Spring 2023

SWCON253: Machine Learning

Lecture 11
K-Nearest Neighbors

EZ

Jinwoo Choi
Assistant Professor
CSE, Kyung Hee University

Table of Contents

- Recap: supervised learning
  - Setup
  - Basic concepts

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

- Recap: Validation, cross-validation, hyperparameter tuning

Supervised learning

- **Input**: x (Images, texts, emails)

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

- **Data**: $\left(x^{(1)}, y^{(1)}\right), \left(x^{(2)}, y^{(2)}\right), \cdots, \left(x^{(N)}, y^{(N)}\right)$ (Labeled dataset)

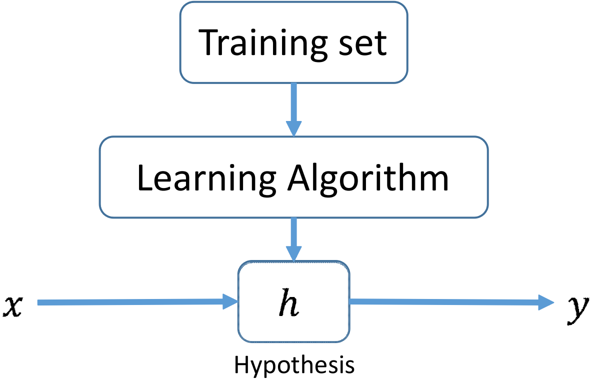
True function

- **(Unknown) Target function**: $f: x \rightarrow y$ ("True" mapping)

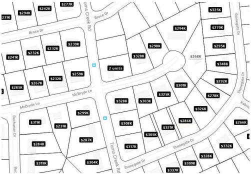
f에 근사

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

- **Learning = search in hypothesis space**


Training set
Learning Algorithm
x
h
y
Hypothesis

Regression



Training set

Learning Algorithm

x → h → y

Size of house

Hypothesis

Estimated price

Classification

Training set

Learning Algorithm

x → h → y ‘Mug’

Unseen image

Hypothesis

Predicted object class

Image credit: CS231n @ Stanford

Procedural view of supervised learning

- **Training Stage:**
  - Raw data $\rightarrow x$ 생 데이터에서 쓸모있는 특징 추출 (Feature Extraction)
  - Training data $\{(x, y)\} \rightarrow h$ (Learning)

    학습

- **Testing Stage**
  - Raw data $\rightarrow x$ (Feature Extraction)
  - Test data $x \rightarrow h(x)$ (Apply function, evaluate error)

Slide credit: Dhruv Batra

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 (→모델 정하기) class: $H = \{h: X \to Y\}$
- **Learning/estimation**: Find best hypothesis in the model class.
- **Model selection:** Try different models. Picks the best one. (More on this later)

Nearest neighbor classifier

- **Training data**

$$\left(x^{(1)}, y^{(1)}\right), \left(x^{(2)}, y^{(2)}\right), \cdots, \left(x^{(N)}, y^{(N)}\right)$$

- **Learning**

Do nothing.

- **Testing**

$$h(x) = y^{(k)}, \text{ where } k = \text{argmin}_{\text{i}}\, D\left(x, x^{(i)}\right)$$

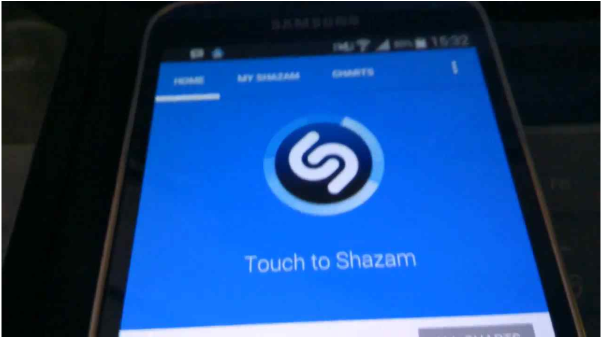
입력값과 훈련 데이터의 거리를 재서 가장 가까운 것이 출력

Face recognition

Image credit: MegaFace

Face recognition – surveillance application

Music identification



https://www.youtube.com/watch?v=TKNNOMddkNc

Synonyms

- Nearest Neighbors

- k-Nearest Neighbors

- Member of following families:
  - Instance-based Learning
  - Memory-based Learning
  - Exemplar methods
  - Non-parametric methods 파라미터/업데이트 이런거 없음

Instance/Memory-based Learning

1. A distance metric

2. How ~~many~~ k nearby neighbors to look at?

k 개 중에서 투표하는 method

3. *How to fit with the local points?*

Slide credit: Carlos Guestrin

Instance/Memory-based Learning

1. A distance metric

2. How many nearby neighbors to look at?

3. *How to fit with the local points?*

Slide credit: Carlos Guestrin

Recall: 1-Nearest neighbor classifier

- **Training data**

$$\left(x^{(1)}, y^{(1)}\right), \left(x^{(2)}, y^{(2)}\right), \cdots, \left(x^{(N)}, y^{(N)}\right)$$

- **Learning**

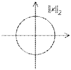
Do nothing.

- **Testing**

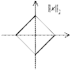
$$h(x) = y^{(k)}, \text{ where } k = \mathrm{argmin_i}\ D\left(x, x^{(i)}\right)$$

Distance metrics (x: continuous variables)

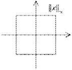
- L_2-norm: Euclidean distance $D(x, x') = \sqrt{\sum_i (x_i - x_i')^2}$



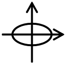
- L_1-norm: Sum of absolute difference $D(x, x') = \sum_i |x_i - x_i'|$



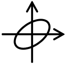
- L_{inf}-norm $D(x, x') = \max(|x_i - x_i'|)$



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

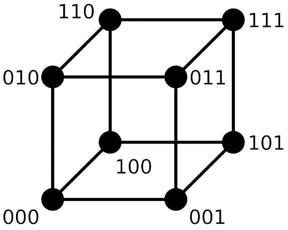


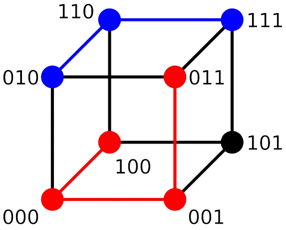
- Mahalanobis distance $D(x, x') = \sqrt{(x - x')^\top A (x - x')}$



Distance metrics (x: discrete variables)

- Example application: document classification
- Hamming distance





Distance metrics (x: Histogram / PDF)

- Histogram intersection 분포

$$\mathrm{histint}(x, x') = 1 - \sum_i \min(x_i, x_i')$$

↳ 2개의 히스토그램을 겹치고 교집합

- 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]

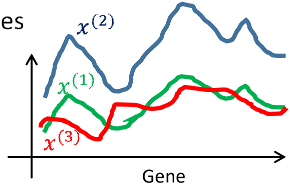
Distance metrics
(x: gene expression microarray data)

- When "shape" matters more than values

유사도가 더 중요

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

- How?
- **Correlation Coefficients** 상관계수
  - Pearson, Spearman, Kendal, etc



Instance/Memory-based Learning

1. A distance metric

2. How many nearby neighbors to look at?

3. *How to fit with the local points?*

Slide credit: Carlos Guestrin

kNN Classification

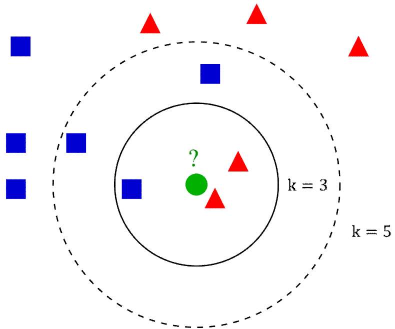


Image credit: Wikipedia

Classification decision boundaries

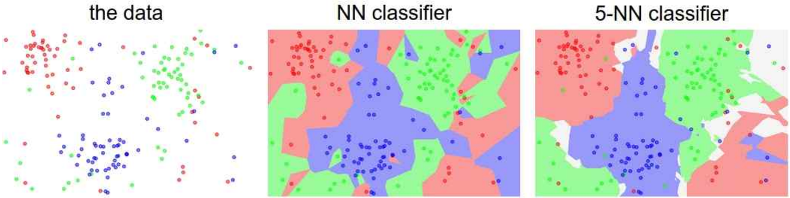


Image credit: CS231 @ Stanford

Instance/Memory-based Learning

1. A distance metric

2. How many nearby neighbors to look at?

3. *How to fit with the local points?*

Slide credit: Carlos Guestrin

Instance/Memory-based Learning

1. A distance metric

2. How many nearby neighbors to look at?

3. *How to fit with the local points?*

Slide credit: Carlos Guestrin

1-NN for Regression

- Just predict the same output as the nearest neighbour.

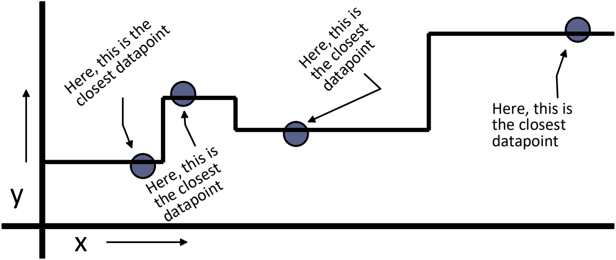
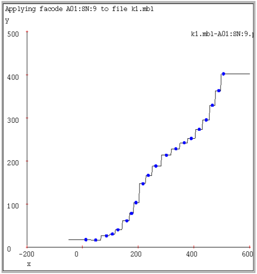


Figure credit: Carlos Guestrin

1-NN for Regression

- Often bumpy (overfits)



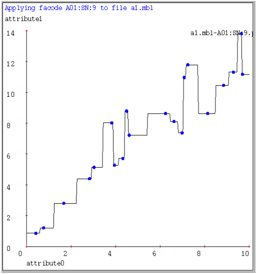


Figure credit: Andrew Moore

9-NN for Regression

- Predict the averaged of k nearest neighbor values

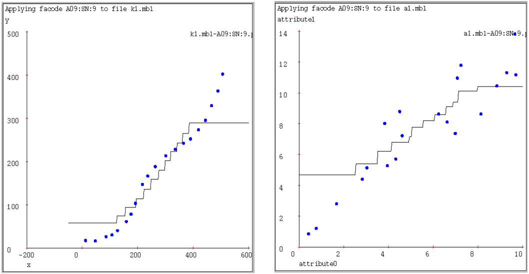


Figure credit: Andrew Moore

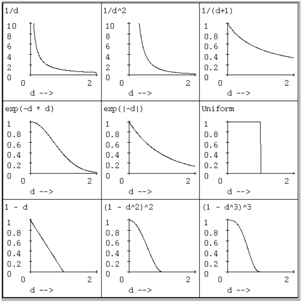
Weighting/Kernel functions

Weight

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

Prediction (use all the data)

$$y = \sum_i w^{(i)} y^{(i)} / \sum_i w^{(i)}$$

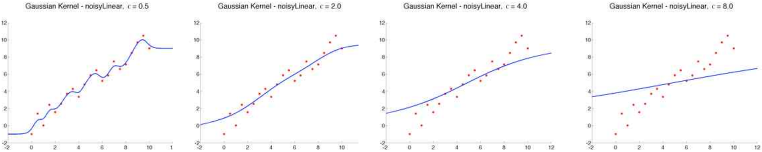


(Our examples use Gaussian)

Effect of Kernel Width

- What happens as σ→inf?

- What happens as σ→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

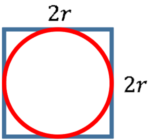
- Curse of Dimensionality
  - Distances become meaningless in high dimensions

Curse of dimensionality

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

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

- Distance between center and the corners is $r\sqrt{d}$
- Hypercube consist almost entirely of the "corners"



$d = 2$

Overcoming drawbacks of kNN

- Overcoming the curse of dimensionality?
  - Dimensionality reduction (more on this later)
  - Random projection

- Test time too slow?
  - Approaximate nearest neighbor search (e.g., http://www.cs.ubc.ca/research/flann/)

- Noisy features?
  - Outlier detection
  - Learnable features

Hyperparameter selection

- How to choose K?
- Which distance metric should I use? L2, L1?
- How large the kernel width σ^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.

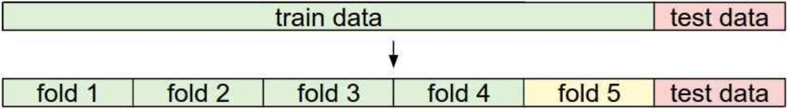
Validation set

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

```
# 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

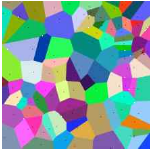
- 5-fold cross-validation -> split the training data into 5 equal folds
- 4 of them for training and 1 for validation



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