K-Nearest Neighbors

Jia-Bin Huang
Virginia Tech
Administrative

• Check out review materials
  • Probability
  • Linear algebra
  • Python and NumPy

• Start your HW 0
  • On your Local machine: Install Anaconda, Jupiter notebook
  • On the cloud: https://colab.research.google.com

• Sign up Piazza discussion forum
Enrollment

• Maximum allowable capacity reached.
Machine learning reading & study group

• **Reading Group**  
  Tuesday 11 AM - 12:00 PM  
  Location: Whittmore Hall 457B  
  • Research paper reading: machine learning, computer vision

• **Study Group**  
  Thursday 11 AM - 12:00 PM  
  Location: Whittmore Hall 457B  
  • Video lecture: machine learning

All are welcome.

More info: [https://github.com/vt-vl-lab/reading_group](https://github.com/vt-vl-lab/reading_group)
# Recap: Machine learning algorithms

<table>
<thead>
<tr>
<th></th>
<th>Supervised Learning</th>
<th>Unsupervised Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Discrete</strong></td>
<td>Classification</td>
<td>Clustering</td>
</tr>
<tr>
<td><strong>Continuous</strong></td>
<td>Regression</td>
<td>Dimensionality reduction</td>
</tr>
</tbody>
</table>
Today’s plan

• Supervised learning
  • Setup
  • Basic concepts

• K-Nearest Neighbor (kNN)
  • Distance metric
  • Pros/Cons of nearest neighbor

• Validation, cross-validation, hyperparameter tuning
Supervised learning

• **Input:** $x$ (Images, texts, emails)

• **Output:** $y$ (e.g., spam or non-spams)

• **Data:** $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(N)}, y^{(N)})$ (Labeled dataset)

• (Unknown) **Target function:** $f: x \rightarrow y$ (“True” mapping)

• **Model/hypothesis:** $h: x \rightarrow y$ (Learned model)

• Learning = search in hypothesis space

Slide credit: Dhruv Batra
Training set → Learning Algorithm → Hypothesis $h$ → $y$
Regression

Training set

Learning Algorithm

\[ h \]

\[ x \rightarrow h \rightarrow y \]

Size of house  Hypothesis  Estimated price
Classification

Training set

Learning Algorithm

Unseen image $x$ $\rightarrow$ Hypothesis $h$ $\rightarrow$ Predicted object class $y$ ‘Mug’

Image credit: CS231n @ Stanford
Procedural view of supervised learning

• **Training Stage:**
  - Raw data $\rightarrow x$ (Feature Extraction)
  - Training data $\{(x, y)\} \rightarrow h$ (Learning)

• **Testing Stage**
  - Raw data $\rightarrow x$ (Feature Extraction)
  - Test data $x \rightarrow h(x)$ (Apply function, evaluate error)
Basic steps of supervised learning

• **Set up** a supervised learning problem
• **Data collection:** Collect training data with the “right” answer.
• **Representation:** Choose how to represent the data.
• **Modeling:** Choose a hypothesis class: \( H = \{ h: X \rightarrow Y \} \)
• **Learning/estimation:** Find best hypothesis in the model class.
• **Model selection:** Try different models. Picks the best one.
  (More on this later)
• If happy stop, else refine one or more of the above
Nearest neighbor classifier

• **Training data**

\[(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(N)}, y^{(N)})\]

• **Learning**

Do nothing.

• **Testing**

\[h(x) = y^{(k)}, \text{ where } k = \arg\min_i D(x, x^{(i)})\]
Face recognition
Face recognition – surveillance application
Music identification

https://www.youtube.com/watch?v=TKNNOMddkNc
Album recognition (Instance recognition)

http://record-player.glitch.me/auth
Scene Completion

Original

Input

Scene Matches

Output

[Hayes & Efros, SIGGRAPH07]
... 200 total

[Hayes & Efros, SIGGRAPH07]
Context Matching

[Hayes & Efros, SIGGRAPH07]
Graph cut + Poisson blending

[Hayes & Efros, SIGGRAPH07]
Synonyms

• Nearest Neighbors

• k-Nearest Neighbors

• Member of following families:
  • Instance-based Learning
  • Memory-based Learning
  • Exemplar methods
  • Non-parametric methods
Instance/Memory-based Learning

1. A distance metric

2. How many nearby neighbors to look at?

3. A weighting function (optional)

4. How to fit with the local points?
Instance/Memory-based Learning

1. A distance metric

2. How many nearby neighbors to look at?

3. A weighting function (optional)

4. How to fit with the local points?

Slide credit: Carlos Guestrin
Recall: 1-Nearest neighbor classifier

- **Training data**
  \[(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(N)}, y^{(N)})\]

- **Learning**
  Do nothing.

- **Testing**
  \[h(x) = y^{(k)}, \text{ where } k = \arg\min_i D(x, x^{(i)})\]
Distance metrics ($x$: continuous variables)

- $L_2$-norm: Euclidean distance
  \[ D(x, x') = \sqrt{\sum_i (x_i - x_i')^2} \]

- $L_1$-norm: Sum of absolute difference
  \[ D(x, x') = \sum_i |x_i - x_i'| \]

- $L_{\infty}$-norm
  \[ D(x, x') = \max(|x_i - x_i'|) \]

- Scaled Euclidean distance
  \[ D(x, x') = \sqrt{\sum_i \sigma_i^2 (x_i - x_i')^2} \]

- Mahalanobis distance
  \[ D(x, x') = \sqrt{(x - x')^T A (x - x')} \]
Distance metrics ($x$: discrete variables)

• Example application: document classification
• Hamming distance
Distance metrics ($x$: Histogram / PDF)

- Histogram intersection
  
  \[
  \text{histint}(x, x') = 1 - \sum_i \min(x_i, x'_i)
  \]

- Chi-squared Histogram matching distance
  
  \[
  \chi^2(x, x') = \frac{1}{2} \sum_i \frac{[x_i - x'_i]^2}{x_i + x'_i}
  \]

- Earth mover’s distance (Cross-bin similarity measure)  
  
  - minimal cost paid to transform one distribution into the other  

[Rubner et al. IJCV 2000]
Distance metrics
($x$: gene expression microarray data)

• When “shape” matters more than values

• Want $D(x^{(1)}, x^{(2)}) < D(x^{(1)}, x^{(3)})$

• How?

  • **Correlation Coefficients**
    • Pearson, Spearman, Kendal, etc
Distance metrics ($x$: Learnable feature)

Large margin nearest neighbor (LMNN)

- Similarly labeled (target neighbor)
- Differently labeled (impostor)
- Differently labeled (impostor)
Instance/Memory-based Learning

1. A *distance metric*

2. *How many nearby neighbors to look at?*

3. A *weighting function (optional)*

4. *How to fit with the local points?*
kNN Classification

Classification decision boundaries

the data

NN classifier

5-NN classifier

Image credit: CS231 @ Stanford
Instance/Memory-based Learning

1. A distance metric

2. How many nearby neighbors to look at?

3. A weighting function (optional)

4. How to fit with the local points?
Issue: Skewed class distribution

• Problem with majority voting in kNN

• Intuition: nearby points should be weighted strongly, far points weakly

• Apply weight

\[ w^{(i)} = \exp\left(-\frac{d(x^{(i)}, query)^2}{\sigma^2}\right) \]

• \( \sigma^2 \): Kernel width
Instance/Memory-based Learning

1. A distance metric

2. How many nearby neighbors to look at?

3. A weighting function (optional)

4. How to fit with the local points?
1-NN for Regression

• Just predict the same output as the nearest neighbour.

Figure credit: Carlos Guestrin
1-NN for Regression

• Often bumpy (overfits)

Figure credit: Andrew Moore
9-NN for Regression

• Predict the averaged of k nearest neighbor values

Figure credit: Andrew Moore
Weighting/Kernel functions

Weight

\[ w(i) = \exp\left(-\frac{d(x^{(i)}, \text{query})^2}{\sigma^2}\right) \]

Prediction (use all the data)

\[ y = \sum_i w(i)y(i) / \sum_i w(i) \]

(Our examples use Gaussian)

Slide credit: Carlos Guestrin
Effect of Kernel Width

• What happens as $\sigma \to \infty$?

• What happens as $\sigma \to 0$?

Kernel regression
Problems with Instance-Based Learning

• Expensive
  • No Learning: most real work done during testing
  • For every test sample, must search through all dataset – very slow!
  • Must use tricks like approximate nearest neighbour search

• Doesn’t work well when large number of irrelevant features
  • Distances overwhelmed by noisy features

• Curse of Dimensionality
  • Distances become meaningless in high dimensions

Slide credit: Dhruv Batra
Curve of dimensionality

- Consider a **hypersphere** with radius \( r \) and dimension \( d \)
- Consider **hypercube** with edge of length \( 2r \)

\[
\frac{V_{\text{hypersphere}}}{V_{\text{hypercube}}} = \frac{\pi^{d/2}}{d2^{d-1}\Gamma(d/2)} \rightarrow 0 \text{ as } d \rightarrow \infty.
\]

- Distance between center and the corners is \( r\sqrt{d} \)
- Hypercube consist almost entirely of the “corners”
Hyperparameter selection

• How to choose K?

• Which distance metric should I use? L2, L1?

• How large the kernel width $\sigma^2$ should be?

• ....
Tune hyperparameters on the test dataset?

• Will give us a stronger performance on the test set!

• Why this is not okay? Let’s discuss

Evaluate on the test set only a single time, at the very end.
Validation set

- Split training set: A *fake* test set to tune hyper-parameters

```python
# assume we have Xtr_rows, Ytr, Xte_rows, Yte as before
# recall Xtr_rows is 50,000 x 3072 matrix
Xval_rows = Xtr_rows[:1000, :] # take first 1000 for validation
Yval = Ytr[:1000]
Xtr_rows = Xtr_rows[1000:, :] # keep last 49,000 for train
Ytr = Ytr[1000:]

# find hyperparameters that work best on the validation set
validation_accuracies = []
for k in [1, 3, 5, 10, 20, 50, 100]:

    # use a particular value of k and evaluation on validation data
    nn = NearestNeighbor()
    nn.train(Xtr_rows, Ytr)
    # here we assume a modified NearestNeighbor class that can take a k as input
    Yval_predict = nn.predict(Xval_rows, k = k)
    acc = np.mean(Yval_predict == Yval)
    print 'accuracy: %f' % (acc,)

    # keep track of what works on the validation set
    validation_accuracies.append((k, acc))
```
Cross-validation

• 5-fold cross-validation -> split the training data into 5 equal folds
• 4 of them for training and 1 for validation
Hyper-parameters selection

• Split training dataset into train/validation set (or cross-validation)

• Try out different values of hyper-parameters and evaluate these models on the validation set

• Pick the best performing model on the validation set

• Run the selected model on the test set. Report the results.
Things to remember

• Supervised Learning
  • Training/testing data; classification/regression; Hypothesis

• k-NN
  • Simplest learning algorithm
  • With sufficient data, very hard to beat “strawman” approach

• Kernel regression/classification
  • Set k to n (number of data points) and chose kernel width
  • Smoother than k-NN

• Problems with k-NN
  • Curse of dimensionality
  • Not robust to irrelevant features
  • Slow NN search: must remember (very large) dataset for prediction
Next class: Linear Regression