CS 2750 Machine Learning

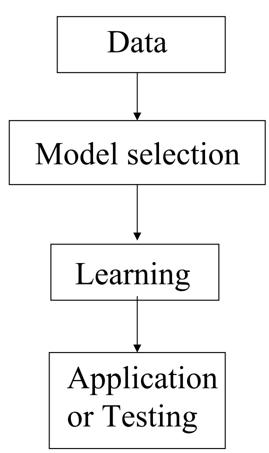
Lecture 2

Designing a learning system

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square, x4-8845

http://www.cs.pitt.edu/~milos/courses/cs2750/

Design of a learning system (first view)



Design of a learning system.

1. Data: $D = \{d_1, d_2, .., d_n\}$

2. Model selection:

- **Select a model** or a set of models (with parameters)

  E.g. $y = ax + b + \varepsilon \qquad \varepsilon = N(0, \sigma)$

- **Select the error function** to be optimized

  E.g. $\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$

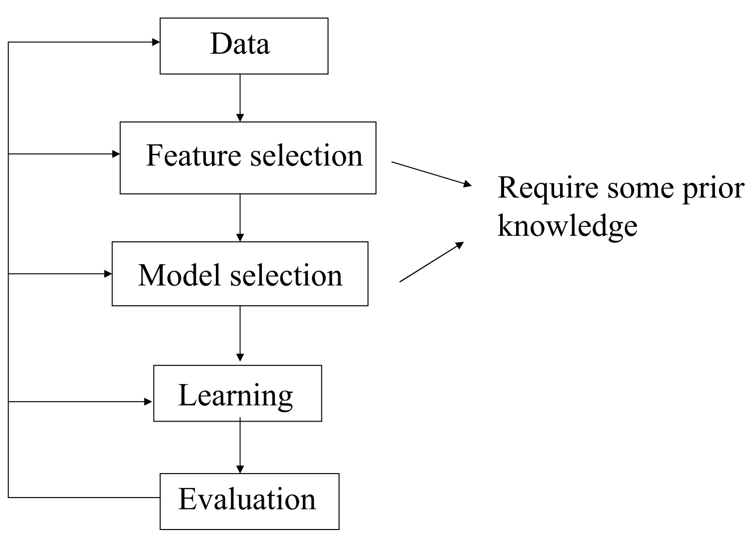
3. Learning:

- **Find the set of parameters optimizing the error function**
  - The model and parameters with the smallest error

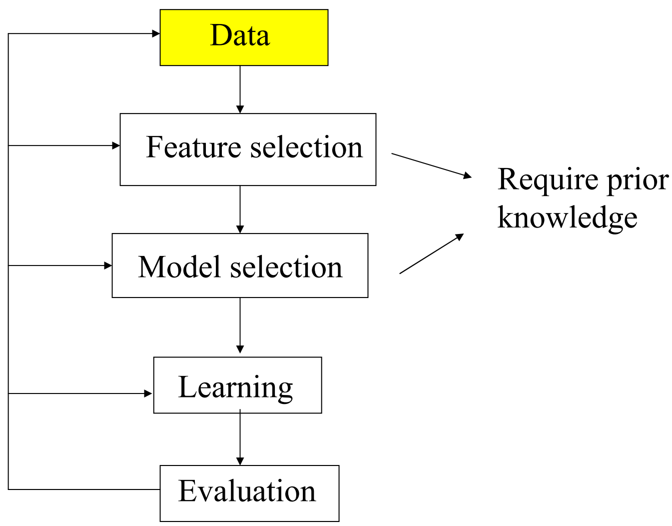
4. Application (Evaluation):

- **Apply the learned model**
  - E.g. predict ys for new inputs **x** using learned $f(\mathbf{x})$

Design cycle



Design cycle



Data

Data may need a lot of:
- **Cleaning**
- **Preprocessing (conversions)**

Cleaning:
- **Get rid of errors, noise,**
- **Removal of redundancies**

Preprocessing:
- **Renaming**
- **Rescaling (normalization)**
- **Discretizations**
- **Abstraction**
- **Agreggation**
- **New attributes**

Data preprocessing

- **Renaming (**relabeling) categorical values to numbers
  - dangerous in conjunction with some learning methods
  - numbers will impose an order that is not warrantied
- **Rescaling (normalization):** continuous values transformed to some range, typically [-1, 1] or [0,1].
- **Discretizations (binning):** continuous values to a finite set of discrete values
- **Abstraction:** merge together categorical values
- **Aggregation:** summary or aggregation operations, such minimum value, maximum value etc.
- **New attributes:**
  - example: obesity-factor = weight/height

Data biases

- **Watch out for data biases:**
  - Try to understand the data source
  - It is very easy to derive "unexpected" results when data used for analysis and learning are biased (pre-selected)
  - Results (conclusions) derived for pre-selected data do not hold in general !!!

Data biases

Example 1: Risks in pregnancy study

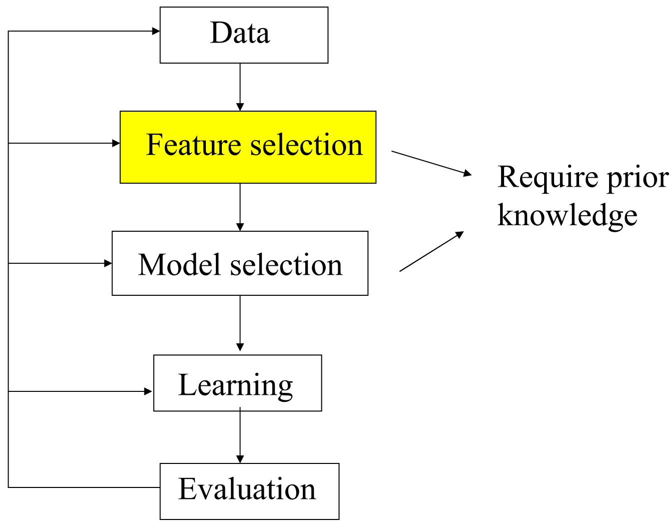
- Sponsored by DARPA at military hospitals
- Study of a large sample of pregnant woman
- **Conclusion:** the factor with the largest impact on reducing risks during pregnancy (statistically significant) is a pregnant woman being single
- Single woman -> the smallest risk
- What is wrong?

Data

Example 2: Stock market trading (example by Andrew Lo)

- Data on stock performances of companies traded on stock market over past 25 year
- **Investment goal:** pick a stock to hold long term
- **Proposed strategy**: invest in a company stock with an IPO corresponding to a Carmichael number
- **Evaluation result:** excellent return over 25 years
- Where the magic comes from?

Design cycle

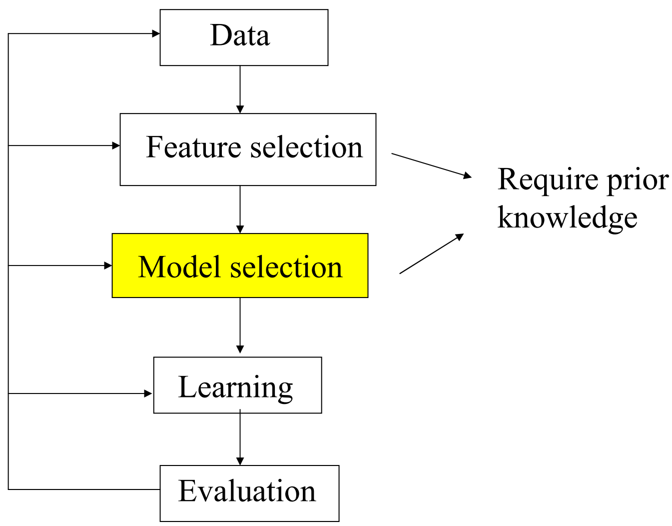


Feature selection

- **The size (dimensionality) of a sample** can be enormous

  $x_i = (x_i^1, x_i^2, .., x_i^d)$ $\quad d$ - very large
- **Example: document classification**
  - 10,000 different words
  - Inputs: counts of occurrences of different words
  - Too many parameters to learn (not enough samples to justify the estimates the parameters of the model)
- **Dimensionality reduction: replace inputs with features**
  - **Extract relevant inputs** (e.g. mutual information measure)
  - **PCA** – principal component analysis
  - **Group (cluster) similar words** (uses a similarity measure)
    - Replace with the group label

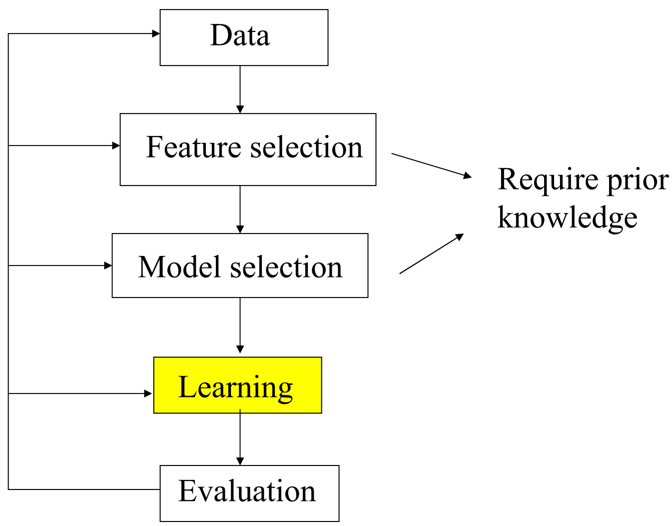
Design cycle



Model selection

- **What is the right model to learn?**
  - A prior knowledge helps a lot, but still a lot of guessing
  - Initial data analysis and visualization
    - We can make a good guess about the form of the distribution, shape of the function
  - Independences and correlations
  - **Overfitting problem**
    - Take into account the **bias and variance** of error estimates

Design cycle



Learning

- **Learning = optimization problem**. Various criteria:
  - **Mean square error**

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} Error(\mathbf{w}) \qquad Error(\mathbf{w}) = \frac{1}{N}\sum_{i=1,..N}(y_i - f(x_i, \mathbf{w}))^2$$

  - **Maximum likelihood (ML) criterion**

$$\Theta^* = \max_{\Theta} P(D \mid \Theta) \qquad Error(\Theta) = -\log P(D \mid \Theta)$$

  - **Maximum posterior probability (MAP)**

$$\Theta^* = \max_{\Theta} P(\Theta \mid D) \qquad P(\Theta \mid D) = \frac{P(D \mid \Theta) P(\Theta)}{P(D)}$$

Learning

Learning = optimization problem

- Optimization problems can be hard to solve. Right choice of a model and an error function makes a difference.
- **Parameter optimizations**
  - Gradient descent, Conjugate gradient
  - Newton-Rhapson
  - Levenberg-Marquard

  Some can be carried **on-line** on a sample by sample basis

  Combinatorial optimizations (over discrete spaces):
  - Hill-climbing
  - Simulated-annealing
  - Genetic algorithms

Parametric optimizations

- Sometimes can be solved directly but this depends on the error function and the model
  – Example: squared error criterion for linear regression
- Very often the error function to be optimized is not that nice.

  $Error(\mathbf{w}) = f(\mathbf{w}) \qquad \mathbf{w} = (w_0, w_1, w_2 \ldots w_k)$

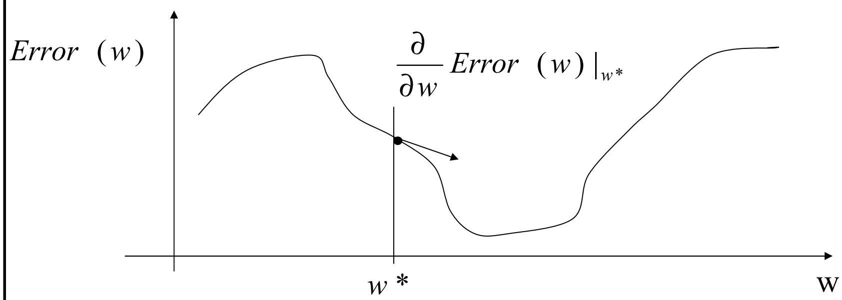
  - a complex function of weights (parameters)

  Goal: $\mathbf{w}^* = \arg\min_{\mathbf{w}} f(\mathbf{w})$
- Typical solution: **iterative methods**.
- **Example: Gradient-descent method**

  Idea: move the weights (free parameters) gradually in the error decreasing direction

Gradient descent method

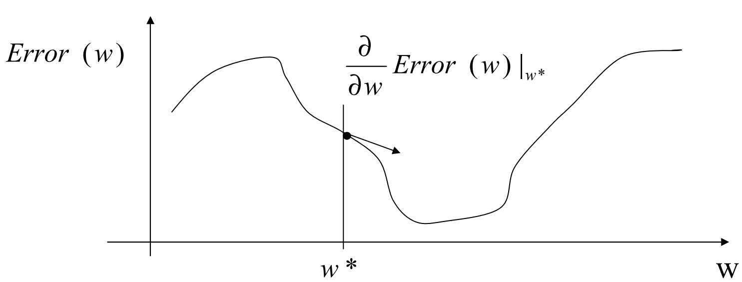
- Descend to the minimum of the function using the gradient information



- Change the parameter value of w according to the gradient

$$w \leftarrow w^* - \alpha \frac{\partial}{\partial w} Error(w)|_{w^*}$$

Gradient descent method



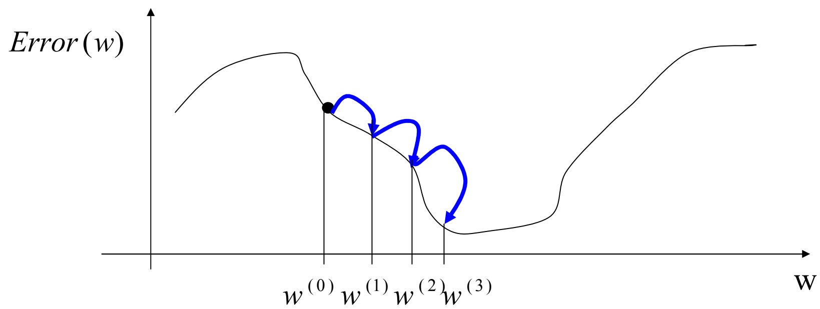
- New value of the parameter

$$w \leftarrow w^* - \alpha \frac{\partial}{\partial w} Error(w)|_{w^*}$$

$\alpha > 0$ - a learning rate (scales the gradient changes)

Gradient descent method

- To get to the function minimum repeat (iterate) the gradient based update few times



- **Problems:** local optima, saddle points, slow convergence
- More complex optimization techniques use additional information (e.g. second derivatives)

On-line learning (optimization)

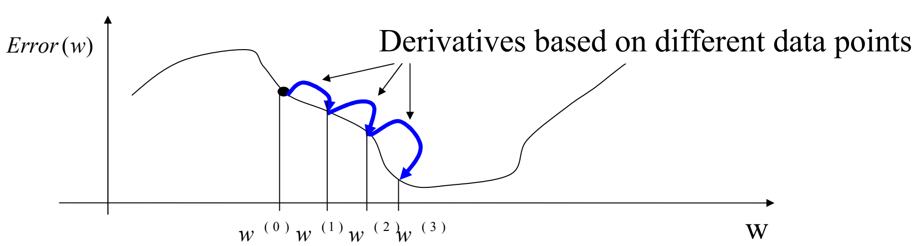
- Error function looks at all data points at the same time

  E.g. $$Error(\mathbf{w}) = \frac{1}{n}\sum_{i=1,..n}(y_i - f(x_i,\mathbf{w}))^2$$

- **On-line error** - separates the contribution from a data point

  $$Error_{\text{ON-LINE}}(\mathbf{w}) = (y_i - f(x_i,\mathbf{w}))^2$$

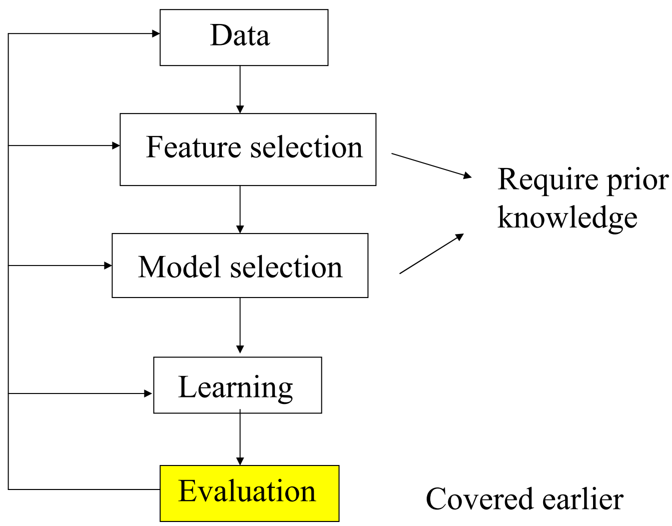
- **Example: On-line gradient descent**



- **Advantages: 1. simple learning algorithm**

  2. no need to store data (on-line data streams)

Design cycle



Evaluation.

- **Simple holdout method.**
  – Divide the data to the training and test data.
- **Other more complex methods**
  – Based on cross-validation, random sub-sampling.
- What if we want to compare the predictive performance on a classification or a regression problem for two different learning methods?
- **Solution:** compare the error results on the test data set
- **Possible answer**: the method with better (smaller) testing error gives a better generalization error.
- Is this a good answer? How sure are we about the method with a better test score being truly better?

Evaluation.

- **Problem:** we cannot be 100 % sure about generalization errors
- **Solution:** test the statistical significance of the result
- **Central limit theorem:**

  Let random variables $X_1, X_2, \cdots X_n$ form a random sample from a distribution with mean μ and variance σ, then if the sample n is large, the distribution

  $$\sum_{i=1}^{n} X_i \approx N(n\mu, n\sigma^2) \quad \text{or} \quad \frac{1}{n}\sum_{i=1}^{n} X_i \approx N(\mu, \sigma^2 / n)$$