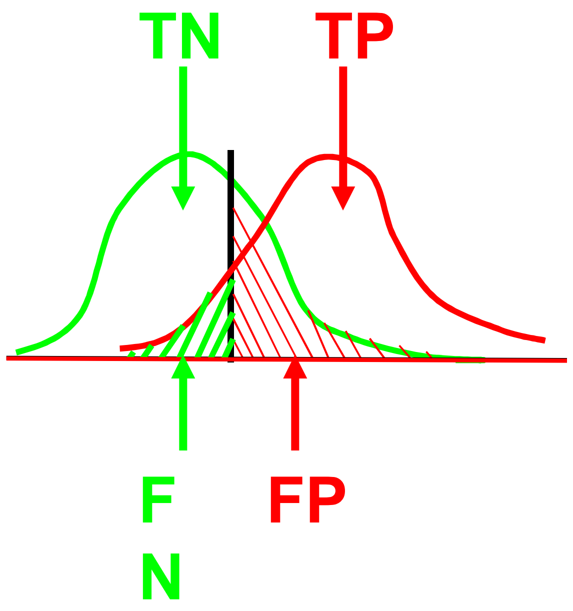
True class labels

	Class 1	Class 2	Class 3	Total #sampl es
Class 1	80	15	5	100
Class 2	5	140	5	150
Class 3	25	50	125	200
Total	110	205	135	450

True / False positives / negatives

- **True positive (TP):**
  Patient has cancer
  and test result is positive.
- True negative (TN):
  A healthy patient
  and a negative test result.
- **False positive (FP):**
  Healthy patient that gets a positive test result.
- **False negative (FN):**
  Cancer patient that gets a negative test result.

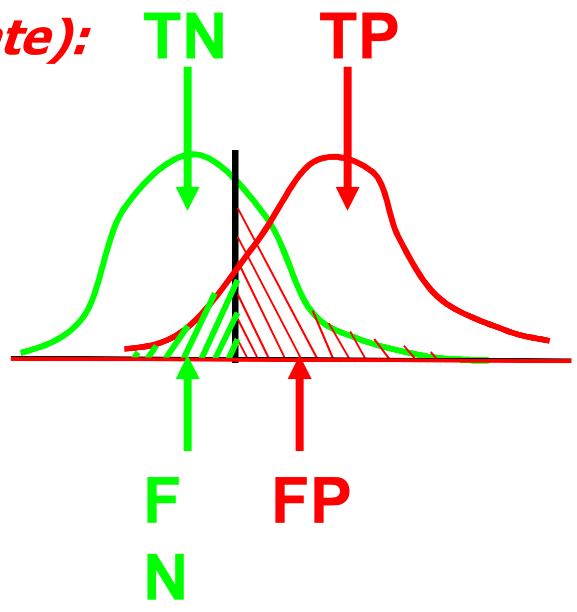
E.g., testing for cancer
No cancer | Cancer



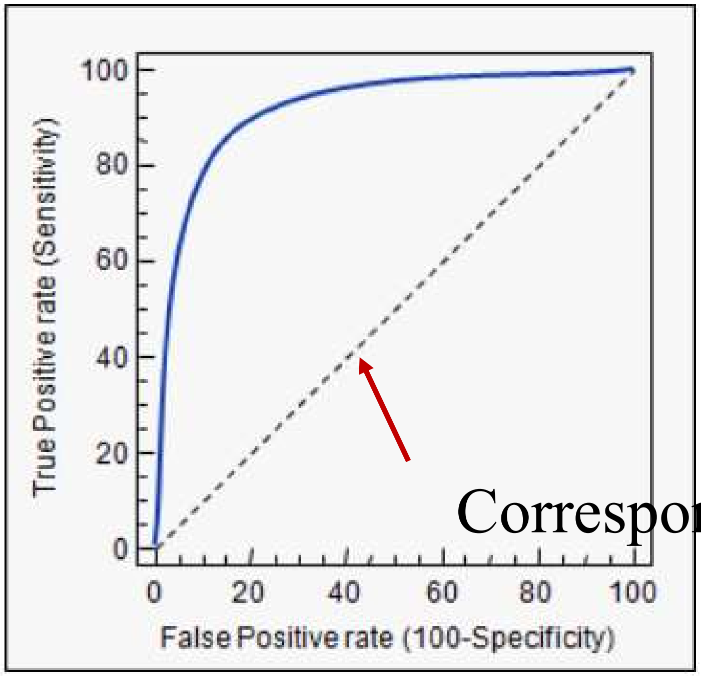
- *Good to have: **TP** & **TN***
- *Bad to have: **FP** (but this will probably be detected)*
- *Worst to have: **FN** **(may go un-detected)***

Sensitivity and specificity, F1-score

- ***Precision:***
  - **Precision = TP/(TP+FP)**
  - The probability that patient is really sick given that he is classified sick.
- ***Sensitivity (also called recall/true positive rate):***
  Sensitivity/TPR = TP/(TP+FN)
  - The probability that the test is positive given that the patient is sick.
  - Higher sensitivity means that fewer desease cases go undetected.
- ***Specificity: (also called true negative rate)***
  Specificity = TN/(TN+FP)
  - The probability that a test is negative given that the patient is not sick.
  - Higher specificity means that fewer healthy patients are labeled as sick.
- **Combined score**
  - F1= 2(precision*recall)/(precision+recall)



Receiver Operating Characteristic(ROC)-curve



Used for binary classification problems to set the detection threshold.

Good performance: the curve should be close to the upper left corner

Corresponds to random guess

Outliers and doubt

- In a classification problem, we might want to identify outliers and doubt samples
- We might want an ideal classifier to report
  - ’this sample is from class l’ (usual case)
  - ’this sample is not from any of the classes’ (outlier)
  - ’this sample is too hard for me’ (doubt/reject)
- The two last cases should lead to a rejection of the sample!

Outliers

- Heuristically defined as ”… samples which did not come from the assumed population of samples”
- The outliers can result from some breakdown in preprocessing.
- Outliers can also come from pixels from other classes than the classes in the training data set.
  - Example: K tree species classes, but a few road pixels divide the forest regions.
- One way to deal with outliers is to model them as a separate class, e.g., a gaussian with very large variance, and estimate prior probability from the training data
- Another approach is to decide on some threshold on the aposteriori probability– and if a sample falls below this threshold for all classes, then declare it an outlier.
- Related to normalization: a max/min normalization will be sensitive to outliers

Doubt samples

- Doubt samples are samples for which the class with the highest probability is not significantly more probable than some of the other classes (e.g. two classes have essentially equal probability).
- Doubt pixels typically occurr on the border between two classes (”mixels”)
  - Close to the decision boundary the probabilities will be almost equal.
- Classification software can allow the user to specify thresholds for doubt.

The ”curse” of dimensionality

– In practice, the curse means that, for a given sample size, there is a maximum number of features one can add before the classifier starts to degrade.

- For a finite training sample size, the correct classification rate initially increases when adding new features, attains a maximum and then begins to decrease.
- For a high dimensionality, we will need lots of training data to get the best performance.
- => ≈10 samples / feature / class.

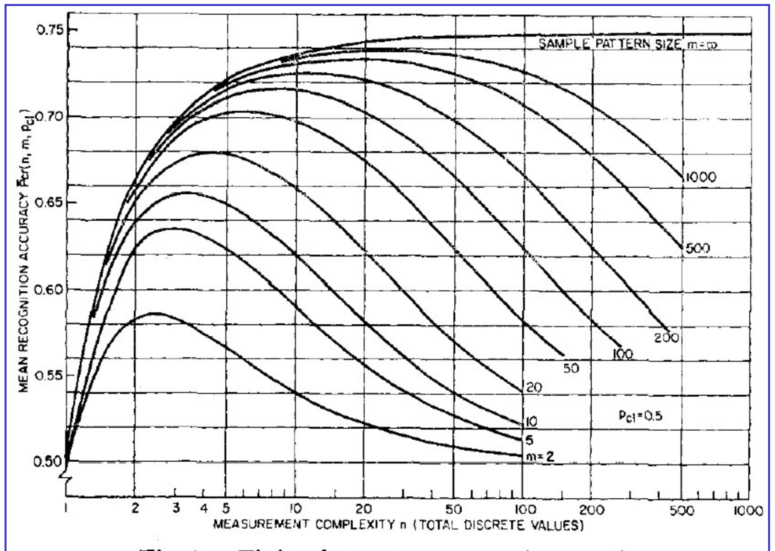


Fig. 3. Finite data set accuracy ($p_{c1} = \frac{1}{2}$).

Correct classification rate as function of feature dimensionality, for different amounts of training data. Equal prior probabilities of the two classes is assumed.

Use few, but good features

- To avoid the ”curse of dimensionality” we must take care in finding a set of relatively few features.
- A good feature has high within-class homogeneity, and should ideally have large between-class separation.
- In practise, one feature is not enough to separate all classes, but a good feature should:
  - separate some of the classes well
  - Isolate one class from the others.
- If two features look very similar (or have high correlation), they are often redundant and we should use only one of them.
- Class separation can be studied by:
  - Visual inspection of the feature image overlaid the training mask
  - Scatter plots
- Evaluating features as done by training can be difficult to do automatically, so manual interaction is normally required.

How do we beat the ”curse of dimensionality”?

- Generate few, but informative features
  - Careful feature design given the application
- Get as much labelled data as possible!
- Try a simple classifier first
  - Do the features work? Do we need additional features?
  - Iterate between feature extraction and classification
- Reducing the dimensionality
  - Feature selection – select a subset of the original features
  - Feature transforms – compute a new subset of features based on a linear combination of all features
    - Example: Principal component transform
      - Unsupervised, finds the combination that maximizes the variance in the data.
- When you are confident that the features are good, consider a more advanced classifier.

Linear feature transforms

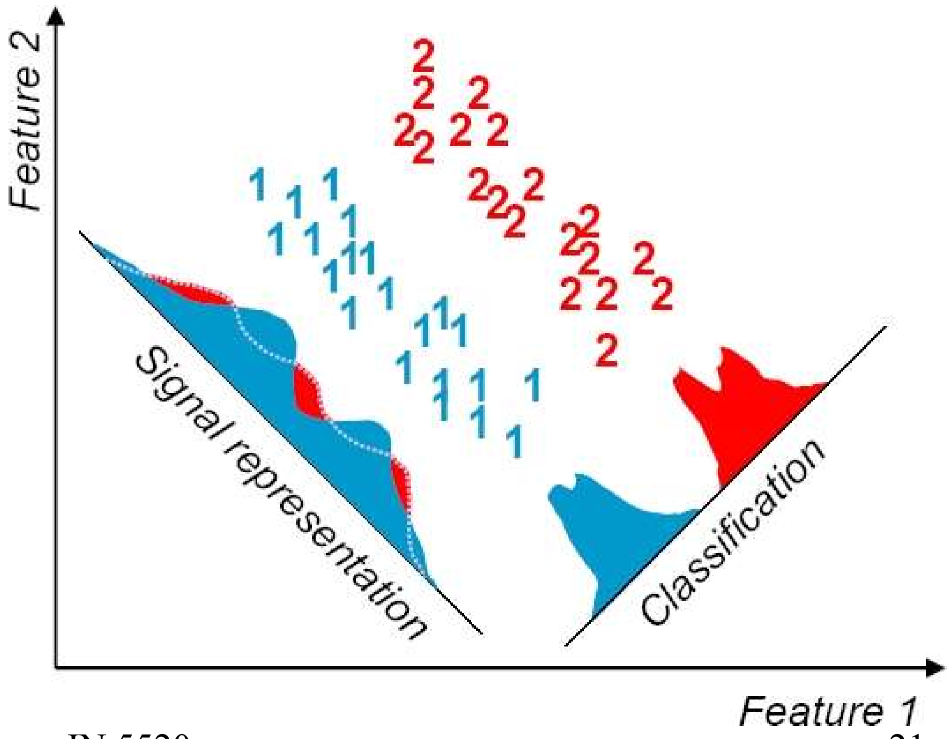
- Feature extraction can be stated as
  - Given a feature space $x_i \in \mathbb{R}_n$ find an optimal mapping $y = f(x) : \mathbb{R}_n \rightarrow \mathbb{R}_m$ with $m < n$.
  - An optimal mapping in classification :the transformed feature vector y yield the same classification rate as x.
- The optimal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

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

Signal representation vs classification

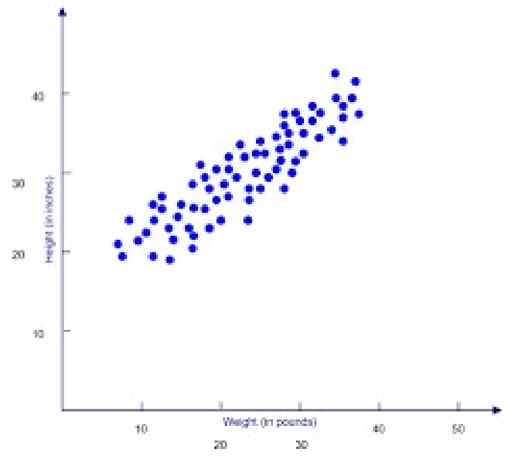
- Principal components analysis (PCA)
  - - signal representation, unsupervised
  - Minimize the mean square representation error (unsupervised)
- Linear discriminant analysis (LDA)
  - -classification, supervised
  - Maximize the distance between the classes (supervised)

    Nice theory, but rarely used as the number of selected features must be less than the number of classes
  - Not covered in this course



Idea behind (Principal Component Transform)

- Find a projection $\mathbf{y}=A^T\mathbf{x}$ of the feature vector $\mathbf{x}$
- Three interpretations of PCA:
  - Find the projection that maximize the variance along the selected projection
  - Minimize the reconstruction error (squared distance between original and transformed data)
  - Find a transform that gives uncorrelated features



Definitions: 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

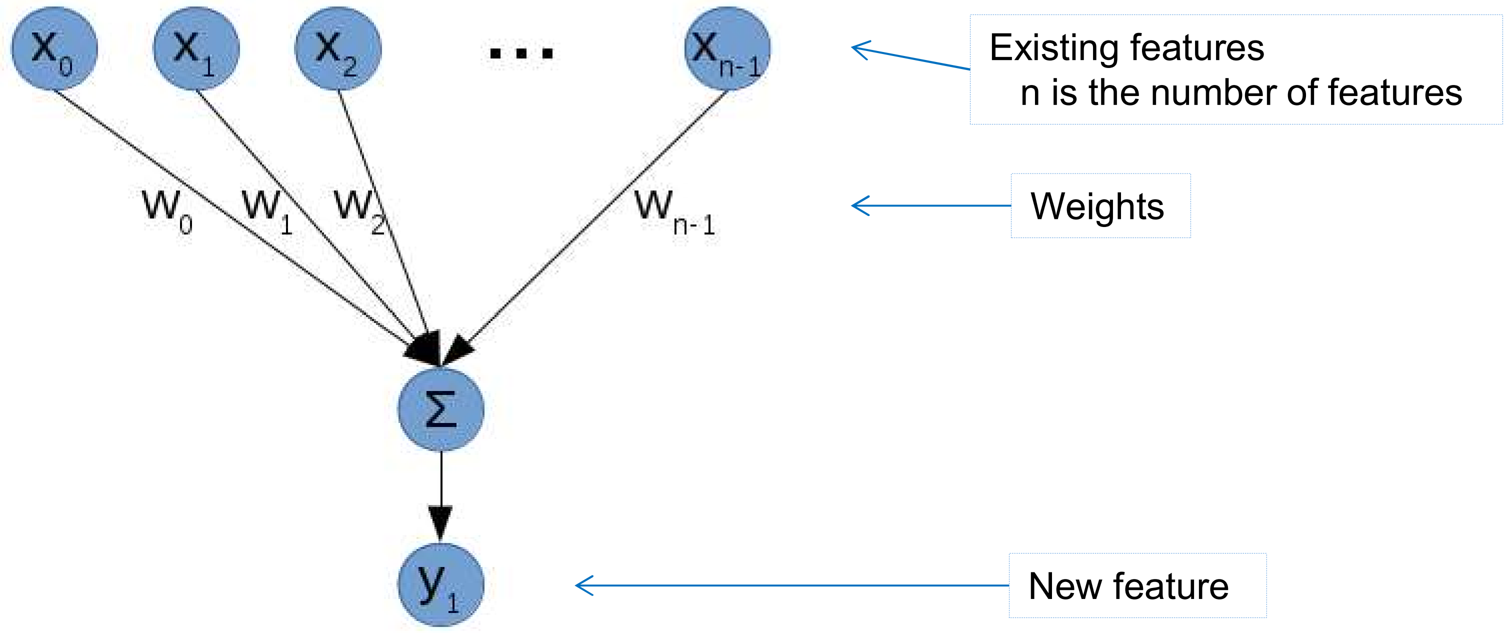
$$R_x = E\left[(x)(x)^T\right]$$

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

Principal component or Karhunen-Loeve transform - motivation

- 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 $E[y(i)y(j)^T]=0$, $i\neq j$.
- If we can express the information in x using uncorrelated features, we might need **fewer** coefficients.

Linear feature transforms I/II



From the original features to **one** new feature

Linear feature transforms II/II

- Multiple output features by applying **different weights for each one**:

$$y_1 = \sum_{i=0}^{n-1} w_{i1} x_i, \quad y_2 = \sum_{i=0}^{n-1} w_{i2} x_i, \quad \dots \quad y_m = \sum_{i=0}^{n-1} w_{im} x_i$$

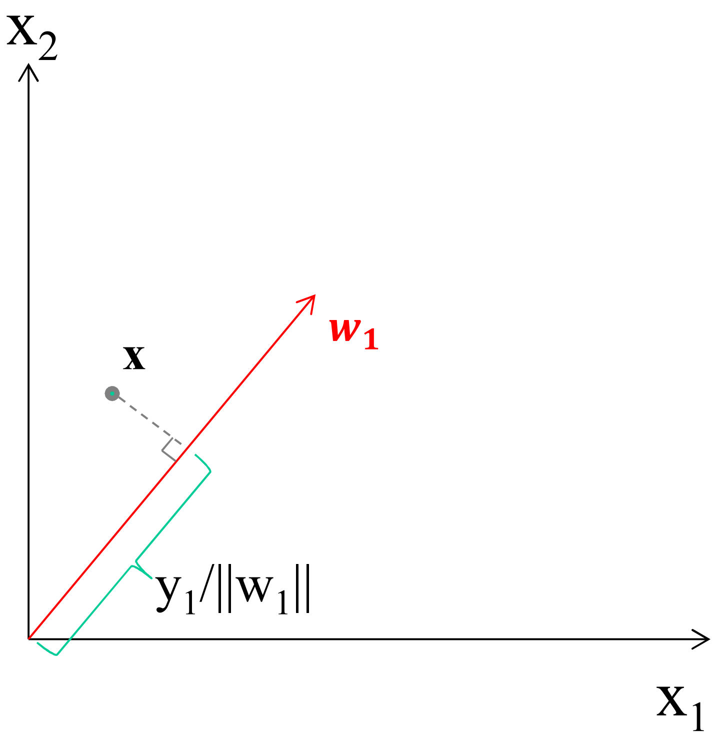
- In matrix notation $\mathbf{y} = \mathbf{A}^T\mathbf{x}$, $\mathbf{A}=[\mathbf{w}_1\ \mathbf{w}_2 \dots \mathbf{w}_m]$

- If **y** has fewer elements than **x**, we get a feature reduction

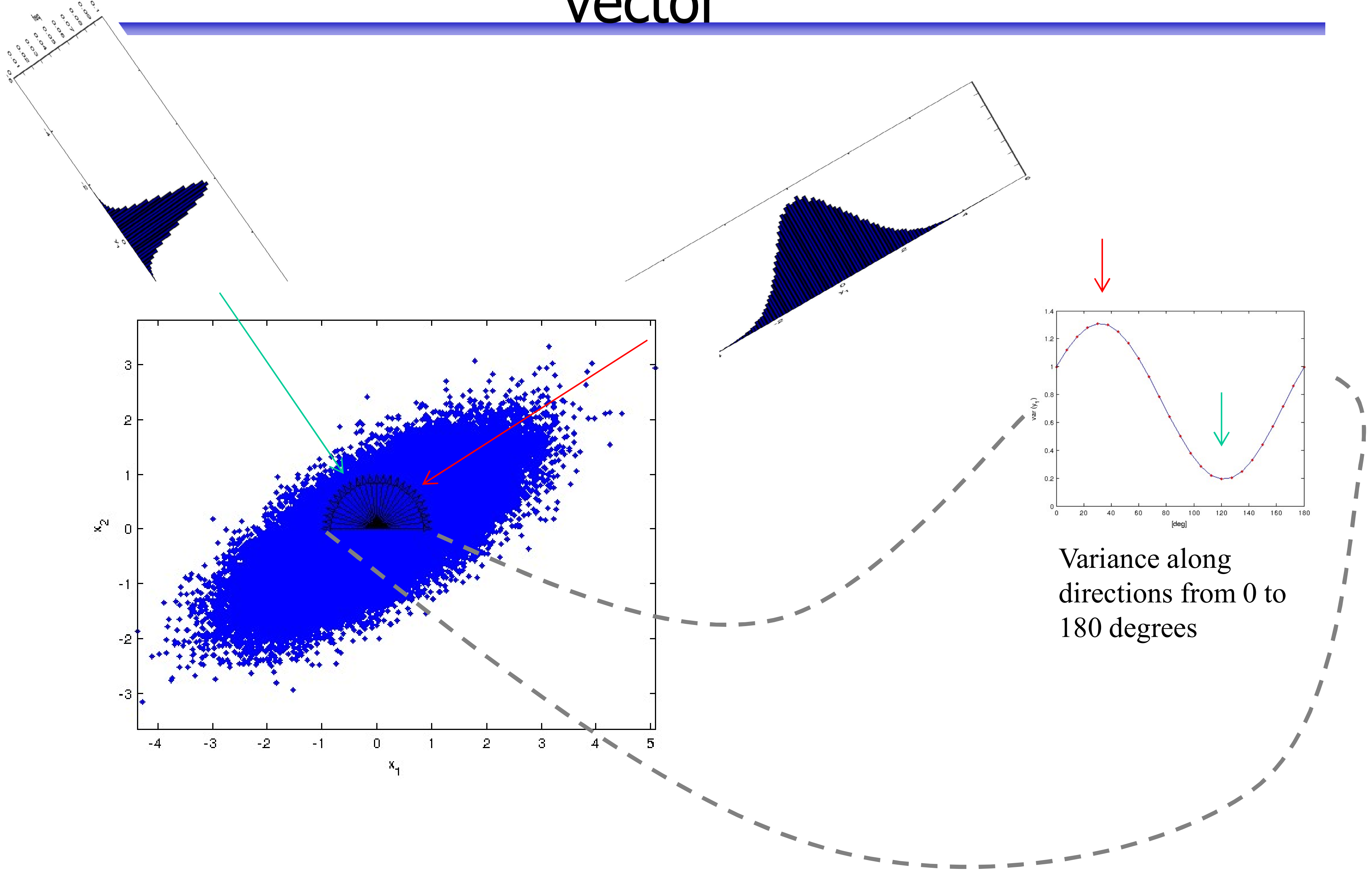
The weights | Visualization and intuition

$$y_1 = \sum_{i=0}^{n-1} w_{i1} x_i = \boldsymbol{w_1}^T \boldsymbol{x}$$

A linear transform is a shift of basis vectors
by projecting the points on the the basis vectors



Variance of y_1 as a function of direction of basis vector



Variance along directions from 0 to 180 degrees

Variance of y_1 cont.

- Assume mean of **x** is subtracted, so **x** has zero mean.
- The mean of the projections will also be zero:

$$\frac{1}{n}\sum_{i=1}^{n}(\vec{x}_i \cdot \vec{w})\vec{w} = \left(\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right) \cdot \vec{w}\right)\vec{w}$$

$$\begin{aligned}\sigma^2_{y_1} &= \frac{1}{N}\sum_i y_i^2 \\ &= \frac{1}{N}\sum_i (\mathbf{w}^T\mathbf{x}_i)^2 = \frac{1}{N}\sum_i \mathbf{w}^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{w} = \mathbf{w}^T\underbrace{\left(\frac{1}{N}\sum_i \mathbf{x}_i\mathbf{x}_i^T\right)}\mathbf{w} \\ &= \mathbf{w}^T\mathbf{R}\mathbf{w}\end{aligned}$$

Called σ^2_w on some slides

The sample covariance matrix / scatter matrix; **R**

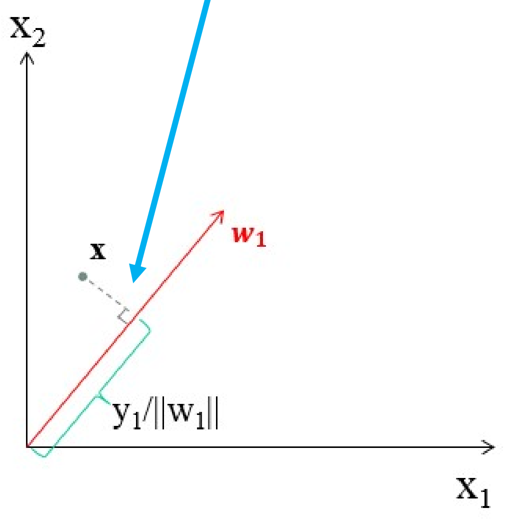
Variance and projection residuals

Single sample

Projection onto **w,** assuming |**w**|=1

$$
\begin{aligned}
||\vec{x}_i - \underbrace{(\vec{w}\cdot\vec{x}_i)}_{«y_i»}\vec{w}||^2 &= (\vec{x}_i - (\vec{w}\cdot\vec{x}_i)\vec{w})\cdot(\vec{x}_i - (\vec{w}\cdot\vec{x}_i)\vec{w}) \\
&= \vec{x}_i\cdot\vec{x}_i - \vec{x}_i\cdot(\vec{w}\cdot\vec{x}_i)\vec{w} \\
&\quad -(\vec{w}\cdot\vec{x}_i)\vec{w}\cdot\vec{x}_i + (\vec{w}\cdot\vec{x}_i)\vec{w}\cdot(\vec{w}\cdot\vec{x}_i)\vec{w} \\
&= ||\vec{x}_i||^2 - 2(\vec{w}\cdot\vec{x}_i)^2 + (\vec{w}\cdot\vec{x}_i)^2\vec{w}\cdot\vec{w} \\
&= \vec{x}_i\cdot\vec{x}_i - \underbrace{(\vec{w}\cdot\vec{x}_i)^2}_{«y_i^2»}
\end{aligned}
$$

w·**w**=1



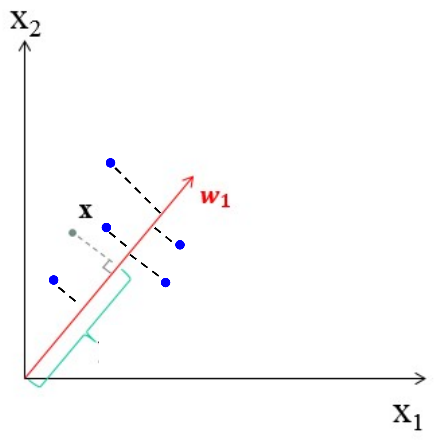
Residuals – sum over all samples

$$
\begin{aligned}
MSE(\vec{w}) &= \frac{1}{n}\sum_{i=1}^{n} \|\vec{x}_i\|^2 - (\vec{w}\cdot\vec{x}_i)^2 \\
&= \frac{1}{n}\left(\sum_{i=1}^{n} \|\vec{x}_i\|^2 - \boxed{\sum_{i=1}^{n} (\vec{w}\cdot\vec{x}_i)^2}\right)
\end{aligned}
$$

Sum all *n* samples (not dimensions)

Does not depend on w

Minimizing MSE is equivalent to maximizing this term



Maximizing the variance

- The mean of a square is always equal to the square of the mean plus the variance:

$$\boxed{\frac{1}{n}\sum_{i=1}^{n}(\vec{w}\cdot\vec{x}_i)^2} = \left(\frac{1}{n}\sum_{i=1}^{n}\vec{x}_i\cdot\vec{w}\right)^2 + \mathrm{Var}\left[\vec{w}\cdot\vec{x}_i\right]$$

The mean of $(\vec{w}\cdot\vec{x}_i)^2$

- But the projected data has zero mean, so this is equivalent to maximizing the variance.

Minimizing the squared errror is equivalent to maximizing the variance in the projected component

Criterion function

- Goal: Find transform minimizing representation error
- We start with a single weight-vector, **w**, giving us a single feature, y_1
- Let $J(\mathbf{w}) = \mathbf{w}^T\mathbf{R}\mathbf{w} = \sigma_w^2$
- Now, let's find $\max\limits_{\boldsymbol{w}} J(\boldsymbol{w})$
  $s.t. ||\boldsymbol{w}|| = 1$

As we learned on the previous slide, maximizing this is equivalent to minimizing representation error

Transform this problem into a unconstrained problem with a Lagrange multiplier (we skip details here)

Maximizing variance of y_1

$$\mathscr{L}(\mathbf{w},\lambda) \equiv \sigma_{\mathbf{w}}^2 - \lambda(\mathbf{w}^T\mathbf{w} - 1)$$

$$\frac{\partial L}{\partial \lambda} = -\mathbf{w}^T\mathbf{w} + 1$$

$$\frac{\partial L}{\partial \mathbf{w}} = 2\mathrm{R}\mathbf{w} - 2\lambda\mathbf{w}$$

Lagrangian function for maximizing σ^2_w with the constraint $\mathbf{w}^T\mathbf{w}$=1

$\Downarrow$ Equating zero

$$\mathbf{w}^T\mathbf{w} = 1$$

$$\mathrm{R}\mathbf{w} = \lambda\mathbf{w}$$

Unfamiliar with Lagrangian multipliers? See http://biostat.mc.vanderbilt.edu/wiki/pub/Main/CourseBios362/LagrangeMultipliers-Bishop-PatternRecognitionMachineLearning.pdf

The maximizing **w** is an eigenvector of R!

And σ^2_w=λ!

$\mathbf{w}_2$, $\mathbf{w}_3$, .. I/III

- $\mathbf{w}_1$ should be the eigenvector of $\mathbf{R_x}$ corresponding to the largest eigenvalue
- Ok, I've got the $\mathbf{w}_1$ giving me the transform (linear weights) that maximizes the variance / minimizes the representation error ..
- .. Now I want another one that again maximizes the variance / minimizes the representation error, but the new feature should be uncorrelated with my previous one ..
- .. Which $\mathbf{w}_2$ would give me this?

Eigendecomposition of covariance matrices

Real-valued, symmetric, «n-dimensional» covariance matrix

$$\mathbf{R} = \lambda_1 \mathbf{a}_1 \mathbf{a}_1' + \lambda_2 \mathbf{a}_2 \mathbf{a}_2' + \ldots + \lambda_n \mathbf{a}_n \mathbf{a}_n'$$

Eigenvalue (let's say largest)

Eigenvector corresponding to λ_1

Smallest eigenvalue

$\mathbf{a}^T_i \mathbf{a}_j = 0$ for $i \neq j$

Remember:
λ_i=variance of $\mathbf{x}^T \mathbf{a}_i$

Remark: $a' = a^T$

$\mathbf{w}_2$, $\mathbf{w}_3$, .. II/III

- What does uncorrelated mean? Zero covariance.
- Covariance of y_1 and y_2:

$$\frac{1}{N}\sum_{i}^{N} y_1(i)' y_2(i) = \frac{1}{N}\sum_{i}^{N} \mathbf{w}_1' \mathbf{x}(i)\mathbf{x}(i)' \mathbf{w}_2 = \mathbf{w}_1' \mathbf{R} \mathbf{w}_2$$

- We already have that $\mathbf{w}_1 = \mathbf{a}_1$
- From last slide, requiring $\mathbf{w}_1'\mathbf{R}\mathbf{w}_2 = \mathbf{a}_1'\mathbf{R}\mathbf{w}_2 = 0$ means requiring $\mathbf{w}_2'\mathbf{a}_1 = 0$

$\mathbf{w}_2$, $\mathbf{w}_3$, .. III/III

- We want $\max_{\mathbf{w}} \mathbf{w}'\mathbf{R}\mathbf{w}$, s.t. $|\mathbf{w}|=1$ *and* $\mathbf{w}'\mathbf{a}_1=0$
- We can simply remove $\lambda_1\mathbf{a}_1\mathbf{a}_1$' from **R**, creating $\mathbf{R}_{\text{next}} = \mathbf{R}- \lambda_1\mathbf{a}_1\mathbf{a}_1$', and again find $\max_{\mathbf{w}} \mathbf{w}'\mathbf{R}_{\text{next}}\mathbf{w}$ s.t. $|\mathbf{w}|=1$
- Studying the decomposition of **R** (a few slides back), we see that the solution is the eigenvector corresponding to the second largest eigenvalue
- Similarly, the $\mathbf{w}_3$, $\mathbf{w}_4$ etc. are given by the following eigenvectors sorted according to their eigenvalues

$\mathbf{w}_2$, $\mathbf{w}_3$, .. III+/III

$\max_{\mathbf{w}}$ $\mathbf{w}'\mathbf{R}\mathbf{w}$, s.t. $|\mathbf{w}|=1$

$$\mathbf{R} = \lambda_1 \mathbf{a}_1 \mathbf{a}_1' + \lambda_2 \mathbf{a}_2 \mathbf{a}_2' + \ldots + \lambda_n \mathbf{a}_n \mathbf{a}_n' \quad \rightarrow \quad \mathbf{w}=\mathbf{a}_1$$

$$\mathbf{R} = \cancel{\lambda_1 \mathbf{a}_1 \mathbf{a}_1'} + \lambda_2 \mathbf{a}_2 \mathbf{a}_2' + \ldots + \lambda_n \mathbf{a}_n \mathbf{a}_n' \quad \rightarrow \quad \mathbf{w}=\mathbf{a}_2$$

$$\mathbf{R} = \cancel{\lambda_1 \mathbf{a}_1 \mathbf{a}_1'} + \cancel{\lambda_2 \mathbf{a}_2 \mathbf{a}_2'} + \ldots + \lambda_n \mathbf{a}_n \mathbf{a}_n' \quad \rightarrow \quad \mathbf{w}=\mathbf{a}_3$$

… etc.

Eigenvectors sorted by their corresponding eigenvalues

Principal component transform (PCA)

- Place the *m* «principle» eigenvectors (the ones with the largest eigenvalues) along the columns of A
  - They are given as the eigenvectors of the covariance matrix R

- Then the transform $\mathbf{y} = \mathbf{A}^T\mathbf{x}$ gives you the *m* first principle components

- The *m*-dimensional **y**
  - have uncorrelated elements
  - retains as much variance as possible
  - gives the best (in the mean-square sense) description of the original data (through the «image»/projection/reconstruction **Ay**)

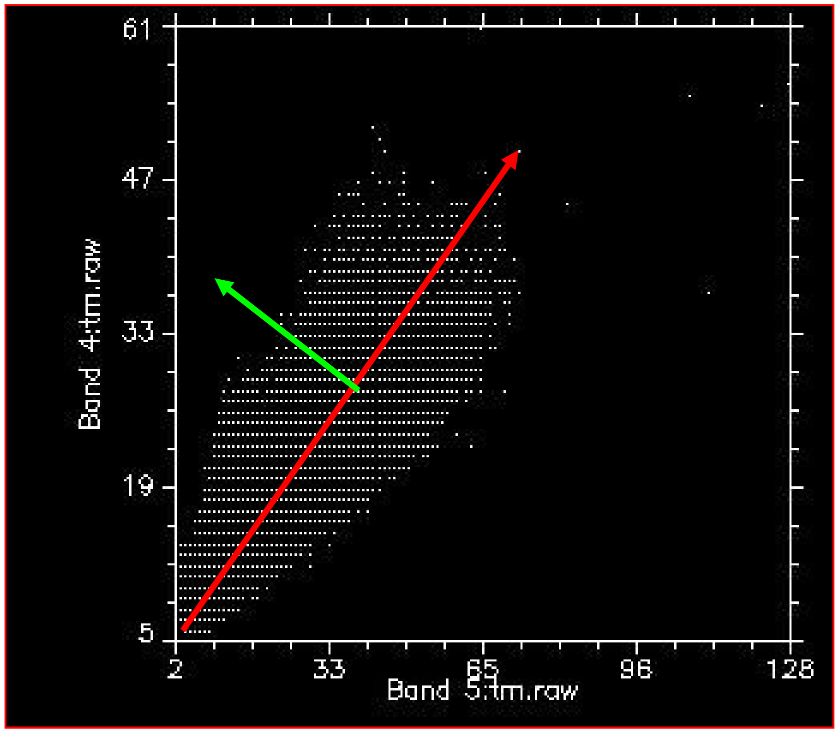
Note: The eigenvectors themselves can often give interesting information

PCA is also known as Karhunen-Loeve transform

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.

→

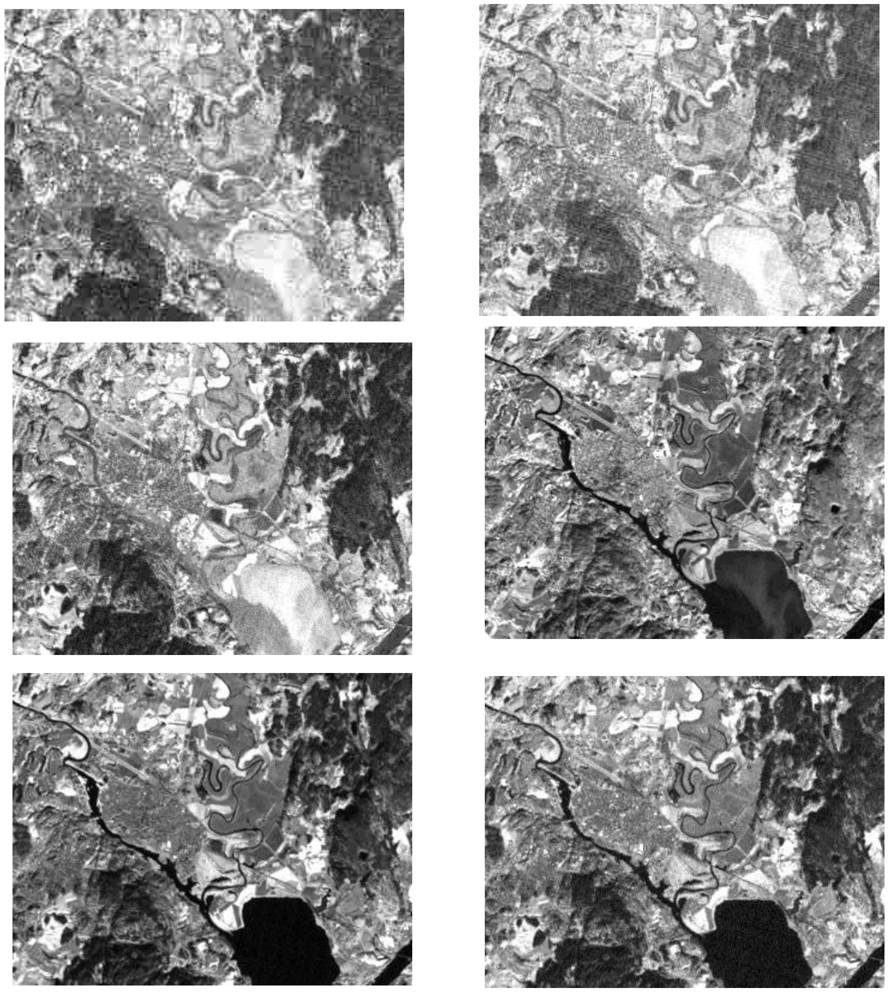


Note that the direction with the highest variance is NOT related to separability between classes.

PCA and multiband images

- We can compute the principal component transform for an image with *n* bands
- Let **X** be an *N*x*n* matrix having a row for each image sample
- Sample covariance matrix (after mean subtracted): $\boldsymbol{R} = \frac{\boldsymbol{1}}{\boldsymbol{N}}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}$
- Place the (sorted) eigenvectors along the columns of **A**
- **Y**=**XA** will then contain the image samples, however most of the variance is in the «bands» with the lowest index (corresponding to the largest eigenvalues), and the new features are uncorrelated

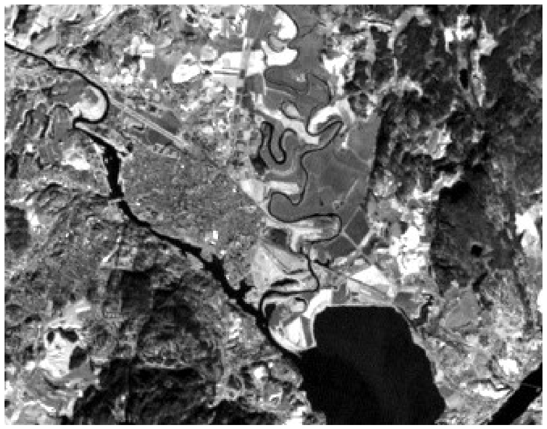
PCA example – original image

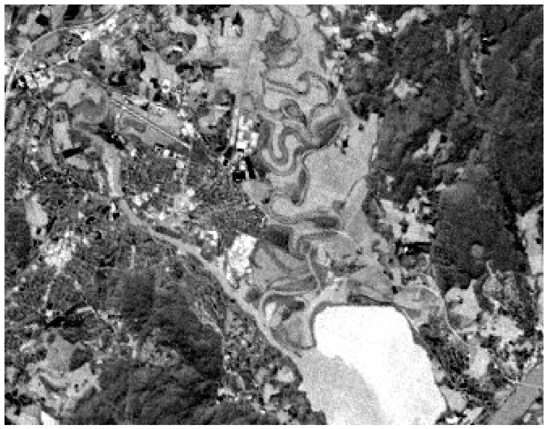


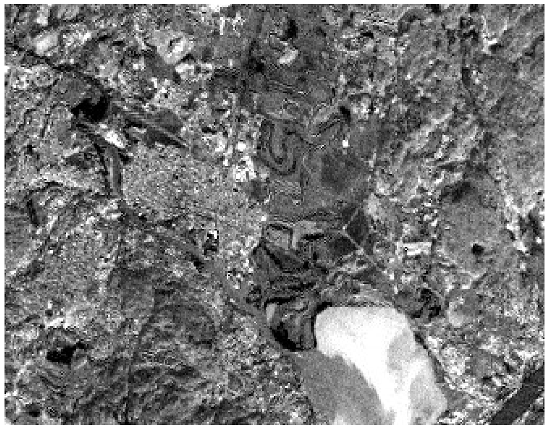
- Satellite image from Kjeller
- 6 spectral bands with different wavelengths

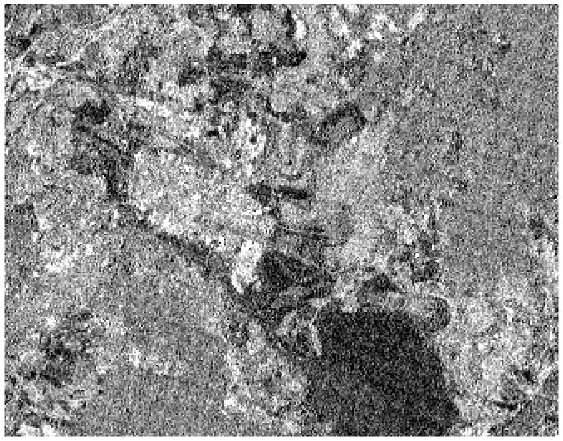
1	Blue	0.45-0.52	Max. penetration of water
2	Green	0.52-0.60	Vegetation and chlorophyll
3	Red	0.63-0.69	Vegetation type
4	Near-IR	0.76-0.90	Biomass
5	Mid-IR	1.55-1.75	Moisture/water content in vegetation/soil
7	Mid-IR	2.08-2.35	Minerals

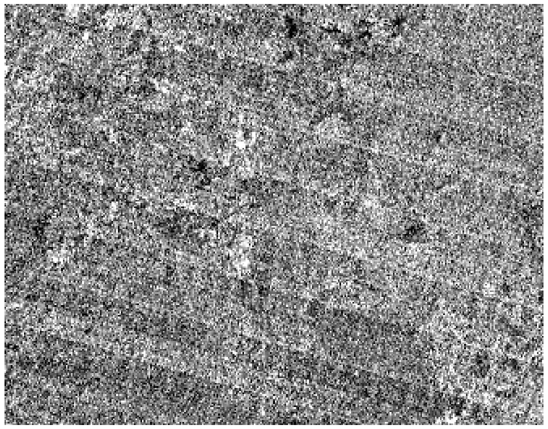
Example cont: Principal component images

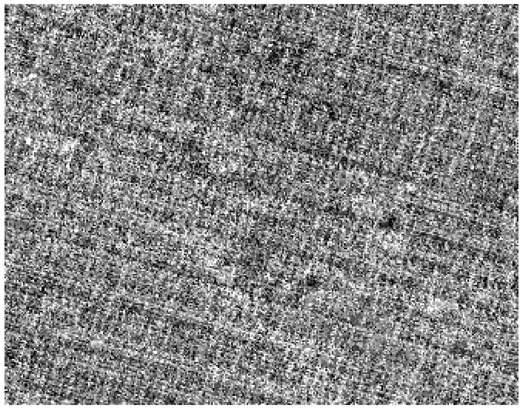

Principal component 1


Principal component 2


Principal component 3


Principal component 4

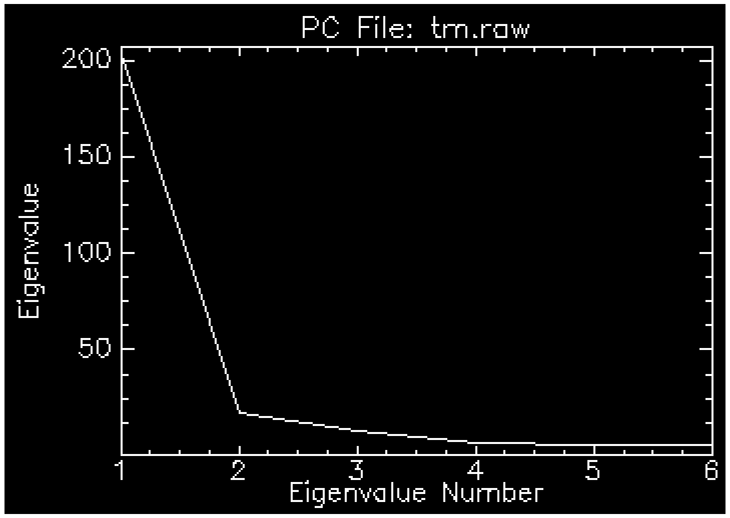

Principal component 5


Principal component 6

Example cont: Inspecting the eigenvalues

The mean-square representation error we get with m of the N PCA-components is given as

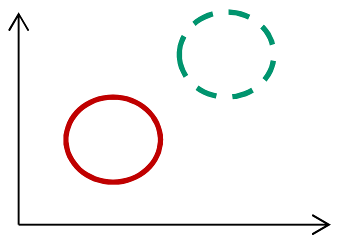
$$E\left[\|x-\hat{x}\|^2\right] = \sum_{i=1}^{N-1}\lambda_i - \sum_{i=1}^{m}\lambda_i = \sum_{i=m}^{N-1}\lambda_i$$



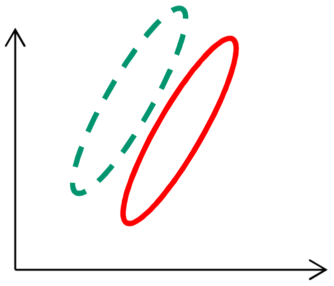
Plotting λs will give indications on how many features are needed for representation (explaining the variance)

PCA and classification

- Reduce overfitting by detecting directions/components without any/very little variance
- Sometimes high variation means useful features for classification:



- .. and sometimes not:



Exhaustive feature subset selection

- If – for some reason – you know that you will use d out of D available features, an exhaustive search will involve a number of combinations to test:

$$n = \frac{D!}{(D-d)!\,d!}$$

d!=1*2*..*d

- If we want to perform an exhaustive search through D features for the optimal subset of the d ≤ m "best features", the number of combinations to test is

$$n = \sum_{d=1}^{m} \frac{D!}{(D-d)!\,d!}$$

- Impractical even for a moderate number of features!

d ≤ 5, D = 100 => n = 79.374.995

Suboptimal feature selection

- Select the best single features based on some quality criteria, e.g., estimated correct classification rate.
  - A combination of the best single features will often imply correlated features and will therefore be suboptimal .
- "Sequential forward selection" implies that when a feature is selected or removed, this decision is final.
- "Stepwise forward-backward selection" overcomes this.
  - A special case of the "add - a, remove - r algorithm".
- Improved into "floating search" by making the number of forward and backward search steps data dependent.
  - "Adaptive floating search"
  - "Oscillating search".

Distance measures used in feature selection

- In feature selection, each feature combination must be ranked based on a criterion function.
- Criteria functions can either be distances between classes, or the classification accuracy on a validation test set.
- If the criterion is based on e.g. the mean values/covariance matrices for the training data, distance computation is fast.
- Better performance at the cost of higher computation time is found when the classification accuracy on a validation data set (different from training and testing) is used as criterion for ranking features.
  - This will be slower as classification of the validattion data needs to be done for every combination of features.

Distance measures between classes

- How do we compute the distance 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?
    - Euclidean distance or Mahalanobis distance?
- Distance between K classes:
  - How do we generalize to more than two classes?
  - Average distance between the classes?
  - Smallest distance between a pair of classes?

Class separability measures

- How do we get an indication of the separability between two classes?
  - Euclidean distance between class means $|\mu_r - \mu_s|$
  - 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:

$$\Delta = (\mu_1 - \mu_2)^T \Sigma^{-1} (\mu_1 - \mu_2)$$

$$\Sigma = N_1\Sigma_1 + N_2\Sigma_2$$

Examples of feature selection - Method 1 - Individual feature selection

- Each feature is treated individually (no correlation/covariance between features is consideren)
- 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.

Method 2 - Sequential backward selection

- Select l features out of d
- 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((d+1)d-l(l+1))

Method 3: 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: $ld-l(l-1)/2$.
  - Backwards selection is faster if l is closer to d than to 1.

Learning goals for this lecture

- Understand how different measures of classification accuracy work:
  - Confusion matrix
  - Sensitivity/specifity/TP/TN/FP/FN/ROC
  - Average classification accuracy
- Be familiar with the curse of dimensionality and the importance of selecting few, but good features
- Know simple forward and backward feature selection.
- Know how PCA works
  - Max variance <-> min projection error
  - Eigenvectors of sample cov.mat. / scatter matrix