Lecture 6:

- k-NN
- Cross-validation
- Regularization
Lazy vs eager learning

- **Eager learning** — generalizes training data before evaluation (e.g. Neural networks)
  - Fast prediction evaluation
  - Summarize training set (noise reduction)

- **Lazy learning** — wait a prediction query to generalize (e.g. k-NN)
  - Local approximation
  - Quick adaptation to variation of the training set
  - Require storage of the full training set
  - Slow evaluation
Instance based learning

- Type of lazy learning

- Store in memory the training set
- Compare a test sample to the samples memory
K-NEAREST NEIGHBORS (K-NN)
**k-NN**

- Simple
- Non-differentiable
- Lazy learning

**The main idea:**
- Find the $k$ closest samples (for instance with Euclidean distance)
- Assign the most frequent class occurring on those $k$ samples
1-NN: Nearest Neighbor

- No computation of the explicit decision boundary
- The decision boundary form a subset of the Voronoi diagram
- Decision boundaries are irregular
The number of neighbors influence

- The best $k$ is data dependent

- Larger values of $k$: robustness to noise but fuzzy boundaries

- Model selection (validation set) is the best heuristic to optimize $k$
Variants

- **Training:**
  - Very fast (basically non-existing)
  - Only input preprocessing (feature extraction and dimensionality reduction)

- **Testing** (k-NN can be used for classification and regression):
  - **Classification:**
    - Majority of votes of its $k$ nearest neighbors
  - **Regression:**
    - Average of its $k$ nearest neighbors.
Pros and cons

Pros:
- Easy to implement/understand
- No training
- Learn very complex decision boundaries
- No information loss (all samples are kept)

Cons:
- Require storage of all the data samples
- Slow at query time
- Bad performance if metric or feature vector is bad
Application tips

• When to use k-NN:
  • Lots of data is available
  • Small number of features

• What if the classes are not evenly represented?
  • In that case a more frequent class tend to dominate the prediction of the new example

• Weighting heuristics
UNDERFITTING AND OVERFITTING - RECAP
Under-/ and Overfitting

Model complexity
(e.g. degree of polynomial terms)

Training error (cost)
Test error (cost)

underfitting

overfitting

„just right“
Under- and Overfitting

- **Underfitting:**
  - Model is too simple
  - High training error, high test error

- **Overfitting:**
  - Model is too complex (often: too many parameters relative to number of training examples)
  - Low training error, high test error

- In between „just right“
  - Moderate training error
  - Lowest test error
How to deal with overfitting

• Use **model selection** to automatically select the right model complexity

• Use **regularization** to keep parameters small

• Collect more data
  (often not possible or inefficient)

• Manually throw out features which are unlikely to contribute
  (often hard to guess which ones, potentially throwing out the wrong ones)

• Pre-processing, change the feature vector or perform dimension reduction
  (endless effort, often not possible or inefficient)
Model selection: Training/Validation/Test set workflow

For example: Linear regression, Polynomial regression, Artificial Neural Network
Cross-validation

- **The goal:**
  Define a validation set to "pre-test" in the training phase.
  Use the full training set

- **Why to use it:**
  Instead of training error keep track of the predictive power

- **The trick:**
  Recycle the data by using different training/validation partitions
Model selection with Cross-validation

1. Compute averaged cross-validated error (CV) for each model

   \[ h^A \]
   \[ \begin{array}{cccc}
   \text{run 1} & \text{run 2} & \text{run 3} & \text{run 4} \\
   .6 & .5 & .7 & .6 \\
   \end{array} \]

   \[ h^B \]
   \[ \begin{array}{cccc}
   \text{run 1} & \text{run 2} & \text{run 3} & \text{run 4} \\
   .4 & .5 & .5 & .6 \\
   \end{array} \]

2. Choose the model with smallest CV

   \[ .6 \]
   \[ .5 \]
Cross-validation approaches

- Disadvantage of a single validation set:
  - Little training data - the function is poorly fitted
  - Little validation data - the true error is poorly estimated

- Tricks and warnings
  - Beware if the variance of the error over partitions is large
  - Train the best class over full data after selection
  - Use the same partitions for all hypothesis

- Common types of partitioning:
  - k-fold
  - 2-fold
  - Leave-one-out
  - Repeated random sub-sampling
K-fold cross-validation

- Useful when training dataset is small

- Steps:
  - Split the data into $k$ equal folds
  - Repeat $k$ times cross-validation process: each of the folds should be used once as a validation set and the rest as a training set
  - Calculate the mean and the variance of $k$ runs

- Disadvantage:
  - It requires $k$ runs of algorithm which means $k$ times as much computation
2-fold cross-validation

- The simplest approach, also called holdout method

- Idea:
  - Split randomly the whole training data into 2 equal folds ($k=2$)
  - Train on the first fold and validate on the second, and vice versa

- Advantage:
  - Both training and validation sets are fairly large
  - Each data point is used for both training and validation on each fold
Leave-one-out cross-validation

- This is a special case where $k$ equals the number of samples in the training set

- Idea:
  - Use a single sample as a validation set and all the rest as training set ($k$ times)

- Used in the case of really small training set
Leave-one-out cross-validation example

1st order

CV = 0.6

3rd order

CV = 1.5

5th order

CV = 6.0

7th order

CV = 15.6
Repeated random sub-sampling validation

- **Idea:**
  - Randomly split the dataset into training and validation sets $k$ times

- **Advantage:**
  - Choose independently how large each validation set is and how many trials you average over

- **Disadvantage:**
  - Validation subsets may overlap (some sample may never be selected)
REGULARIZATION
Bias-variance dilemma (tradeoff)

- Expected prediction error = Variance + Bias^2 + Noise

- The bias–variance dilemma (tradeoff) is the problem of simultaneously minimizing the bias and the variance of the model error

- **Variance** - how sensitive the model is to small changes in the training set
- **Bias** - how accurate a model is across different training sets

- High bias -> underfitting (the model is too simple)
- High variance -> overfitting (the model is too complex)

- To achieve good performance on data outside the training set a tradeoff must be made!
Under-/ and Overfitting

- Model complexity (e.g., degree of polynomial terms)
- Training error (cost)
- Test error (cost)

- Low variance
- High bias
- Underfitting

- High variance
- Low bias
- Overfitting

"Just right"
Polynomial regression under-/overfitting

underfitting
high bias
\[ h_\theta(x) = \theta_0 + \theta_1 \cdot x \]

„just right“
\[ h_\theta(x) = \theta_0 + \theta_1 \cdot x + \cdots + \theta_5 \cdot x^5 \]
\[ \theta_6 \approx 0 \quad \ldots \quad \theta_{15} \approx 0 \]

overfitting
high variance
\[ h_\theta(x) = \theta_0 + \theta_1 \cdot x + \cdots + \theta_5 \cdot x^5 + \theta_6 \cdot x^6 + \cdots + \theta_{15} \cdot x^{15} \]

\[ J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + 1000 \cdot \theta_6^2 + \cdots + 1000 \cdot \theta_{15}^2 \]

suppose we penalize \( \theta_6 \ldots \theta_{15} \)
Regularization

- Prefer simple models
- Exclude extreme models

How to do it:
- Instead of minimizing the original problem $J(\theta)$ minimize $J(\theta) + \lambda \|\theta\|^2$
- where $\|\theta\|$ is $L_2$ norm (Euclidean norm)

- Large $\lambda$ leads to underfitting (high bias)
- Low $\lambda$ to overfitting (high variance)
Regularized linear regression

• Regularized cost function:

\[ J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left( h_\theta \left( x^{(i)} \right) - y^{(i)} \right)^2 + \frac{\lambda}{m} \sum_j \theta_j^2 \]

\[ J(\theta) = \frac{1}{m} \| X\theta - y \|^2 + \frac{\lambda}{m} \| \theta \|^2 \]

• Analytical solution:

\[ \theta^* = \left( X^T X + \lambda I \right)^{-1} X^T y \]

• Gradient descent solution:

\[ \theta_j := \theta_j (1 - 2\eta \frac{\lambda}{m}) - 2\eta \cdot \frac{1}{m} \sum_{i=1}^{m} \left( h_\theta \left( x^{(i)} \right) - y^{(i)} \right) \cdot x_j^{(i)} \]
Regularization of NN

- How many hidden layers and how many neurons?
  - Fewer – risk of underfitting
  - More – risk of overfitting

- Reduce the parameter space:
  - Weight decay
  - Network structure (weight sharing)

- Keep track of predictive power:
  - Early stopping
Weight decay

- “Weight decay” is a $L_2$ norm regularization for Neural networks.
- The weights of a NN will be an additional term in an Error function:

$$E(w) = MSE(w) + \frac{\lambda}{2} ||w||^2$$
Sparse structure

- Weights: $32 \times 32 \times K_{\text{hidden}}$
- Weights: $8 \times 8 \times K_{\text{hidden}}$
- Different role between hidden units

Sparse = many weights set to null
Weight sharing

- Weights: $8 \times 8 \times K_{\text{hidden}}$
- $W_i = W_j = W_0$
- Weights: $8 \times 8$
- Spatial **invariance**
  Even positions were pixels are always 0 may learn to recognize some shapes
Early stopping

- A form of regularization based on the scheme of model selection

- Steps:
  - The weights are initialized to small values
  - Stop when the error on validation data increases
SUMMARY (QUESTIONS)
Some questions…

• Difference between lazy and eager learning?
• What is Instance based learning?
• Training and testing procedure for k-NN?
• How does the number of neighbors influence k-NN?
• When to use k-NN and what are pros/cons?

• What is overfitting and how to deal with it?
• What is validation set?
• What is cross-validation?
• Types of partitioning in cross-validation?

• What is the bias-variance tradeoff?
• What is regularization and how is it used?
• What are regularization methods for NN?