CS 1675 Introduction to Machine Learning
Lecture 22

# Dimensionality reduction
# Feature selection

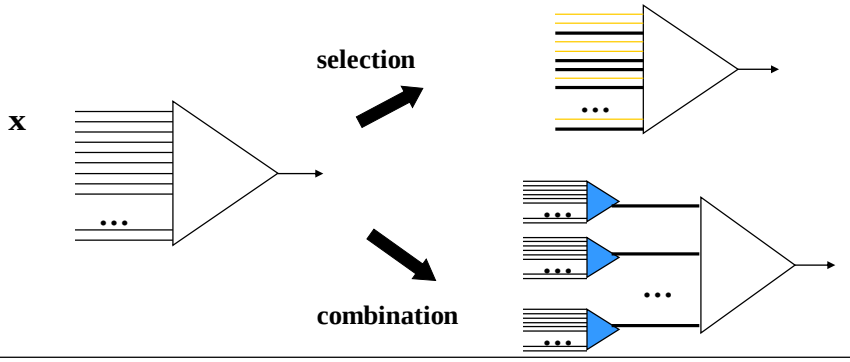
Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

## Dimensionality reduction. Motivation.

- **ML methods are sensitive to the dimensionality *d* of data**
- **Question:** Is there a lower dimensional representation of the data that captures well its characteristics?
- **Objective of dimensionality reduction:**
  - **Find a lower dimensional representation of data**
- **Two learning problems:**
  - **Supervised** $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), .., (\mathbf{x}_n, , y_n)\}$
    $\mathbf{x}_i = (x_i^1, x_i^2, .., x_i^d)$
  - **Unsupervised** $D = \{\mathbf{x}_1, \mathbf{x}_2, .., \mathbf{x}_n\}$
    $\mathbf{x}_i = (x_i^1, x_i^2, .., x_i^d)$
- **Goal: replace** $\mathbf{x}_i = (x_i^1, x_i^2, .., x_i^d)$
  **with** $\mathbf{x}_i'$ **of dimensionality d'< d**

## Dimensionality reduction

- **Solutions**:
  - **Selection of a smaller subset** of inputs (features) from a large set of inputs; train classifier on the reduced input set
  - **Combination of high dimensional inputs** to a smaller set of features $\phi_k(\mathbf{x})$; train classifier on new features




## Task-dependent feature selection

**Assume: Classification problem**:

- $\mathbf{x}$ – input vector, $y$ - output

**Objective:** Find a subset of inputs/features that gives/preserves most of the output prediction capabilities

**Selection approaches:**

- **Filtering approaches**
  - Filter out features with small predictive potential
  - Done before classification; typically uses univariate analysis
- **Wrapper approaches**
  - Select features that directly optimize the accuracy of the multivariate classifier
- **Embedded methods**
  - Feature selection and learning closely tied in the method
  - Regularization methods, decision tree methods

# Feature selection through filtering

**Assume:** **Classification problem**: **x** – input vector, $y$ - output

**How to select the features/inputs?**

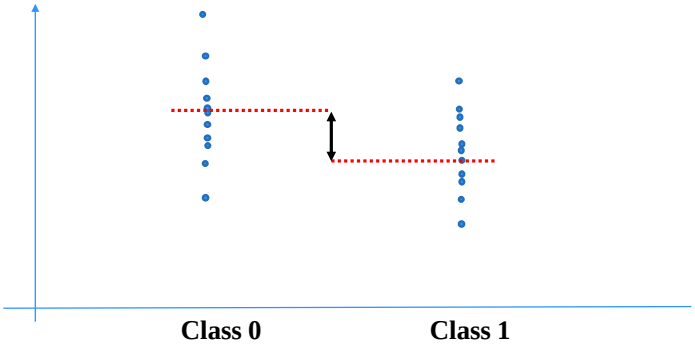
- **Step 1 .** For each input $x_i$ in data calculate $Score(x_i, y)$ reflecting how well $x_i$ predicts the output $y$ alone
- **Step 2.** Pick a subset of inputs with the best scores $Score(x_i, y)$ (or equivalently eliminate/filter the inputs with the worst scores)

| $x_1$ | $x_2$ | $x_3$ | | $x_{100}$ | $y$ |
|---|---|---|---|---|---|
| 1.2 | 3.3 | 0.2 | | 9.3 | 1 |
| 7.5 | 3.7 | 8.6 | | 2.1 | 0 |
| | | | | | |
| | | | | | |
| 1.3 | 2.6 | 6.5 | | 7.5 | 1 |

$Score(x_1, y)$ $Score(x_2, y)$ $Score(x_3, y)$ $Score(x_{100}, y)$

# Feature scoring for classification

- **Scores for measuring the differential expression**
  - **T-Test score** (Baldi & Long)
    - Based on the test that two groups come from the same population
    - Null hypothesis: **is mean of class 0 = mean of class 1**

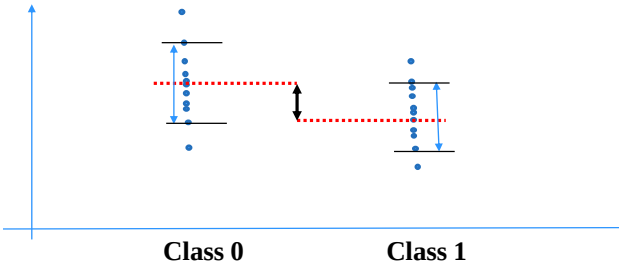


# Feature scoring for classification

**Scores for measuring the differential expression**

- **Fisher Score**

$$Fisher(i) = \frac{(\mu_i^{(+)} - \mu_i^{(-)})^2}{\sigma_i^{(+)^2} + \sigma_i^{(-)^2}}$$




- **AUROC score:** Area under Receiver Operating Characteristic curve

# Feature scoring

- **Correlation coefficients**
  - **Measures linear dependences**

$$\rho(x_k, y) = \frac{Cov(x_k, y)}{\sqrt{Var(x_k)Var(y)}}$$

- **Mutual information**
  - **Measures dependences**
  - **Needs discretized input values**

$$I(x_k, y) = \sum_i \sum_j \tilde{P}(x_k = j, y = i) \log_2 \frac{\tilde{P}(x_k = j, y = i)}{\tilde{P}(x_k = j)\tilde{P}(y = i)}$$

# Feature/input dependences

**Univariate score assumptions:**

- Only one input and its effect on *y* is incorporated in the score
- Effects of two features on *y* are considered to be independent

**Correlation based feature selection**

- A partial solution to the above problem
- **Idea:** good feature subsets contain features that are highly correlated with the class but independent of each other
- **Assume a set of features S of size *d*.** Then

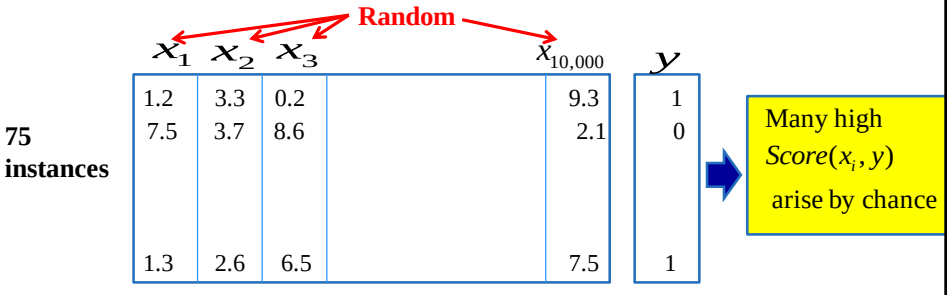
$$M(S) = \frac{d\bar{r}_{yx}}{\sqrt{d + d(d+1)\bar{r}_{xx}}}$$

- Average correlation between x and class y $\bar{r}_{yx}$
- Average correlation between pairs of xs $\bar{r}_{xx}$

# Feature selection: low sample size

**Problems: Many inputs and low sample size**

- if many random features, and not many instances we can learn from, the features with a good predictive score **may arise simply by chance.** The probability of this can be quite large.




- Techniques to address the problem:
  - reduce **FDR** (False discovery rate) and
  - **FWER** (Family wise error)

## Feature selection: wrappers

**Wrapper approach**:

- The input/feature selection is driven by the prediction accuracy of the classifier (regressor) we actually want to built

**Two problems:**

**How to judge the quality of a subset of inputs on the model?**

**How to find the best subset of inputs out of d inputs efficiently?**

## Feature selection: wrappers

**Wrapper approach**:

- The input/feature selection is driven by the prediction accuracy of the classifier (regressor) we actually want to built

**Two problems:**

**How to judge the quality of a subset of inputs on the model?**

- Internal cross-validation (k-fold cross validation)

## Internal cross-validation

- **Split train set: to internal train and test sets**
- **Internal train set: train different models** (defined e.g. on different subsets of features)
- **Internal test set/s**: **estimate the generalization error** and select the best model among possible models
- **Internal cross-validation (*k*-fold):**
  - Divide the train data into *m* equal partitions (of size *N/k*)
  - Hold out one partition for validation, train the classifiers on the rest of data
  - Repeat such that every partition is held out once
  - The estimate of the generalization error of the learner is the mean of errors of on all partitions

## Feature selection: wrappers

**Wrapper approach**:

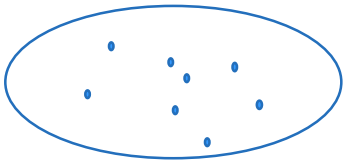
- The input/feature selection is driven by the prediction accuracy of the classifier (regressor) we actually want to built

**Two problems:**

**How to judge the quality of a subset of inputs on the model?**

- Internal cross-validation (k-fold cross validation)

**How to find the best subset of inputs out of d inputs efficiently?**



**d inputs**

## Feature selection: wrappers

**Wrapper approach**:

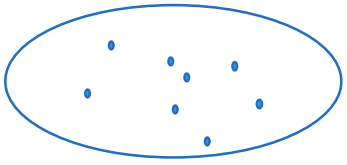
- The input/feature selection is driven by the prediction accuracy of the classifier (regressor) we actually want to built

**Two problems:**

**How to judge the quality of a subset of inputs on the model?**

- Internal cross-validation (k-fold cross validation)

**How to find the best subset of inputs out of d inputs efficiently?**



**d inputs**

**For *d* inputs/features there are $2^d$ different input subsets to evaluate and compare**

---

## Feature selection: wrappers

**How to find the appropriate feature subset S efficiently?**

- For *d* inputs/features there are $2^d$ different feature subsets

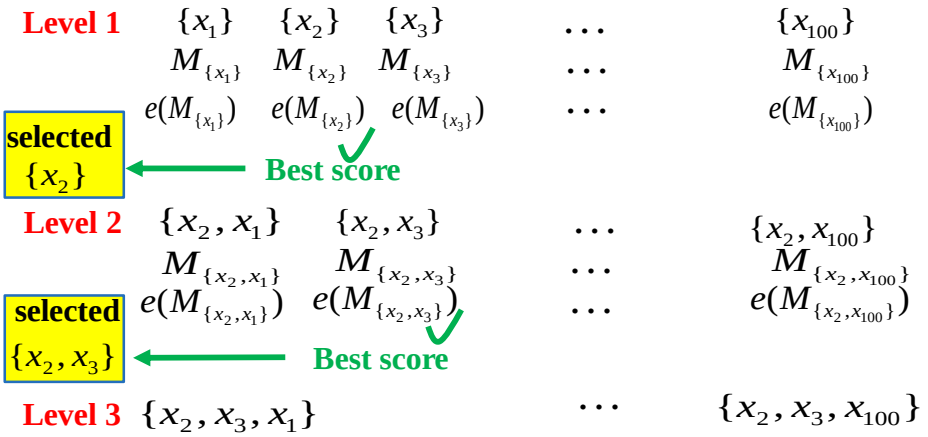
- **Solution : Greedy search in the space of classifiers**
  - **Option 1: Build the set incrementally**
    - Add features one by one. Add features that improve the quality of the model the most
  - **Option 2: Gradually remove features**
    - Remove features that effect the accuracy the least

- **Model quality:**
  - Internal cross-validation (k-fold cross validation)

---

## Feature selection: wrappers

**Greedy selection**




## Feature selection: wrappers

**Stopping criterion:**

- **Compare:**
  - **The best score at the previous level k-1**
  - **The best score at the current level k**
- **Stop when there is a decrease in performance on the set of features at level k**

## Embedded methods

**Feature selection + model learning** done jointly

- **Examples of embedded methods:**
  - **Regularized models**
    - Models of higher complexity are explicitly penalized leading to 'virtual' removal of inputs from the model
    - **Covers:**
      - Regularized logistic/linear regression
      - Support vector machines
        - » Optimization of margins penalizes nonzero weights

$$\underbrace{J_n(\mathbf{w}, D)}_{\text{Function to optimize}} = \underbrace{L(\mathbf{w}, D)}_{\text{Loss function (fit of the data)}} + \underbrace{R(\mathbf{w})}_{\text{Regularization penalty}}$$

  - **CART/Decision trees**

## Unsupervised dimensionality reduction

- **Is there a lower dimensional representation of the data that captures well its characteristics?**
- **Assume:**
  - We have data $D = \{\mathbf{x}_1, \mathbf{x}_2, .., \mathbf{x}_N\}$ such that $\mathbf{x}_i = (x_i^1, x_i^2, .., x_i^d)$
  - Assume the dimension $d$ of the data point $\boldsymbol{x}$ is very large
  - We want to analyze $\boldsymbol{x}$, there is no class label $y$
- **Our goal:**
  - **Find a lower dimensional representation of data of dimension $d' < d$**

## Principal component analysis (PCA)

**Objective:** We want to replace a high-dimensional input vector with a lower dimension vector (obtained by combining inputs)

– Different from the feature subset selection !!!

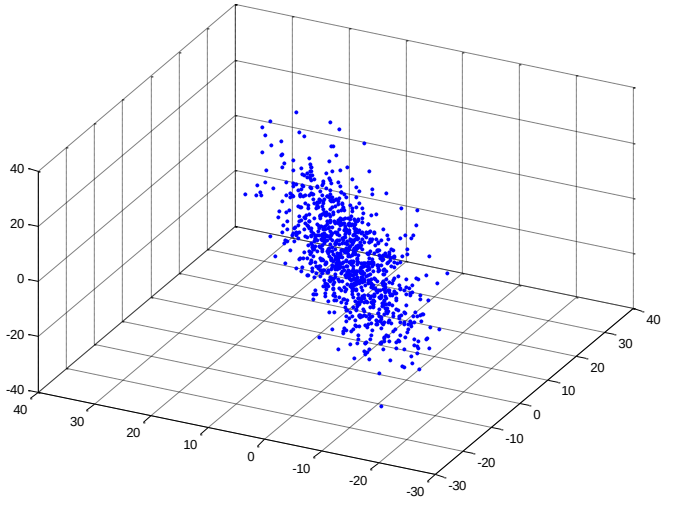
**PCA:**

- A linear transformation of the *d* dimensional input *x* to the M dimensional feature vector *z* such that $M < d$

$$\mathbf{z} = \mathbf{A}\mathbf{x}$$

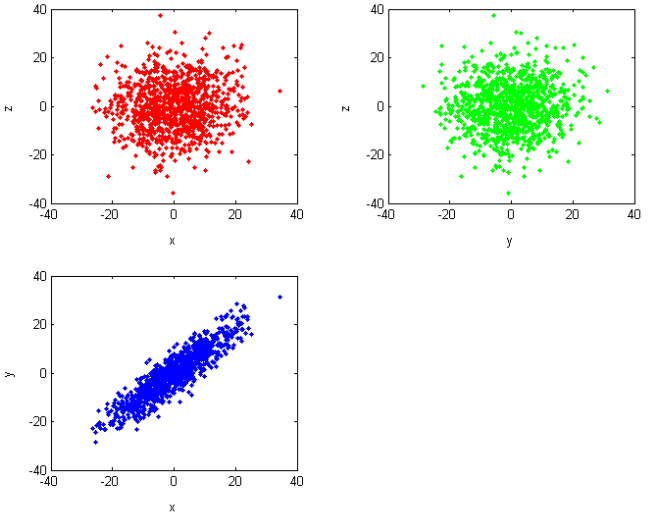
- Many different transformations exists, which one to pick?
- PCA –selects the linear transformation for which **the retained variance is maximal**
- Or, equivalently it is the linear transformation for which the sum of squares reconstruction cost is minimized

## PCA: example



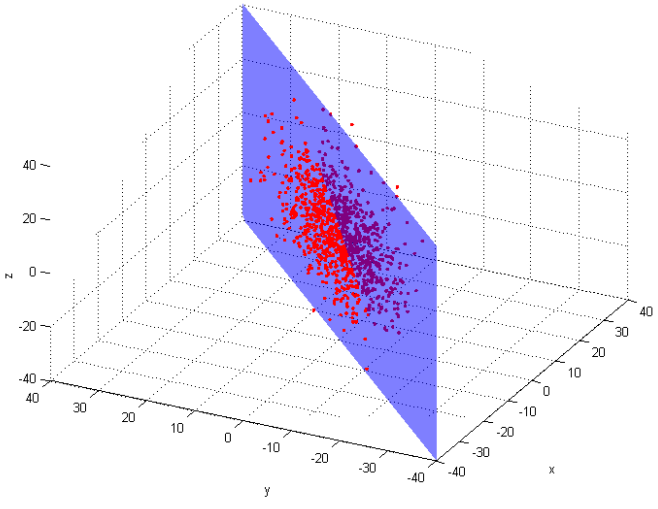
# PCA

**Projections to different axis**



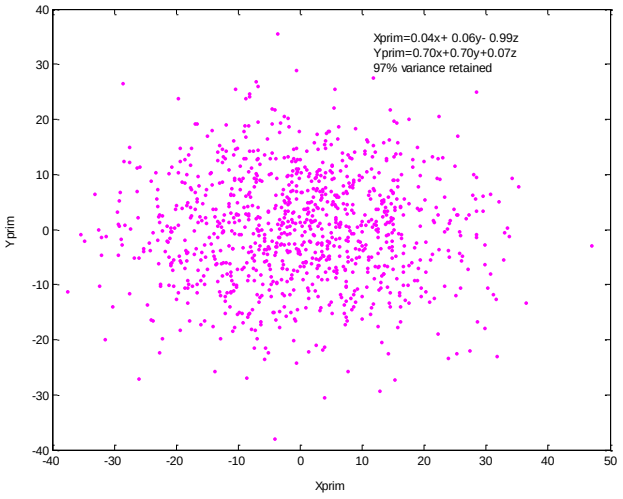

# PCA

- PCA projection to the 2 dimensional space



# PCA

- PCA projection to the 2 dimensional space




# Principal component analysis (PCA)

- **PCA:**
  - linear transformation of a $d$ dimensional input $\mathbf{x}$ to M dimensional vector $\mathbf{z}$ such that $M < d$ under which the retained variance is maximal. **Remember:** no y is needed
- **Fact:**
  - A vector $\mathbf{x}$ can be represented using a set of orthonormal vectors $\mathbf{u}$ (basis vectors)

$$\mathbf{x} = \sum_{i=1}^{d} z_i \mathbf{u}_i$$

  - Leads to transformation of coordinates (from $\mathbf{x}$ to $\mathbf{z}$ using $\mathbf{u}$'s)

$$z_i = \mathbf{u}_i^T \mathbf{x} \qquad \mathbf{z} = \mathbf{U}\mathbf{x} \qquad \mathbf{U} = \begin{bmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \\ .. \\ \mathbf{u}_d^T \end{bmatrix}$$

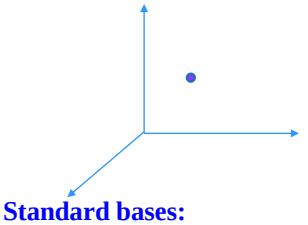
# Principal component analysis (PCA)

- **Fact:** A vector **x** can be represented using a set of orthonormal vectors **u** (basis vectors)

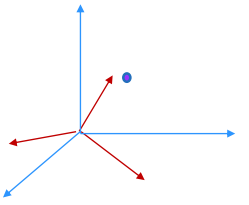
$$\mathbf{x} = \sum_{i=1}^{d} z_i \mathbf{u}_i$$

  - Leads to transformation of coordinates (from **x** to **z** using **u**'s)

$$z_i = \mathbf{u}_i^T \mathbf{x} \qquad \mathbf{z} = \mathbf{U}\mathbf{x} \qquad \mathbf{U} = \begin{bmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \\ .. \\ \mathbf{u}_d^T \end{bmatrix}$$



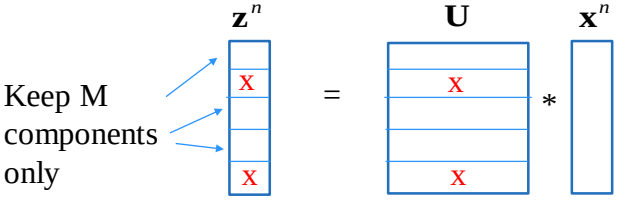
**Standard bases: (1,0,0); (0,1,0); (0,0,1)**



**New bases: $\mathbf{u}_1$ , $\mathbf{u}_2$ , $\mathbf{u}_3$**

# PCA

- **Idea:** represent d-dimensional $\mathbf{x}^n$ with an *M*-dimensional $\mathbf{z}^n$ formed by subset of $z_i$ coordinates for the bases defined by **U** .




- **Goal:** We want to find:

  (1) Basis vectors U and (2) a subset *of basis of size M* to keep

- **This effectively replaces $\mathbf{x}^n$ with its approximation $\tilde{\mathbf{x}}^n$**

$$\mathbf{x}^n = \sum_{i=1}^{d} z_i^n \mathbf{u}_i \quad \Rightarrow \quad \tilde{\mathbf{x}}^n = \sum_{i=1}^{M} z_i^n \mathbf{u}_i + \sum_{i=M+1}^{d} b_i \mathbf{u}_i$$

$b_i$ - constant and fixed for all data-points

# PCA

- **Goal:** We want to find: $z_i$

  Basis vectors U and a subset *of basis of size M* to keep

$$\mathbf{x}^n = \sum_{i=1}^{d} z_i^n \mathbf{u}_i \quad \Rightarrow \quad \tilde{\mathbf{x}}^n = \sum_{i=1}^{M} z_i^n \mathbf{u}_i + \sum_{i=M+1}^{d} b_i \mathbf{u}_i$$

  $b_i$ - constant and fixed for all data-points

- **How to choose the best set of basis vectors?**
  - We want the subset that gives the best approximation of data *x* in the dataset on average (we use least squares fit)

  Error for data entry $\mathbf{x}^n$ $\quad \mathbf{x}^n - \tilde{\mathbf{x}}^n = \sum_{i=M+1}^{d} (z_i^n - b_i)\mathbf{u}_i$

  **Reconstruction error**

$$E_M = \frac{1}{2}\sum_{n=1}^{N} \left\| \mathbf{x}^n - \tilde{\mathbf{x}}^n \right\| = \frac{1}{2}\sum_{n=1}^{N} \sum_{i=M+1}^{d} (z_i^n - b_i)^2$$

# PCA

- **Differentiate the error function** with regard to all $b_i$ and set equal to 0 we get:

$$b_i = \frac{1}{N}\sum_{n=1}^{N} z_i^n = \mathbf{u}_i^T \bar{\mathbf{x}} \qquad \bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}^n$$

- Then we can rewrite:

$$E_M = \frac{1}{2}\sum_{i=M+1}^{d} \mathbf{u}_i^T \boldsymbol{\Sigma} \mathbf{u}_i \qquad \boldsymbol{\Sigma} = \sum_{n=1}^{N} (\mathbf{x}^n - \bar{\mathbf{x}})(\mathbf{x}^n - \bar{\mathbf{x}})^T$$

- The error function is optimized when basis vectors satisfy:

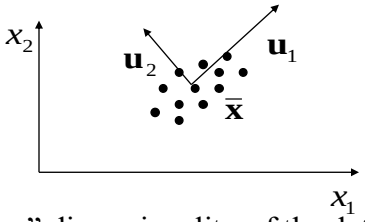
$$\boldsymbol{\Sigma}\mathbf{u}_i = \lambda_i \mathbf{u}_i \qquad E_M = \frac{1}{2}\sum_{i=M+1}^{d} \lambda_i$$

**The best *M* basis vectors**: discard vectors with *d-M* smallest eigenvalues (or keep vectors with M largest eigenvalues)

Eigenvector $\mathbf{u}_i$ – is called a **principal component**

# PCA

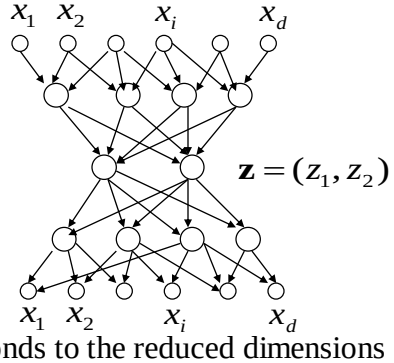
- Once eigenvectors $\mathbf{u}_i$ with largest eigenvalues are identified, they are used to transform the original *d*-dimensional data to *M* dimensions




- To find the "true" dimensionality of the data *d'* we can just look at eigenvalues that contribute the most (small eigenvalues are disregarded)
- **Problem:** PCA is a linear method. The "true" dimensionality can be overestimated. There can be non-linear correlations.
- **Modifications for nonlinearities:** kernel PCA

# Dimensionality reduction with neural nets

- **PCA** is limited to linear dimensionality reduction
- To do non-linear reductions we can use neural nets
- **Auto-associative (or auto-encoder) network:** a neural network with the same inputs and outputs ( ***x*** )



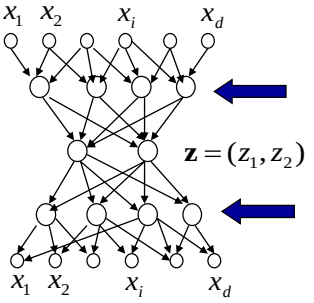

- The middle layer corresponds to the reduced dimensions

## Dimensionality reduction with neural nets

- **Error criterion:**

$$E = \frac{1}{2}\sum_{n=1}^{N}\sum_{i=1}^{d}\left(y_i(x^n) - x^n\right)^2$$

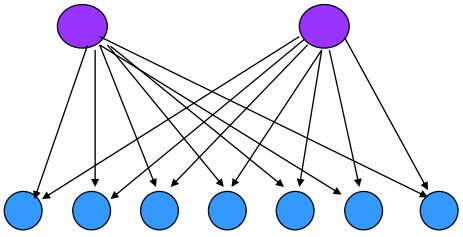
- Error measure tries to recover the original data through limited number of dimensions in the middle layer
- **Non-linearities** modeled through intermediate layers between the middle layer and input/output
- If no intermediate layers are used the model replicates PCA optimization through learning




## Latent variable models

- **Learning using unsupervised learning**
- **Dimensionality reduction via inference**

**Latent variables (s): Dimensionality k**



**Dimensionality reduction via inference**

**Observed variables x: real valued vars**
**Dimensionality d**