Lecture 16: K-nearest-neighbor Classifier / Bias-Variance Tradeoff

Dr. Yanjun Qi

University of Virginia
Department of Computer Science

10/27/15

Announcements: Rough Plan

• HW3: due on Nov. 8th midnight
• Midphase Project Report: due on Nov. 4th

• Late Midterm:
  – Open note / Open lecture
  – Nov. 18th / conflicts with many students’ conference trips
  – Nov. 23rd ??? / conflicts ???

• HW4:
  – 20 samples questions for the preparation for exam
  – Due depending on Late-Midterm Date; if 23rd, due on 20th
Where are we? ➔

Five major sections of this course

- Regression (supervised)
- Classification (supervised)
- Unsupervised models
- Learning theory
- Graphical models

Where are we? ➔

Three major sections for classification

- We can divide the large variety of classification approaches into **roughly three major types**

1. **Discriminative**
   - directly estimate a decision rule/boundary
   - e.g., *logistic regression*, support vector machine, decisionTree

2. **Generative:**
   - build a generative statistical model
   - e.g., *naïve bayes classifier*, Bayesian networks

3. **Instance based classifiers**
   - Use observation directly (no models)
   - e.g. *K nearest neighbors*
Today:

- K-nearest neighbor
- Model Selection / Bias Variance Tradeoff

Nearest neighbor classifiers

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it’s probably a duck

![Diagram of nearest neighbor classification process]

- Training samples
- Compute distance
- Choose k of the “nearest” samples
- Test sample
Nearest neighbor classifiers

Requires three inputs:
1. The set of stored training samples
2. Distance metric to compute distance between samples
3. The value of $k$, i.e., the number of nearest neighbors to retrieve

To classify unknown sample:
1. Compute distance to other training records
2. Identify $k$ nearest neighbors
3. Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)
Definition of nearest neighbor

(a) 1-nearest neighbor  (b) 2-nearest neighbor  (c) 3-nearest neighbor

$k$-nearest neighbors of a sample $x$ are datapoints that have the $k$ smallest distances to $x$
Nearest neighbor classification

- Compute distance between two points:
  - For instance, Euclidean distance
    \[ d(x, y) = \sqrt{\sum (x_i - y_i)^2} \]
- Options for determining the class from nearest neighbor list
  - Take majority vote of class labels among the \(k\)-nearest neighbors
  - Weight the votes according to distance
    - example: weight factor \(w = 1 / d^2\)

Choosing the value of \(k\):
- If \(k\) is too small, sensitive to noise points
- If \(k\) is too large, neighborhood may include points from other classes
Nearest neighbor classification

- Scaling issues
  - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
  - Example:
    - height of a person may vary from 1.5 m to 1.8 m
    - weight of a person may vary from 90 lb to 300 lb
    - income of a person may vary from $10K to $1M

Problem with Euclidean measure:
- High dimensional data
  - curse of dimensionality
- Can produce counter-intuitive results

\[
\begin{array}{ccccccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}
\text{ vs }
\begin{array}{cccccccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
\]

\[d = 1.4142\]

\[d = 1.4142\]

One solution: normalize the vectors to unit length
Nearest neighbor classification

- *k*-Nearest neighbor classifier is a **lazy** learner
  - Does not build model explicitly.
  - Classifying unknown samples is relatively expensive.

- *k*-Nearest neighbor classifier is a **local** model, vs. global model of linear classifiers.

---

K-Nearest Neighbor

<table>
<thead>
<tr>
<th>Task</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Score Function</td>
</tr>
<tr>
<td></td>
<td>Search/Optimization</td>
</tr>
</tbody>
</table>

classification

- Local Smoothness
  - NA
  - NA

Training Samples

Models, Parameters

KNN: num_train/all train samples

Testing & SVM: num support vectors, points

Dr. Yanjun Qi / UVA CS 6316 / f15

10/27/15
Decision boundaries in global vs. local models

- **Global**
  - Linear regression
  - Stable
  - Can be inaccurate

- **Local**
  - 15-nearest neighbor
  - 1-nearest neighbor
  - Accurate
  - Unstable

What ultimately matters: **GENERALIZATION**

K-Nearest-Neighbours for Classification (Extra)

- K acts as a smoother
- For $N \to \infty$, the error rate of the 1-nearest-neighbour classifier is never more than twice the optimal error (obtained from the true conditional class distributions).
KNN METHODS IN HIGH DIMENSIONS (Extra)

- In high dimensions, all sample points are close to the edge of the sample
- \( N \) data points uniformly distributed in a \( p \)-dimensional unit ball centered at the origin
- Median distance from the closest point to the origin

\[
d(p, N) = \left( 1 - \frac{1}{2N} \right)^{1/p}
\]

- \( d(10, 500) = 0.52 \)
  - More than half the way to the boundary (unit ball’s boundary edge is 1 distance to the origin)

Vs. Locally weighted regression

- aka locally weighted regression, locally linear regression, LOESS, …

\[
K_\lambda(x_i, x_0)
\]

\[
\text{linear_func}(x) \rightarrow y
\]

\[\Rightarrow\]

could represent only the neighbor region of \( x_0 \)

Use RBF function to pick out/emphasize the neighbor region of \( x_0 \)

Figure 2: In locally weighted regression, points are weighted by proximity to the current \( x \) in question using a kernel. A regression is then computed using the weighted points.
Vs. Locally weighted regression

- Separate weighted least squares at each target point $x_0$:

$$\min_{\alpha(x_0), \beta(x_0)} \sum_{i=1}^{N} K_{\lambda}(x_i, x_0)[y_i - \alpha(x_0) - \beta(x_0)x_i]^2$$

$$\hat{f}(x_0) = \hat{\alpha}(x_0) + \hat{\beta}(x_0)x_0$$

$$K_{\tau}(x_i, x_0) = \exp\left(-\frac{(x_i - x_0)^2}{2\tau^2}\right)$$

Today:

- K-nearest neighbor
- Model Selection / Bias Variance Tradeoff
- EPE
- Decomposition of MSE
- Bias-Variance tradeoff
- High bias? High variance? How to respond?
e.g. Training Error from KNN, Lesson Learned

- When \( k = 1 \),
- No misclassifications (on training): **Overtraining**

- Minimizing training error is not always good (e.g., 1-NN)

---

**Statistical Decision Theory**

- Random input vector: \( X \)
- Random output variable: \( Y \)
- Joint distribution: \( \Pr(X, Y) \)
- Loss function \( L(Y, f(X)) \)

- Expected prediction error (EPE):
  
  \[
  \text{EPE}(f) = E(L(Y, f(X))) = \int L(y, f(x)) \Pr(dx, dy)
  \]
  
  e.g. \( \int (y - f(x))^2 \Pr(dx, dy) \)

  e.g. Squared error loss (also called L2 loss)
Expected prediction error (EPE)

\[ \text{EPE}(f) = \mathbb{E}(L(Y, f(X))) = \int L(y, f(x)) \Pr(dx, dy) \]

- For L2 loss: e.g., \( \int (y - f(x))^2 \Pr(dx, dy) \)
  under L2 loss, best estimator for EPE (Theoretically) is:
  \[ \hat{f}(x) = \mathbb{E}(Y | X = x) \]

  e.g. KNN

  NN methods are the direct implementation (approximation)

- For 0-1 loss: \( L(k, \emptyset) = 1 - d_{kl} \)
  \[ \hat{f}(X) = C_k \text{ if } \Pr(C_k | X = x) = \max_{g \in C} \Pr(g | X = x) \]

Bayes Classifier

LR VS. KNN FOR MINIMIZING EPE

- We know under L2 loss, best estimator for EPE (Theoretically) is:
  \[ f(x) = \mathbb{E}(Y | X = x) \]

- Two simple approaches using different approximations:
  - Least squares assumes that \( f(x) \) is well approximated by a globally linear function
  - Nearest neighbors assumes that \( f(x) \) is well approximated by a locally constant function.
Review: WHEN EPE USES DIFFERENT LOSS

<table>
<thead>
<tr>
<th>Loss Function</th>
<th>Estimator ( \hat{f}(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L_2 )</td>
<td>( \hat{f}(x) = E[Y</td>
</tr>
<tr>
<td>( L_1 )</td>
<td>( \hat{f}(x) = \text{median}(Y</td>
</tr>
<tr>
<td>0-1</td>
<td>( \hat{f}(x) = \arg \max_{Y} P(Y</td>
</tr>
</tbody>
</table>

Today:

- K-nearest neighbor
- Model Selection / Bias Variance Tradeoff
- EPE
- Decomposition of MSE
- Bias-Variance tradeoff
- High bias? High variance? How to respond?
Decomposition of EPE

- When additive error model:
  
  - Notations
  
    \[ Y = f(X) + \epsilon, \; \epsilon \sim (0, \sigma^2) \]

    - Output random variable: \( Y \)
    - Prediction function: \( f \)
    - Prediction estimator: \( \hat{f} \)

  \[
  \text{EPE}(x_0) = E[(Y - \hat{f})^2|X = x_0] \\
  = E[(Y - f) + (f - \hat{f})^2|X = x_0] \\
  = E[(Y - f)^2|X = x_0] + E[(f - \hat{f})^2|X = x_0] \\
  = \sigma^2 + \text{Var}(f) + \text{Bias}^2(f)
  \]

  Irreducible / Bayes error

Bias-Variance Trade-off for EPE:

\[
\text{EPE} (x_0) = \text{noise}^2 + \text{bias}^2 + \text{variance}
\]

- Unavoidable error
- Error due to incorrect assumptions
- Error due to variance of training samples
BIAS AND VARIANCE TRADE-OFF for MSE (more general setting !!!):

- **Bias** $E[(\hat{\theta} - \theta)^2]$
  - measures accuracy or quality of the estimator
  - low bias implies on average we will accurately estimate true parameter or function from training data

- **Variance** $E[(\bar{\theta} - \theta)^2]$
  - Measures precision or specificity of the estimator
  - Low variance implies the estimator does not change much as the training set varies

$\theta$ could be $w$ for LR, $f$ for EPE

---

**BIAS AND VARIANCE TRADE-OFF for MSE of parameter estimation:**

In EPE case, $E[(f - \hat{f})^2 | X = X_0] = MSE$

$$MSE(\hat{\theta}) = E[(\hat{\theta} - \theta)^2]$$

$$= E[((\hat{\theta} - \theta) + (\bar{\theta} - \theta))^2]$$

$$= E[(\hat{\theta} - \bar{\theta})^2] + E[(\bar{\theta} - \theta)^2] + 2E[(\hat{\theta} - \bar{\theta})(\bar{\theta} - \theta)]$$

$$= Var(\hat{\theta}) + Bias^2(\hat{\theta}) + 0$$

Error due to variance of training samples

Error due to incorrect assumptions

$$MSE(\hat{\theta}) = E[(\hat{\theta} - \theta)^2] = Bias^2(\hat{\theta}) + Var(\hat{\theta})$$

10/27/15
e.g., BIAS AND VARIANCE IN KNN (Extra)

- **Prediction**
  \[
  \hat{f}_k(x_0) = \frac{1}{k} \sum_{l=1}^{k} f(x_l)
  \]

- **Bias**
  \[
  \text{Bias}^2(\hat{f}_k(x_0)) = E[ f(x_0) - \hat{f}_k(x_0) ]^2 = E[ f(x_0) - \frac{1}{k} \sum_{l=1}^{k} f(x_l) ]^2
  \]

- **Variance**
  \[
  \text{Var}(\hat{f}_k(x_0)) = \frac{\sigma^2}{k}
  \]

- When under data model:
  \[
  Y = f(X) + \epsilon, \ \epsilon \sim (0, \sigma^2)
  \]

---

**Today:**

- ✔ K-nearest neighbor
- ✔ Model Selection / Bias Variance Tradeoff
  - ✔ EPE
  - ✔ Decomposition of MSE
- ✔ Bias-Variance tradeoff
  - ✔ High bias ? High variance ? How to respond ?
Bias-Variance Tradeoff / Model Selection

Which is the best?

Highest Bias
Lowest variance
Model complexity = low

Medium Bias
Medium Variance
Model complexity = medium

Smallest Bias
Highest variance
Model complexity = high

Why not choose the method with the best fit to the data?

How well are you going to predict future data?
Bias-Variance Trade-off

- Models with too few parameters are inaccurate because of a large bias (not enough flexibility).
- Models with too many parameters are inaccurate because of a large variance (too much sensitivity to the sample randomness).

Training vs Test Error

- Training error can always be reduced when increasing model complexity,
- But risks over-fitting and generalize poorly.
Regression: Complexity versus Goodness of Fit

What ultimately matters: **GENERALIZATION**

Classification, Decision boundaries in global vs. local models

What ultimately matters: **GENERALIZATION**
Model “bias” & Model “variance”

• Middle RED:
  – TRUE function

• Error due to bias:
  – How far off in general from the middle red

• Error due to variance:
  – How wildly the blue points spread

need to make assumptions that are able to generalize

• Components of generalization error
  – **Bias**: how much the average model over all training sets differ from the true model?
    • Error due to inaccurate assumptions/simplifications made by the model
  – **Variance**: how much models estimated from different training sets differ from each other

• **Underfitting**: model is too “simple” to represent all the relevant class characteristics
  – High bias and low variance
  – High training error and high test error

• **Overfitting**: model is too “complex” and fits irrelevant characteristics (noise) in the data
  – Low bias and high variance
  – Low training error and high test error
MODEL COMPLEXITY CONTROL,
EXAMPLES (Extra)

- Regularization (Bayesian)
  \[
  PRSS(f; \lambda) = \sum_{i=1}^{N} (y_i - f(x_i))^2 + \lambda \int [f''(x)]^2 dx
  \]

- Kernel methods and local regression
  \[
  RSS(f_{\theta}; x_0) = \sum_{i=1}^{N} K(\lambda, x_i, x_0)(y_i - f_{\theta}(x_i))^2
  \]

- Basis functions
  \[
  f_{\theta} = \sum_{m=1}^{m} \theta_m h_m(x)
  \]

Today:

- K-nearest neighbor
- Model Selection / Bias Variance Tradeoff
  - EPE
  - Decomposition of MSE
  - Bias-Variance tradeoff
- High bias ? High variance ? How to respond ?
High variance

Typical learning curve for high variance:

- Test error still decreasing as \( m \) increases. Suggests larger training set will help.
- Large gap between training and test error.
- Low training error and high test error

Could also use CrossV Error

How to reduce variance?

- Choose a simpler classifier
- Regularize the parameters
- Get more training data
- Try smaller set of features
High bias

Typical learning curve for high bias:

- Even training error is unacceptably high.
- Small gap between training and test error.

High training error and high test error

How to reduce Bias?

- E.g.
  - Get additional features
  - Try adding basis expansions, e.g. polynomial
  - Try more complex learner
For instance, if trying to solve “spam detection” using (Extra)

L2 - logistic regression, implemented with gradient descent.

Fixes to try:  **If performance is not as desired**
- Try getting more training examples.
- Try a smaller set of features.
- Try a larger set of features.
- Try email header features.
- Run gradient descent for more iterations.
- Try Newton’s method.
- Use a different value for $\lambda$.
- Try using an SVM.

Model Selection and Assessment

- **Model Selection**
  - Estimating performances of different models to choose the best one

- **Model Assessment**
  - Having chosen a model, estimating the *prediction error* on new data
Model Selection and Assessment (Extra)

- Data Rich Scenario: Split the dataset
  - Train | Validation | Test
  - Model Selection
  - Model assessment

- Insufficient data to split into 3 parts
  - Approximate validation step analytically
    - AIC, BIC, MDL, SRM
  - Efficient reuse of samples
    - Cross validation, bootstrap

Today Recap:

- K-nearest neighbor
- Model Selection / Bias Variance Tradeoff
  - EPE
  - Decomposition of MSE
  - Bias-Variance tradeoff
  - High bias? High variance? How to respond?
References

- Prof. Tan, Steinbach, Kumar’s “Introduction to Data Mining” slide
- Prof. Andrew Moore’s slides
- Prof. Eric Xing’s slides