COMP 551 – Applied Machine Learning
Lecture 7: Instance-based learning

Instructor: Herke van Hoof (herke.vanhoof@mcgill.ca)

Slides mostly by: Joelle Pineau (jpineau@cs.mcgill.ca)

Class web page: www.cs.mcgill.ca/~vanho2/comp551

Unless otherwise noted, all material posted for this course are copyright of the instructor, and cannot be reused or reposted without the instructor’s written permission.
Main types of machine learning problems

• Supervised learning
  – Classification
  – Regression

• Unsupervised learning

• Reinforcement learning
Aim of supervised learning

- Minimize the true error function over all possible data points
  \[
  \int_{\mathbb{R}^d} \int_{-\infty}^{\infty} p(x, y) L(f(x), y) \, dy \, dx
  \]
- Re-write, for e.g. squared loss:
  \[
  \hat{f} = \min_{f \in F} \mathbb{E}_{x, y} (f(x) - y)^2
  \]
- If we assume \( f(x) \) is linear in \( x \), we can find the best \( x \) by taking the derivative and setting to 0
  \[
  \hat{w} = \min_w \mathbb{E}_{x, y} (w^T x - y)^2 = (\mathbb{E}_x xx^T)^{-1} \mathbb{E}_x y x y
  \]
- Replacing the expectation by observed average yields the linear regression solution - no surprise so far…
Parametric supervised learning

- Linear regression example of **parametric** supervised learning.
  Input: dataset of labeled examples.

- From this, **learn a parameter vector of a fixed size** such that some error measure based on the training data is minimized.

- Main goal is to summarize the data using the parameters.
  - Parametric methods are typically **global** = one set of parameters for the entire data space.
  - ‘Shape’ is specified in advance – not very flexible!
Aim of supervised learning

- Minimize the true error function over all possible data points

\[
\int_{\mathbb{R}^d} \int_{-\infty}^{\infty} p(x, y) L(f(x), y) dy dx
\]

- Re-write, for e.g. squared loss:

\[
\hat{f} = \min_{f} \mathbb{E}_{x,y} (f(x) - y)^2 = \min_{f} \mathbb{E}_x \mathbb{E}_y |x [(f(x) - y)^2 |x]
\]

- Optimize \( f(x) \) separately for each \( x \)

\[
\hat{f}(x) = \min_{c} \mathbb{E}_y |x [(c - y)^2 |x]
\]
Aim of supervised learning

- Minimize the true error function over all possible data points
  \[
  \int_{\mathbb{R}^d} \int_{-\infty}^{\infty} p(x, y) L(f(x), y) dy dx
  \]

- Re-write, for e.g. squared loss:
  \[
  \hat{f} = \min_{f} \mathbb{E}_{x,y} (f(x) - y)^2 = \min_{f} \mathbb{E}_x \mathbb{E}_y |x| [(f(x) - y)^2 |x] 
  \]

- Optimize \( f(x) \) separately for each \( x \)
  \[
  \hat{f}(x) = \min_c \mathbb{E}_{y|x} [ (c - y)^2 |x] 
  \]
  \[
  \nabla_c \mathbb{E}_{y|x} [ (c - y)^2 |x] = \mathbb{E}_{y|x} \left[ \nabla_c c^2 - 2cy + y^2 \right] = 0 
  \]
  \[
  \mathbb{E}_{y|x} [2c] - \mathbb{E}_{y|x} [2y] = 0 
  \]
  \[
  c = \mathbb{E}_{y|x} [y] 
  \]
Aim of supervised learning

$$\hat{f}(x) = \mathbb{E}_{y|x}[y]$$

- Minimize error by again substituting average for expectation?
- How many data points will have the same value for \( x \)?
- Idea: look at a small neighborhood around \( x \)

$$\hat{f}(x) = \text{Average}[y_i | x_i \in \text{neighbours}(x)]$$
Instance based learning methods

- Key idea: just store all training examples $<x_i, y_i>$. 

- When a query is made, **locally** compute the value $y$ of new instance based on the values of the most similar points.
Instance based learning methods

• Key idea: just store all training examples \(< x_i, y_i >\).

• When a query is made, **locally** compute the value \(y\) of new instance based on the values of the most similar points.

• The regressor / classifier can now **not** be represented by a fixed-sized vector: representation depends on dataset.
Instance based learning methods

• Key idea: just store all training examples $< x_i, y_i >$.

• When a query is made, **locally** compute the value $y$ of new instance based on the values of the most similar points.

• The regressor / classifier can now **not** be represented by a fixed-sized vector: *representation depends on dataset*

• Different algorithms for computing the value of the new point based on the existing values
Simple idea: Connect the dots!

- Use single most similar data point

**What kind of distance metric?**

- Euclidean distance
- Maximum/minimum difference along any axis
- Weighted Euclidean distance: $d(x, x_0) = \sum_{j=1}^{n} u_j (x_j - x_0_j)^2$

- Pn arbitrary distance or similarity function $d_S$ specific for the application $a$ works best if you have one

*Wisconsin data set* Wisconsin Data Set Classification Lecture 1 September 1993
Simple idea: Connect the dots!

- Use single most similar data point

Wisconsin data set, classification
Simple idea: Connect the dots!

- Use single most similar data point
Simple idea: Connect the dots!

- Use single most similar data point

Wisconsin data set, regression
One-nearest neighbor

- **Given**: Training data $X$, distance metric $d$ on $X$.

- **Learning**: Nothing to do! (Just store the data).
One-nearest neighbor

- **Given**: Training data $X$, distance metric $d$ on $X$.

- **Learning**: Nothing to do! (Just store the data).

- **Prediction**: For $x \in X$
  
  Find nearest training sample $x_i$.

  $i^* = \arg\min_i d(x_i, x)$

  Predict $y = y_{i^*}$
What does the approximator look like?

- What do you think the decision boundary looks like?
What does the approximator look like?

- Nearest-neighbor does not explicitly compute decision boundaries.
- But the effective decision boundaries are a subset of the Voronoi diagram for the training data.
- Each decision boundary is a line segment that is equidistant between two points of opposite classes.
What does the approximator look like?

- Example
One-nearest neighbor

- **Given**: Training data $X$, distance metric $d$ on $X$.

- **Learning**: Nothing to do! (Just store the data).

- **Prediction**: For $x \in X$

  Find nearest training sample $x_i$.

  $$i^* = \underset{i}{\text{argmin}} \ d(x_i, x)$$

  Predict $y = y_{i^*}$
What kind of distance metric?
What kind of distance metric?

• Euclidean distance.

• Weighted Euclidean distance (with weights based on domain knowledge):
  \[ d(x, x') = \sum_{j=1:m} w_j (x_j - x_j')^2 \]
What kind of distance metric?

- Euclidean distance.

- Weighted Euclidean distance (with weights based on domain knowledge): 
  \[ d(x, x') = \sum_{j=1:m} w_j (x_j - x_j')^2 \]

- Maximum / minimum difference along any axis.

- An arbitrary distance or similarity function \( d \), specific for the application at hand (works best, if you have one.)
Choice of distance metric is important!

Left: both attributes weighted equally; Right: second attributes weighted more
Distance metric tricks

• You may need to do feature preprocessing:
  – Scale the input dimensions (or normalize them).
  – Remove noisy and irrelevant inputs.
  – Determine weights for attributes based on cross-validation (or information-theoretic methods).
Distance metric tricks

• You may need to do feature preprocessing:
  – Scale the input dimensions (or normalize them).
  – Remove noisy and irrelevant inputs.
  – Determine weights for attributes based on cross-validation (or information-theoretic methods).

• Distance metric is often domain-specific.
  – E.g. string edit distance in bioinformatics.
  – E.g. trajectory distance in time series models for walking data.

• Distance can be learned sometimes.
**k-nearest neighbor (kNN)**

- In case of noise, a **single** bad label can cause a patch to be misclassified.
- Safer to look at more than one close point?
**k-nearest neighbor (kNN)**

- **Given:** Training data $X$, distance metric $d$ on $X$.

- **Learning:** Nothing to do! (Just store the data).

- **Prediction:**
  - For $x \in X$, find the $k$ nearest training samples to $x$.
  - Let their indices be $i_1, i_2, \ldots, i_k$.
  - Predict: $y = \text{mean/median of } \{y_{i_1}, y_{i_2}, \ldots, y_{i_k}\}$ for regression
    
    $y = \text{majority of } \{y_{i_1}, y_{i_2}, \ldots, y_{i_k}\}$ for classification, or
    
    empirical probability of each class.
Classification, 2-nearest neighbor

- A line chart illustrates the relationship between tumor size (mm?) and non-recurring (0) vs. recurring (1) events. The y-axis represents the probability of non-recurring or recurring events, while the x-axis shows tumor size in millimeters.
Classification, 3-nearest neighbor

3-nearest neighbor, mean

non-recurring (0) / recurring (1)

tumor size (mm?)
Classification, 10-nearest neighbor

![Graph showing tumor size vs. non-recurring (0) / recurring (1)]
Classification, 20-nearest neighbor
Regression, 2-nearest neighbor

![Graph showing the relationship between nucleus size and time to recurrence.](image)

- The x-axis represents nucleus size, ranging from 10 to 28.
- The y-axis represents time to recurrence, ranging from 0 to 80.
- The graph uses blue 'x' markers for the data points.
- The red line connects the data points, indicating a trend in the relationship.

This visual representation helps in understanding the correlation between nucleus size and time to recurrence.
Regression, 5-nearest neighbor

![Graph showing the relationship between nucleus size and time to recurrence]
Regression, 10-nearest neighbor
What is the best regressor?

Regression with nearest neighbor

- K=2
- K=5
- K=10
Bias-variance trade-off

- What happens if $k$ is low?

- What happens if $k$ is high?
Bias-variance trade-off

• What happens if $k$ is low?
  Very non-linear functions can be approximated, but we also capture the noise in the data. Bias is low, variance is high.

• What happens if $k$ is high?
  The output is much smoother, less sensitive to data variation. High bias, low variance.

• A validation set can be used to pick the best $k$. 
Limitations of $k$-nearest neighbor (kNN)

- A lot of discontinuities!
- Sensitive to small variations in the input data.
- Can we fix this but still keep it (fairly) local?
**k-nearest neighbor (kNN)**

- **Given:** Training data $X$, distance metric $d$ on $X$.

- **Learning:** Nothing to do! (Just store the data).

- **Prediction:**
  - For $x \in X$, find the $k$ nearest training samples to $x$.
  - Let their indices be $i_1, i_2, \ldots, i_k$.
  - Predict: $y = \text{mean/median of } \{y_{i_1}, y_{i_2}, \ldots, y_{i_k}\}$ for regression
    
    $y = \text{majority of } \{y_{i_1}, y_{i_2}, \ldots, y_{i_k}\}$ for classification, or
    
    empirical probability of each class.
Distance-weighted (kernel-based) NN

• **Given:** Training data $X$, distance metric $d$ on $X$, weighting function $w : R \rightarrow R$.

• **Learning:** Nothing to do! (Just store the data).

• **Prediction:**
  
  – Given input $x$.
  
  – For each $x_i$ compute $w_i = w(d(x_i, x))$.
  
  – Predict: $y = \sum_i w_i y_i / \sum_i w_i$. 
Distance-weighted (kernel-based) NN

- **Given:** Training data $X$, distance metric $d$ on $X$, weighting function $w : R \rightarrow R$.

- **Learning:** Nothing to do! (Just store the data).

- **Prediction:**
  - Given input $x$.
  - For each $x_i$ compute $w_i = w(d(x_i, x))$.
  - Predict: $y = \sum_i w_i y_i / \sum_i w_i$.

- How should we weigh the distances?
Some weighting functions

\[
\begin{align*}
\frac{1}{d(x_i, x)} & \quad \frac{1}{d(x_i, x)^2} & \quad \frac{1}{c + d(x_i, x)^2} & \quad e^{-\frac{d(x_i, x)^2}{\sigma^2}}
\end{align*}
\]

![Graphs of the weighting functions](image-url)
Gaussian weighting, small $\sigma$

Gaussian-weighted nearest neighbor with $\sigma=0.25$

- Non-recurring (0) / recurring (1)
- Tumor size (mm?)
Gaussian weighting, medium $\sigma$
Gaussian weighting, large $\sigma$

All examples get to vote! Curve is smoother, but perhaps too smooth?
Scaling up

• kNN in high-dimensional feature spaces?

• kNN with larger number of datapoints?
Scaling up

- **kNN in high-dimensional feature spaces?**
  - In high dim spaces, the distance between points appears similar.
  - A few points (“hubs”) show up repeatedly in the top kNN [Radovanovic et al., 2009].

- **kNN with larger number of datapoints?**
Scaling up

• kNN in high-dimensional feature spaces?
  – In high dim spaces, the distance between points appears similar.
  – A few points (“hubs”) show up repeatedly in the top kNN [Radovanovic et al., 2009].

• kNN with larger number of datapoints?
  – Can be implemented efficiently, $O(\log n)$ at retrieval time, if we use smart data structures:
    • Condensation of the dataset (Use prototypes)
    • Hash tables in which the hashing function is based on the distance metric.
    • KD-trees (Tutorial: http://www.autonlab.org/autonweb/14665)
Instance based learning

- Instance-based learning refers to techniques where previous samples are used directly to make predictions.
- What makes instance based methods different?
  - Model is typically non-parametric (no fixed parameter vector).
  - Algorithms are typically lazy.
Lazy vs eager learning

• **Lazy learning**: Wait for query before generalization.
  – E.g. Nearest neighbour.

• **Eager learning**: Generalize before seeing query.

• Which is faster?
  – Training time?
  – Query answering time?
Pros and cons of lazy and eager learning

• Eager learners create global approximation.
• Lazy learners create many local approximations.
• If they use the same hypothesis space, a lazy learner can represent more complex functions (e.g., consider $H = \text{constant}$).
Pros and cons of lazy and eager learning

- Eager learners create global approximation.
- Lazy learners create many local approximations.
- If they use the same hypothesis space, a lazy learner can represent more complex functions (e.g., consider $H = \text{constant}$).

- Lazy learning has much faster training time.
- Lazy learner typically has slower query answering time (depends on number of instances and number of features) and requires more memory (must store all the data).
- Eager learner does the work off-line.
Non-parametric method

• Representation for parametric method is specified in advance
  – Fixed size representation

• Representation for non-parametric methods depends on dataset
  – Size of representation typically linear in # of examples
Pros and cons of non-parametric method

• Representation for parametric method is specified in advance
  – Good if a good representation is known in advance
  – Can easily leverage knowledge about structure

• Representation for non-parametric methods depends on dataset
  – High resolution where much data available / decisions are complex
  – If little is known data distribution (no good representation known)
  – Still requires a good distance metric

• Non-parametric methods often require complex computations
• Non-parametric methods typically larger storage requirement
Lazy / eager and non-parametric

• Lazy / eager: Generalization before or after seeing query?
• Parametric or not: fixed # of parameters or determined by data?

• Usually, parametric methods are also eager
• Often, non-parametric are also lazy
  – But there are exceptions!
When to use instance-based learning

- Instances map to points in $\mathbb{R}^n$. Or else a given distance metric.

- Not too many attributes per instance (e.g. <20), otherwise all points look at a similar distance, and noise becomes a big issue.

- Not too many irrelevant attributes: easily fooled! (for most distance metrics.)

- Structure of model not known in advance

- Uneven spread of data: Provides variable resolution approximation (based on density of points).
Application

Hays & Efros, Scene Completion Using Millions of Photographs, CACM, 2008.

Locally weighted regression
What you should know

• Difference between **eager** vs **lazy** learning.

• Key idea of **non-parametric** learning.

• The **k-nearest neighbor** algorithm for classification and regression, and its properties.

• The distance-weighted NN algorithm

• How NN and linear regression relate to minimizing true error