

# Learning From Data

## Lecture 27

### Learning Aides

Input Preprocessing  
 Dimensionality Reduction and Feature Selection  
 Principal Components Analysis (PCA)  
 Hints, Data Cleaning, Validation, ...

M. Magdon-Ismail  
 CSCI 4100/6100

## Learning Aides

Additional tools that can be applied to all techniques

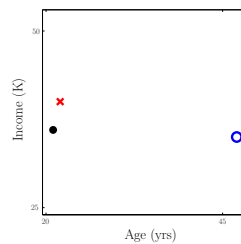
- Preprocess data to account for arbitrary choices during data collection (input normalization)
- Remove irrelevant dimensions that can mislead learning (PCA)
- Incorporate known properties of the target function (hints and invariances)
- Remove detrimental data (deterministic and stochastic noise)
- Better ways to validate (estimate  $E_{out}$ ) for model selection

## Nearest Neighbor

Mr. Good and Mr. Bad were both given credit cards by the Bank of Learning (BoL).

	Mr. Good	Mr. Bad
(Age in years, Income in \$ × 1,000)	(47,35)	(22,40)

Mr. Unknown who has “coordinates” (21yrs,\$36K) applies for credit. Should the BoL give him credit, according to the nearest neighbor algorithm?



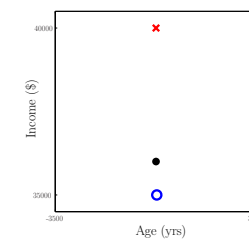
## Nearest Neighbor Uses Euclidean Distance

Mr. Good and Mr. Bad were both given credit cards by the Bank of Learning (BoL).

	Mr. Good	Mr. Bad
(Age in years, Income in \$)	(47,35000)	(22,40000)

Mr. Unknown who has “coordinates” (21yrs,\$36000) applies for credit. Should the BoL give him credit, according to the nearest neighbor algorithm?

What if, income is measured in dollars instead of “K” (thousands of dollars)?



## Uniform Treatment of Dimensions

Most learning algorithms treat each dimension equally

Nearest neighbor:  $d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|$

Weight Decay:  $\Omega(\mathbf{w}) = \lambda \mathbf{w}^T \mathbf{w}$

SVM: margin defined using **Euclidean** distance

RBF: bump function decays with **Euclidean** distance

### Input Preprocessing

Unless you want to emphasize certain dimensions, the data should be *preprocessed* to present each dimension on an equal footing

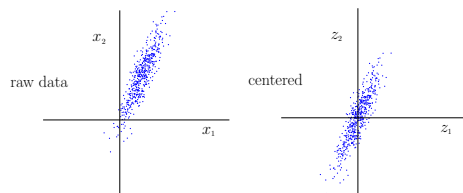
## Input Preprocessing is a Data Transform

$$X = \begin{bmatrix} - & \mathbf{x}_1^T & - \\ - & \mathbf{x}_2^T & - \\ & \vdots & \\ - & \mathbf{x}_N^T & - \end{bmatrix} \quad \mathbf{x}_n \mapsto \mathbf{z}_n$$

$$g(\mathbf{x}) = \tilde{g}(\Phi(\mathbf{x})).$$

Raw  $\{\mathbf{x}_n\}$  have (for example) arbitrary scalings in each dimension, and  $\{\mathbf{z}_n\}$  will not.

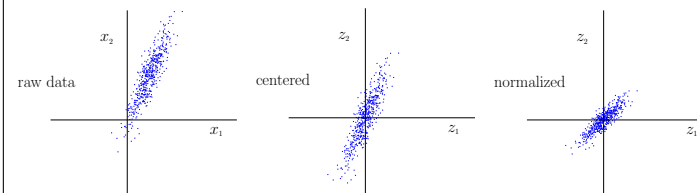
## Centering



$$\mathbf{z}_n = \mathbf{x}_n - \bar{\mathbf{x}}$$

$$\bar{\mathbf{z}} = \mathbf{0}$$

## Normalizing



$$\mathbf{z}_n = \mathbf{x}_n - \bar{\mathbf{x}}$$

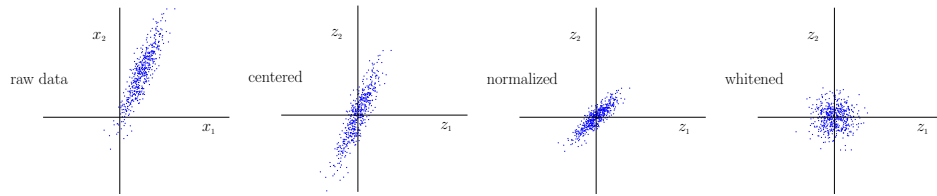
$$\mathbf{z}_n = D\mathbf{x}_n$$

$$D_{ii} = \frac{1}{\sigma_i}$$

$$\bar{\mathbf{z}} = \mathbf{0}$$

$$\tilde{\sigma}_i = 1$$

## Whitening



$$\mathbf{z}_n = \mathbf{x}_n - \bar{\mathbf{x}}$$

$$\mathbf{z}_n = \mathbf{D}\mathbf{x}_n$$

$$\mathbf{z}_n = \Sigma^{-\frac{1}{2}}\mathbf{x}_n$$

$$D_{ii} = \frac{1}{\sigma_i}$$

$$\Sigma = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^T = \frac{1}{N} \mathbf{X}^T \mathbf{X}$$

$$\bar{\mathbf{z}} = \mathbf{0}$$

$$\tilde{\sigma}_i = 1$$

$$\tilde{\Sigma} = \frac{1}{N} \mathbf{Z}^T \mathbf{Z} = \mathbf{I}$$

## Only Use Training Data For Preprocessing



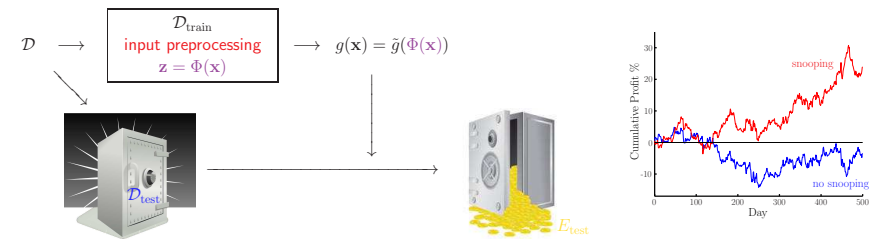
### WARNING!



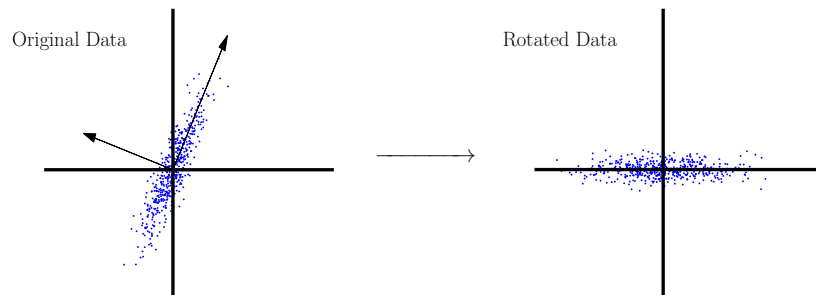
Transforming data into a more convenient format has a hidden trap which leads to data snooping.

When using a test set, determine the input transformation from training data *only*.

Rule: lock away the test data until you have your **final** hypothesis.



## Principal Components Analysis



Rotate the data so that it is easy to

Identify the dominant directions (information)

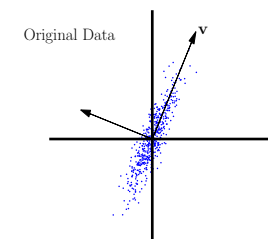
Throw away the smaller dimensions (noise)

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \rightarrow [z_1]$$

## Projecting the Data to Maximize Variance

(Always center the data first)

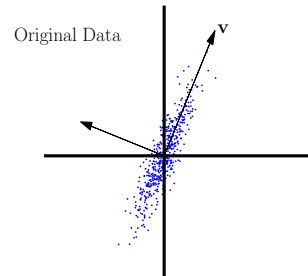
$$z = \mathbf{x}_n^T \mathbf{v}$$



Find  $\mathbf{v}$  to maximize the variance of  $z$

## Maximizing the Variance

$$\begin{aligned} \text{var}[z] &= \frac{1}{N} \sum_{n=1}^N z_n^2 \\ &= \frac{1}{N} \sum_{n=1}^N \mathbf{v}^T \mathbf{x}_n \mathbf{x}_n^T \mathbf{v} \\ &= \mathbf{v}^T \left( \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^T \right) \mathbf{v} \\ &= \mathbf{v}^T \Sigma \mathbf{v}. \end{aligned}$$



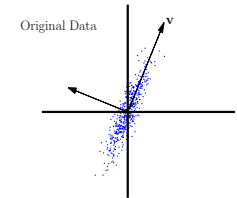
Choose  $\mathbf{v}$  as  $\mathbf{v}_1$ , the top eigenvector of  $\Sigma$  — the top principal component (PCA)

## The Principal Components

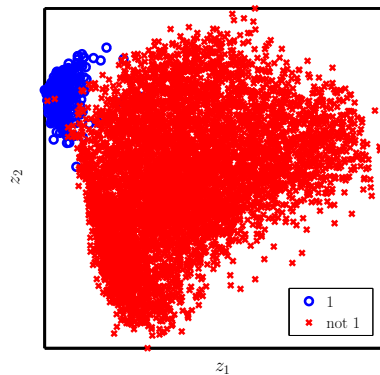
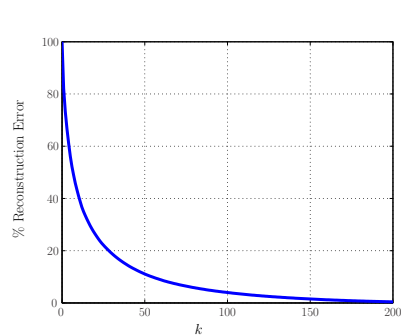
$$\begin{aligned} z_1 &= \mathbf{x}^T \mathbf{v}_1 \\ z_2 &= \mathbf{x}^T \mathbf{v}_2 \\ z_3 &= \mathbf{x}^T \mathbf{v}_3 \\ &\vdots \end{aligned}$$

$\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_d$  are the eigenvectors of  $\Sigma$  with eigenvalues  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d$

**Theorem [Eckart-Young].** These directions give best reconstruction of data; also capture maximum variance.



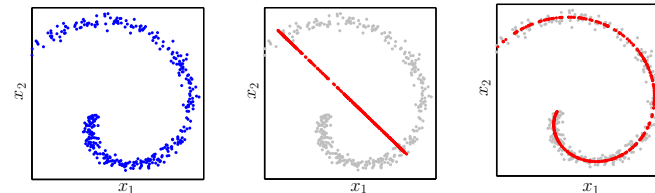
## PCA Features for Digits Data



Principal components are **automated**  
Captures dominant directions of the data.  
May not capture dominant dimensions for  $f$ .

## Other Learning Aides

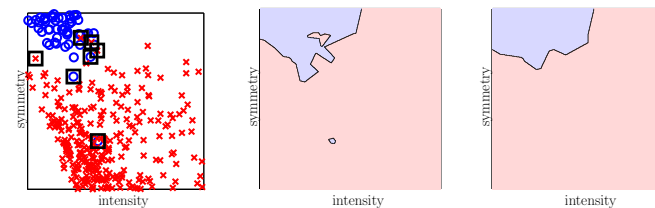
### 1. Nonlinear dimension reduction:



### 2. Hints (invariances and prior information):

rotational invariance, monotonicity, symmetry, ...

### 3. Removing noisy data:



### 4. Advanced validation techniques: Rademacher and Permutation penalties

More efficient than CV, more convenient and accurate than VC.