A learning system: basic cycle

1. **Data:** \( D = \{d_1, d_2, ..., d_n\} \)
2. **Model selection:**
   - Select a model or a set of models (with parameters)
   
   E.g. \( y = ax + b \)
3. **Choose the objective function**
   - Squared error \( \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2 \)
4. **Learning:**
   - Find the set of parameters optimizing the error function
     - The model and parameters with the smallest error
5. **Testing/validation:**
   - Evaluate on the test data
6. **Application**
   - Apply the learned model to new data \( f(x) \)
A learning system: basic cycle

1. Data: \( D = \{d_1, d_2, \ldots, d_n\} \)

2. Model selection:
   - Select a model or a set of models (with parameters)
   - E.g. \( y = ax + b \)

3. Choose the objective function
   - Squared error
     \[ \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2 \]

4. Learning:
   - Find the set of parameters optimizing the error function
     - The model and parameters with the smallest error

5. Testing/validation:
   - Evaluate on the test data

6. Application
   - Apply the learned model to new data \( f(x) \)
Steps taken when designing an ML system

Data
Model selection
Choice of Error function
Learning/optimization
Evaluation
Application

Add some complexity

Data
Data cleaning/preprocessing
Feature selection/dimensionality reduction
Model selection
Choice of Error function
Learning/optimization
Evaluation
Application
Designing an ML solution

- Data
- Data cleaning/preprocessing
- Feature selection/dimensionality reduction
- Model selection
- Choice of Error function
- Learning/optimization
- Evaluation
- Application
Data source and data biases

- Understand the data source
- Understand the data your models will be applied to
- Watch out for data biases:
  - Make sure the data we make conclusions on are the same as data we used in the analysis
  - It is very easy to derive “unexpected” results when data used for analysis and learning are biased

• Results (conclusions) derived for a biased dataset do not hold in general !!!

Data biases

Example: Assume you want to build an ML program for predicting the stock behavior and for choosing your investment strategy

Data extraction:
- pick companies that are traded on the stock market on January 2017
- Go back 30 years and extract all the data for these companies
- Use the data to build an ML model supporting your future investments

Question:
- Would you trust the model?
- Are there any biases in the data?
Steps taken when designing an ML system

- Data
- Data cleaning/preprocessing
- Feature selection/dimensionality reduction
- Model selection
- Choice of Error function
- Learning/optimization
- Evaluation
- Application

Data cleaning and preprocessing

Data you receive may not be perfect:
- Cleaning
- Preprocessing (conversions)

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

Preprocessing:
- Renaming
- Rescaling (normalization)
- Discretizations
- Abstraction
- Aggregation
- 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 warranted

  - High $\rightarrow$ 2
  - Normal $\rightarrow$ 1
  - Low $\rightarrow$ 0
  - True $\rightarrow$ 2
  - False $\rightarrow$ 1
  - Unknown $\rightarrow$ 0
  - Red $\rightarrow$ 2
  - Blue $\rightarrow$ 1
  - Green $\rightarrow$ 0

- **Rescaling (normalization):** continuous values transformed to some range, typically [-1, 1] or [0,1].

![Diagram of rescaling](image)

Data preprocessing

- **Discretizations (binning):** continuous values to a finite set of discrete values

- **Example:**

![Diagram of discretizations](image)

- **Another Example:**

![Another diagram](image)
Data preprocessing

- **Abstraction:** merge together categorical values

- **Aggregation:** summary or aggregation operations, such as minimum value, maximum value, average etc.

- **New attributes:**
  - example: obesity-factor = weight/height

---

Steps taken when designing an ML system

- Data
- Data cleaning/preprocessing
- Feature selection/dimensionality reduction
- Model selection
- Choice of Error function
- Learning/optimization
- Evaluation
- Application
Feature selection

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

\[ x_i = (x_i^1, x_i^2, ..., x_i^d) \quad d \text{ - very large} \]

• **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

---

**Steps taken when designing an ML system**

- Data
- Data cleaning/preprocessing
- Feature selection/dimensionality reduction
- **Model selection**
- Choice of Error function
- Learning/optimization
- Evaluation
- Application
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
  – Simpler (more biased) model – parameters can be estimated more reliably (smaller variance of estimates)
  – Complex model with many parameters – parameter estimates are less reliable (large variance of the estimate)

Solutions for overfitting

**How to make the learner avoid the overfit?**

• **Assure sufficient number of samples** in the training set
  – May not be possible (small number of examples)

• **Hold some data out of the training set = validation set**
  – Train (fit) on the training set (w/o data held out);
  – Check for the generalization error on the validation set, choose the model based on the validation set error (cross-validation techniques)

• **Regularization (Occam’s Razor)**
  – Penalize for the model complexity (number of parameters) in the objective function
  – Explicit preference towards simple models
Steps taken when designing an ML system

- Data
- Data cleaning/preprocessing
- Feature selection/dimensionality reduction
- Model selection
- Choice of Error function
- Learning/optimization
- Evaluation
- Application

Learning: objective functions

- **Learning = optimization problem.** Various criteria:
  - Mean square error
    \[ w^* = \arg \min_w Error(w) \quad Error(w) = \frac{1}{N} \sum_{i=1}^{N} (y_i - f(x_i, w))^2 \]
  - Maximum likelihood (ML) criterion
    \[ \Theta^* = \max_\Theta P(D \mid \Theta) \quad Error(\Theta) = -\log P(D \mid \Theta) \]
  - Maximum posterior probability (MAP)
    \[ \Theta^* = \max_\Theta P(\Theta \mid D) \quad P(\Theta \mid D) = \frac{P(D \mid \Theta)P(\Theta)}{P(D)} \]
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 objective function and the model
  – Example: squared error criterion for the linear regression
• Very often the objective function to be optimized is not that nice.
  \[ Error(w) = f(w) \]
  - a complex function of weights (parameters)
  \[ \text{Goal: } w^* = \arg \min_w f(w) \]
• One solution: iterative optimization 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

\[ \frac{\partial}{\partial w} \text{Error}(w) \big|_{w^*} \]

\[ w^* \]

\[ w \]

• Change the parameter value of \( w \) according to the gradient

\[ w \leftarrow w^* - \alpha \frac{\partial}{\partial w} \text{Error}(w) \big|_{w^*} \]

\( \alpha > 0 \) - a learning rate (scales the gradient changes)
**Gradient descent method**

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

![Gradient descent method diagram](image)

- **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. \[ Error(w) = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i, w))^2 \]

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

\[ Error_{ON-LINE}(w) = (y_i - f(x_i, w))^2 \]

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

![On-line learning diagram](image)

- **Advantages**: 1. simple learning algorithm
  2. no need to store data (on-line data streams)
Steps taken when designing an ML system

- Data
- Data cleaning/preprocessing
- Feature selection/dimensionality reduction
- Model selection
- Choice of Error function
- Learning/optimization
- Evaluation
- Application

Evaluation of models

- Simple holdout method

Dataset

- Training set
- Testing set

Evaluate

Learn (fit)

Predictive model
Evaluation measures

Regression model $f: X \rightarrow Y$ where $Y$ is real valued

- Mean Squared Error
  
  \[ MSE(D, f) = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2 \]

- Mean Absolute Error
  
  \[ MAE(D, f) = \frac{1}{n} \sum_{i=1}^{n} |y_i - f(x_i)| \]

- Mean Absolute Percentage Error
  
  \[ MAPE(D, f) = \frac{100}{n} \sum_{i=1}^{n} \left| \frac{y_i - f(x_i)}{y_i} \right| \]

---

Evaluation measures

Regression model $f: X \rightarrow Y$ where $Y$ is real valued

- The error is calculated on the data $D$, say
  
  \[ MSE(D, f) = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2 \]

- This is an estimate of the error for $f$ on the complete population

**Important question:**

- How close is our estimate to the true mean error?

**To answer the question we need to resort to statistics:**

- How confident we are the true error falls into interval around our estimate $\mu$ ?

**Answer:** with probability 0.95 the true error is in interval $[\mu^-, \mu^+]$
**Evaluation**

- **Central limit theorem:**
  Let random variables $X_1, X_2, \ldots, X_n$ form a random sample from a distribution with mean $\mu$ and variance $\sigma$, then if the sample $n$ is large, the distribution

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

- **Statistical significance test**

- **Statistical tests for the mean**
  - H0 (null hypothesis) $E[X] = \mu^0$
  - H1 (alternative hypothesis) $E[X] \neq \mu^0$

- **Basic idea:**
  we use the sample mean $\bar{X} = \frac{1}{n} \sum_{i=1}^{n} X_i$
  and check how probable it is that $E[X] = \mu^0$ holds

  If the probability that $\bar{X}$ comes from the normal distribution with mean $\mu^0$ is small – we reject the null hypothesis on that probability level
Statistical significance test

• Statistical tests for the mean
  – H0 (null hypothesis) \( E[X] = \mu^0 \)
  – H1 (alternative hypothesis) \( E[X] \neq \mu^0 \)

• Assume we know the standard deviation \( \sigma \) for the sample

\[
z = \frac{\bar{X} - \mu^0}{\sigma} \sqrt{n} \approx N(0,1) \quad \text{with} \quad P=0.95 \quad z \in [-1.96,1.96]
\]

Statistical significance test

• Statistical tests for the mean
  – H0 (null hypothesis) \( E[X] = \mu^0 \)

• Assume we know the standard deviation \( \sigma \)

\[
z = \frac{\bar{X} - \mu^0}{\sigma} \sqrt{n} \approx N(0,1) \quad \text{with} \quad P=0.95 \quad z \in [-1.96,1.96]
\]

• Z-test: If \( z \) is outside of the interval – reject the null hypothesis at significance level \( 1-P \) if \( P=0.95 \) it is 0.05
Statistical significance test

- **Statistical tests for the mean**
  - H0 (null hypothesis) \( E[X] = \mu^0 \)
- **Problem:** we do not know the standard deviation \( \sigma \)
- **Solution:**
  \[
  t = \frac{\bar{X} - \mu^0}{s} \sqrt{n} \approx t \text{ - distribution} \quad \text{(Student distribution)}
  \]
  \[
  s = \sqrt{\frac{\sum_{i=1}^{n} (X_i - \bar{X})^2}{n-1}}
  \]
  - Estimate of the standard deviation
- **T-test:** If \( t \) is outside of the tabulated interval reject the null hypothesis at the corresponding significance level

Confidence interval

- Assume we have calculated the average error \( \bar{X} = \frac{1}{n} \sum_{i=1}^{n} X_i \)
- There are many values of \( \mu^0 \) around it that are not rejected at some significance level (say 0.05)
- These values form a confidence interval around it

\[
\mu^- \quad \bar{X} \quad \mu^+ \quad \text{95% confidence interval}
\]

Confidence interval

- Significance level: 0.1

\[
\mu^- \quad \bar{X} \quad \mu^+ \quad \text{90% confidence interval}
\]
Statistical tests

The statistical tests lets us answer:

• The probability with which the true error falls into the interval around our estimate, say:

\[
MSE(D, f) = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2
\]

• Compare two models M1 and M2 and determine based on the error on the data entries the probability with which model M1 is different (or better) than M2

\[
MSE(D, f_1) = \frac{1}{n} \sum_{i=1}^{n} (y_i - f_1(x_i))^2 \quad MSE(D, f_2) = \frac{1}{n} \sum_{i=1}^{n} (y_i - f_2(x_i))^2
\]

Trick:

\[
MSE(D, f_1) - MSE(D, f_2) = \frac{1}{n} \sum_{i=1}^{n} (y_i - f_1(x_i))^2 - \frac{1}{n} \sum_{i=1}^{n} (y_i - f_2(x_i))^2
= \frac{1}{n} \sum_{i=1}^{n} (y_i - f_1(x_i))^2 - (y_i - f_2(x_i))^2
\]

Evaluation measures

Similarly evaluations measures can be defined for the classification tasks

Assume binary classification:

<table>
<thead>
<tr>
<th></th>
<th>Actual</th>
<th>Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Case</td>
<td>Control</td>
</tr>
<tr>
<td>Case</td>
<td>TP 0.3</td>
<td>FP 0.1</td>
</tr>
<tr>
<td>Control</td>
<td>FN 0.2</td>
<td>TN 0.4</td>
</tr>
</tbody>
</table>

Misclassification error:

\[
E = FP + FN
\]

Sensitivity:

\[
SN = \frac{TP}{TP + FN}
\]

Specificity:

\[
SP = \frac{TN}{TN + FP}
\]
Evaluation of models

- We started with a simple holdout method

**Problem:** the mean error results may be influenced by a lucky or an unlucky *training and testing* split especially for a small size D

**Solution:** try multiple train-test splits and average their results

---

Evaluation of models via random resampling

**Other more complex methods**

- Use multiple train/test sets
- Based on various random re-sampling schemes:
  - Random sub-sampling
  - Cross-validation
  - Bootstrap
Evaluation of models using random subsampling

- **Random sub-sampling**
  - Repeat a simple holdout method $k$ times

Evaluation of models using $k$-fold cross-validation

**Cross-validation ($k$-fold)**

- Divide data into $k$ disjoint groups, test on $k$-th group/train on the rest
- Typically 10-fold cross-validation
- Leave one out cross-validation ($k = \text{size of the data D}$)
Evaluation of models using bootstrap

**Bootstrap**

- The training set of size $N = \text{size of the data } D$
- Sampling with the replacement

```plaintext
Data

Generate the training set of size $N$ with replacement, the rest goes to the test set

Train → Test

Learning → Classify/Evaluate

Average Stats
```