

CSCI567 Machine Learning (Fall 2020)

Prof. Haipeng Luo

U of Southern California

Aug 27, 2020

1 / 62

About this course

Outline

- 1 About this course
- 2 Overview of machine learning
- 3 Classification and Nearest Neighbor Classifier (NNC)
- 4 Some theory on NNC

3 / 62

Outline

- 1 About this course
- 2 Overview of machine learning
- 3 Classification and Nearest Neighbor Classifier (NNC)
- 4 Some theory on NNC

2 / 62

About this course

Overview

Nature of this course

- Covers standard statistical machine learning methods (supervised learning, unsupervised learning, etc.)
- Particular focuses are on the conceptual understanding and derivation of these methods

Learning objectives:

- Hone skills on grasping abstract concepts and thinking critically to solve problems with machine learning techniques
- Solidify your knowledge with hand-on programming tasks
- Prepare you for studying advanced machine learning techniques

4 / 62

Teaching logistics

Lectures: Thu 5:00-7:20pm

Discussions: Thu 7:30-8:20pm (by TAs, same Zoom link)

Zoom:

- have to sign in
- feel free to unmute and ask questions (avoid chat box)
- be patient if connection is lost
- let me know if you have any comments

Teaching staff

3 TAs (lecture/discussion, quiz, etc.)

- Liyu Chen
- Sungyong Seo
- Mengxiao Zhang

4 CPs (homework, project, etc.)

- Shubham Nagarkar
- Sowmya Penmetsa
- Anamay Sarkar
- Dhiti Thakkar

Emails are on course website; office hours are on Piazza→Resources→Staff

Online platforms

Course website:

https://haipeng-luo.net/courses/CSCI567/2020_fall

- general information (schedule, slides, homework, etc.)

Piazza: <https://piazza.com/usc/fall2020/csci567>

- main discussion forum
- everyone has to enroll

D2L: <https://courses.uscdcn.net/d2l/login>

- recorded lectures/discussions
- submit written assignments
- grade posting

Vocareum and Crowdmark

Prerequisites

- Undergraduate level training in **probability and statistics, linear algebra, (multivariate) calculus**

Important: attend today's discussion session to see if you have the required background

- Programming: Python and necessary packages (e.g. numpy)
not an intro-level CS course, no training of basic programming skills.

Slides and readings

Lectures

Lecture slides/handouts will be posted before the class (and possibly updated after).

Readings

- No required textbooks
- Main recommended readings:
 - Machine Learning: A Probabilistic Perspective by Kevin Murphy
 - Elements of Statistical Learning by Hastie, Tibshirani and Friedman
- More: see course website

9 / 62

Homework

5 written assignments (problem sets):

- submit one pdf to D2L (scanned copy or typeset with LaTeX etc.)
- graded based on correctness
- collaboration is permitted at high-level but must be stated (each member has to make a separate submission)
- finding solutions online or from other sources → *zero grade*
- 3 late days in total, at most *one* can be used for each assignment
- A two-day window for re-grading (regarding *factual errors*)

11 / 62

Grade

Structure:

- 30%: 5 written assignments
- 40%: 2 quizzes
- 30%: 1 programming project

Initial cut-offs (for A and B):

- B- = [70, 75), B = [75, 80), B+ = [80, 85)
- A- = [85, 90), A = [90, 100]

Important: final cut-offs will NOT be released. If adjusted they could only be LOWER.

10 / 62

Programing Project

Done on **Vocareum**

- easy-to-use platform to submit your code for auto-grading
- you will be invited to register next week
- consists of several tasks related to lectured materials (in Python)
- you can submit at most 10 times for each task and see your grade immediately
- *important changes*: the project is available throughout the semester (due 12/08, no late days), you can either
 - do each task right after the respective lecture to strengthen your understanding
 - or do everything in the end of the semester if you want to focus on the theory first

12 / 62

Quizzes

First one on **10/08**, second one on **11/19**.

- in class, 5:00-7:30
- open-book, no collaboration or consultation from others allowed
- Zoom breakout rooms, each proctored by one TA/CP (camera on)
- done on Crowdmark (no printer required, but need to take pictures)

13 / 62

Outline

- 1 About this course
- 2 Overview of machine learning
- 3 Classification and Nearest Neighbor Classifier (NNC)
- 4 Some theory on NNC

15 / 62

Academic honesty and integrity

Plagiarism and other unacceptable violations

- neither ethical nor in your self-interest
- zero-tolerance

14 / 62

What is machine learning?

One possible definition¹

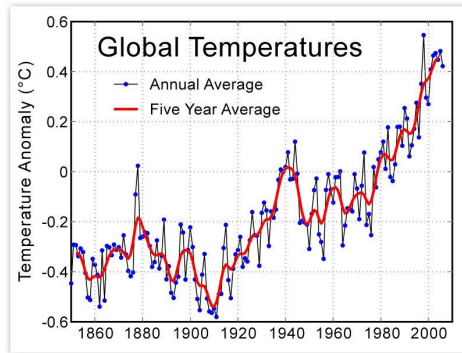
a set of methods that can automatically *detect patterns* in data, and then use the uncovered patterns to *predict future data*, or to perform other kinds of decision making *under uncertainty*

cf. Murphy's book

16 / 62

Example: detect patterns

How the temperature has been changing?



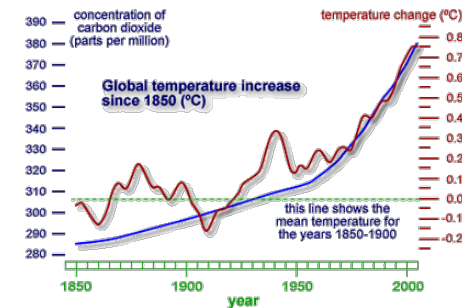
Patterns

- Seems going up
- Repeated periods of going up and down.

17 / 62

How do we describe the pattern?

Build a model: fit the data with a polynomial function

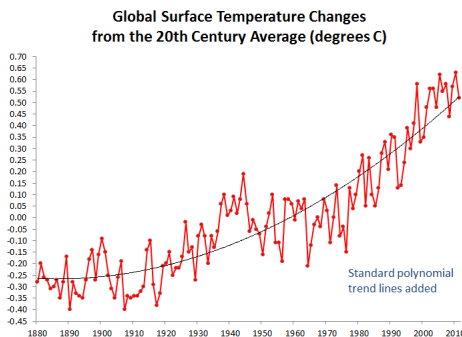


- The model is not accurate for individual years
- But collectively, the model captures the major trend

18 / 62

Predicting future

What is temperature of 2010?



- Again, the model is not accurate for that specific year
- But then, it is close to the actual one

19 / 62

What we have learned from this example?

Key ingredients in machine learning

- Data
collected from past observation (we often call them *training data*)
- Modeling
designed to capture the patterns in the data
 - The model does not have to be true — "All models are wrong, but some are useful" by George Box.
- Prediction
apply the model to forecast what is going to happen in future

20 / 62

A rich history of applying statistical learning methods

Recognizing flowers (by R. Fisher, 1936)

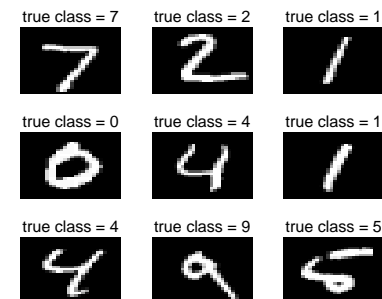
Types of Iris: setosa, versicolor, and virginica



21 / 62

Huge success 30 years ago

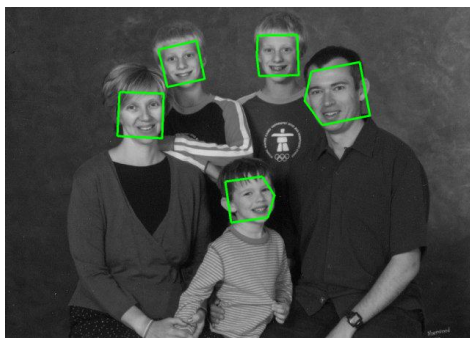
Recognizing handwritten zipcodes (AT&T Labs, late 1990s)



22 / 62

More modern ones, in your social life

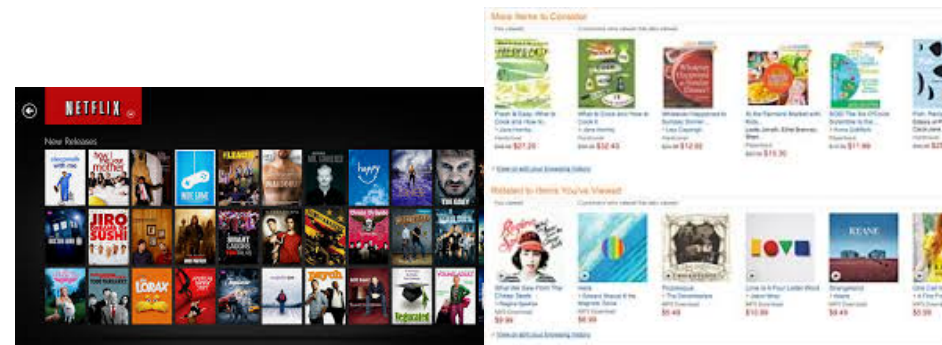
Recognizing your friends on Facebook



23 / 62

It might be possible to know about you than yourself

Recommending what you might like



24 / 62

Why is machine learning so hot?

• Tons of consumer applications:

- speech recognition, information retrieval and search, email and document classification, stock price prediction, object recognition, biometrics, etc
- Highly desirable expertise from industry: Google, Facebook, Microsoft, Uber, Twitter, IBM, Amazon, ...

• Enable scientific breakthrough

- Climate science: understand global warming cause and effect
- Biology and genetics: identify disease-causing genes and gene networks
- Social science: social network analysis; social media analysis
- Business and finance: marketing, operation research
- Emerging ones: healthcare, energy, ...

25 / 62

Outline

- 1 About this course
- 2 Overview of machine learning
- 3 Classification and Nearest Neighbor Classifier (NNC)
 - Intuitive example
 - General setup for classification
 - Algorithm
 - How to measure performance
 - Variants, Parameters, and Tuning
 - Summary
- 4 Some theory on NNC

27 / 62

What is in machine learning?

Different flavors of learning problems

- Supervised learning
Aim to predict (as in previous examples)
- Unsupervised learning
Aim to discover hidden and latent patterns and explore data
- Decision making (e.g. reinforcement learning)
Aim to act optimally under uncertainty
- Many other paradigms

The focus and goal of this course

- Supervised learning (before Quiz 1)
- Unsupervised learning (after Quiz 1)

26 / 62

Recognizing flowers

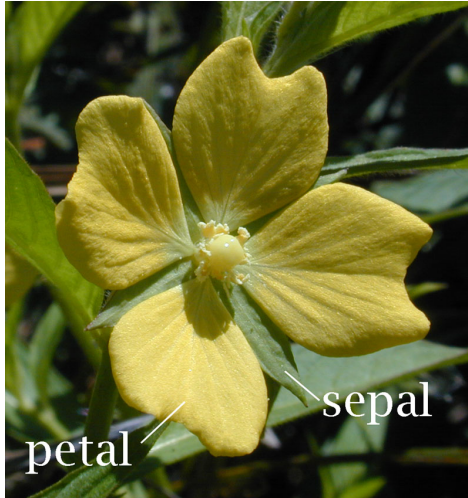
Types of Iris: setosa, versicolor, and virginica



28 / 62

Measuring the properties of the flowers

Features and attributes: the widths and lengths of sepal and petal



29 / 62

Often, data is conveniently organized as a table

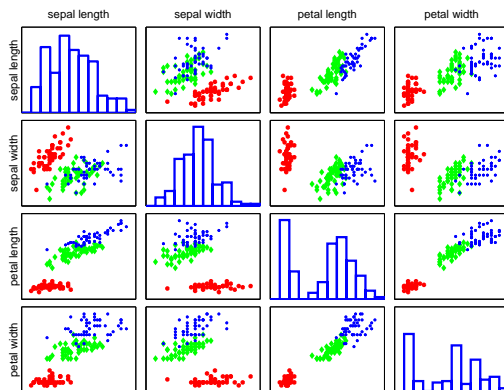
Fisher's Iris Data				
Sepal length ↕	Sepal width ↕	Petal length ↕	Petal width ↕	Species ↕
5.1	3.5	1.4	0.2	<i>I. setosa</i>
4.9	3.0	1.4	0.2	<i>I. setosa</i>
4.7	3.2	1.3	0.2	<i>I. setosa</i>
4.6	3.1	1.5	0.2	<i>I. setosa</i>
5.0	3.6	1.4	0.2	<i>I. setosa</i>
5.4	3.9	1.7	0.4	<i>I. setosa</i>
4.6	3.4	1.4	0.3	<i>I. setosa</i>
5.0	3.4	1.5	0.2	<i>I. setosa</i>
4.4	2.9	1.4	0.2	<i>I. setosa</i>
4.9	3.1	1.5	0.1	<i>I. setosa</i>

30 / 62

Pairwise scatter plots of 131 flower specimens

Visualization of data helps identify the right learning model to use

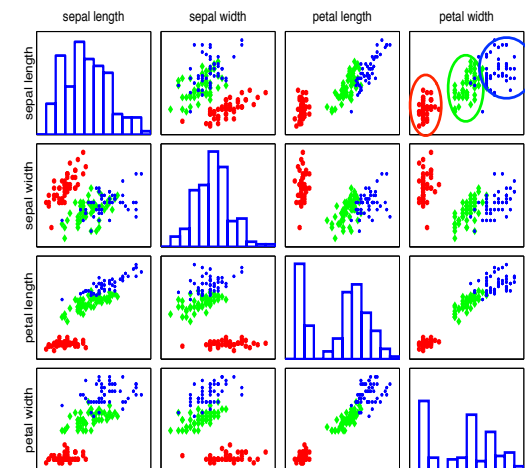
Each colored point is a flower specimen: **setosa**, **versicolor**, **virginica**



31 / 62

Different types seem well-clustered and separable

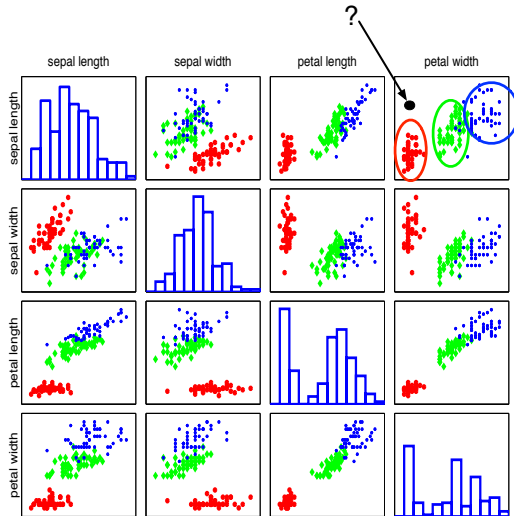
Using two features: petal width and sepal length



32 / 62

Labeling an unknown flower type

Closer to red cluster: so labeling it as **setosa**



33 / 62

General setup for multi-class classification

Training data (set)

- N samples/instances: $\mathcal{D}^{\text{TRAIN}} = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$
- Each $x_n \in \mathbb{R}^D$ is called a feature vector.
- Each $y_n \in [C] = \{1, 2, \dots, C\}$ is called a label/class/category.
- They are used to learn a *classifier* $f : \mathbb{R}^D \rightarrow [C]$ for future prediction.

Special case: binary classification

- Number of classes: $C = 2$
- Conventional labels: $\{0, 1\}$ or $\{-1, +1\}$

34 / 62

Nearest neighbor classification (NNC)

The index of the **nearest neighbor** of a point x is

$$\text{nn}(x) = \underset{n \in [N]}{\operatorname{argmin}} \|x - x_n\|_2 = \underset{n \in [N]}{\operatorname{argmin}} \sqrt{\sum_{d=1}^D (x_d - x_{nd})^2}$$

where $\|\cdot\|_2$ is the L_2 /Euclidean distance.

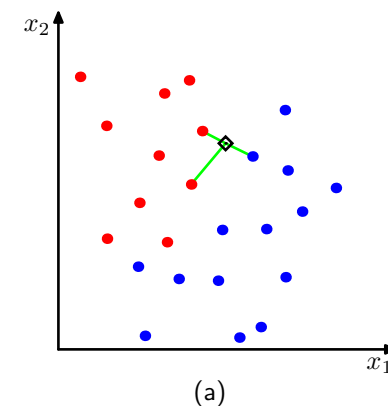
Classification rule

$$f(x) = y_{\text{nn}(x)}$$

35 / 62

Visual example

In this 2-dimensional example, the nearest point to x is a **red training instance**, thus, x will be labeled as **red**.



36 / 62

Example: classify Iris with two features

Training data

ID (n)	petal width (x_1)	sepal length (x_2)	category (y)
1	0.2	5.1	setoas
2	1.4	7.0	versicolor
3	2.5	6.7	virginica
\vdots	\vdots	\vdots	

Flower with unknown category

petal width = 1.8 and sepal width = 6.4 (i.e. $\mathbf{x} = (1.8, 6.4)$)

Calculating distance $\|\mathbf{x} - \mathbf{x}_n\|_2 = \sqrt{(x_1 - x_{n1})^2 + (x_2 - x_{n2})^2}$

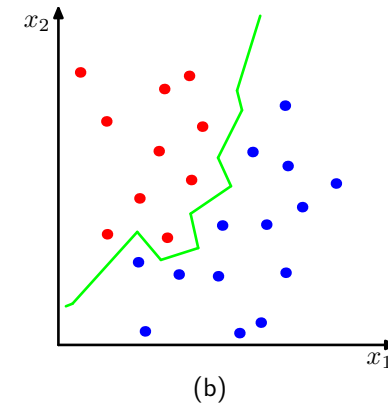
ID	distance
1	1.75
2	0.72
3	0.76

Thus, the category is *versicolor*.

37 / 62

Decision boundary

For every point in the space, we can determine its label using the NNC rule. This gives rise to a *decision boundary* that partitions the space into different regions.



38 / 62

Is NNC doing the right thing for us?

Intuition

We should compute **accuracy** — the percentage of data points being correctly classified, or the **error rate** — the percentage of data points being incorrectly classified. (accuracy + error rate = 1)

Defined on the training data set

$$A^{\text{TRAIN}} = \frac{1}{N} \sum_n \mathbb{I}[f(\mathbf{x}_n) == y_n], \quad \varepsilon^{\text{TRAIN}} = \frac{1}{N} \sum_n \mathbb{I}[f(\mathbf{x}_n) \neq y_n]$$

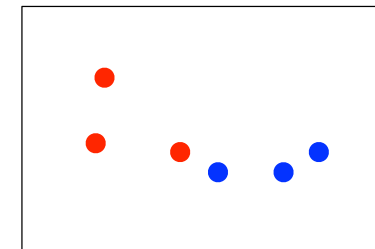
where $\mathbb{I}[\cdot]$ is the indicator function.

Is this the right measure?

39 / 62

Example

Training data



What are A^{TRAIN} and $\varepsilon^{\text{TRAIN}}$?

$$A^{\text{TRAIN}} = 100\%, \quad \varepsilon^{\text{TRAIN}} = 0\%$$

For every training data point, its nearest neighbor is itself.

40 / 62

Test Error

Does it mean nearest neighbor is a very good algorithm?

Not really, having zero training error is simple!

We should care about accuracy when predicting unseen data

Test/Evaluation data

- $\mathcal{D}^{\text{TEST}} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_M, y_M)\}$
- A fresh dataset, *not* overlap with training set.
- Test accuracy and test error

$$A^{\text{TEST}} = \frac{1}{M} \sum_m \mathbb{I}[f(\mathbf{x}_m) == y_m], \quad \varepsilon^{\text{TEST}} = \frac{1}{M} \sum_m \mathbb{I}[f(\mathbf{x}_m) \neq y_m]$$

- Good measurement of a classifier's performance

Variant 1: measure nearness with other distances

Previously, we use the Euclidean distance

$$\text{nn}(\mathbf{x}) = \underset{n \in [\mathbf{N}]}{\operatorname{argmin}} \|\mathbf{x} - \mathbf{x}_n\|_2$$

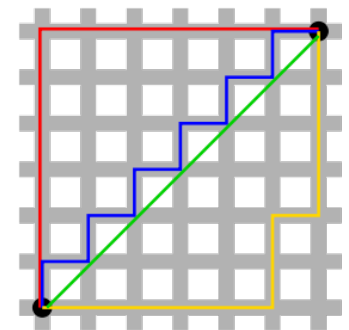
Many other alternative distances

E.g., the following L_1 distance (i.e., city block distance, or Manhattan distance)

$$\|\mathbf{x} - \mathbf{x}_n\|_1 = \sum_{d=1}^D |x_d - x_{nd}|$$

More generally, L_p distance (for $p \geq 1$):

$$\|\mathbf{x} - \mathbf{x}_n\|_p = \left(\sum_d |x_d - x_{nd}|^p \right)^{1/p}$$



Green line is Euclidean distance.
Red, Blue, and Yellow lines are L_1 distance

Variant 2: K-nearest neighbor (KNN)

Increase the number of nearest neighbors to use?

- 1-nearest neighbor: $\text{nn}_1(\mathbf{x}) = \underset{n \in [\mathbf{N}]}{\operatorname{argmin}} \|\mathbf{x} - \mathbf{x}_n\|_2$
- 2-nearest neighbor: $\text{nn}_2(\mathbf{x}) = \underset{n \in [\mathbf{N}] \setminus \{\text{nn}_1(\mathbf{x})\}}{\operatorname{argmin}} \|\mathbf{x} - \mathbf{x}_n\|_2$
- 3-nearest neighbor: $\text{nn}_3(\mathbf{x}) = \underset{n \in [\mathbf{N}] \setminus \{\text{nn}_1(\mathbf{x}), \text{nn}_2(\mathbf{x})\}}{\operatorname{argmin}} \|\mathbf{x} - \mathbf{x}_n\|_2$

The set of K-nearest neighbor

$$\text{knn}(\mathbf{x}) = \{\text{nn}_1(\mathbf{x}), \text{nn}_2(\mathbf{x}), \dots, \text{nn}_K(\mathbf{x})\}$$

Note: we have

$$\|\mathbf{x} - \mathbf{x}_{\text{nn}_1(\mathbf{x})}\|_2 \leq \|\mathbf{x} - \mathbf{x}_{\text{nn}_2(\mathbf{x})}\|_2 \leq \dots \leq \|\mathbf{x} - \mathbf{x}_{\text{nn}_K(\mathbf{x})}\|_2$$

How to classify with K neighbors?

Classification rule

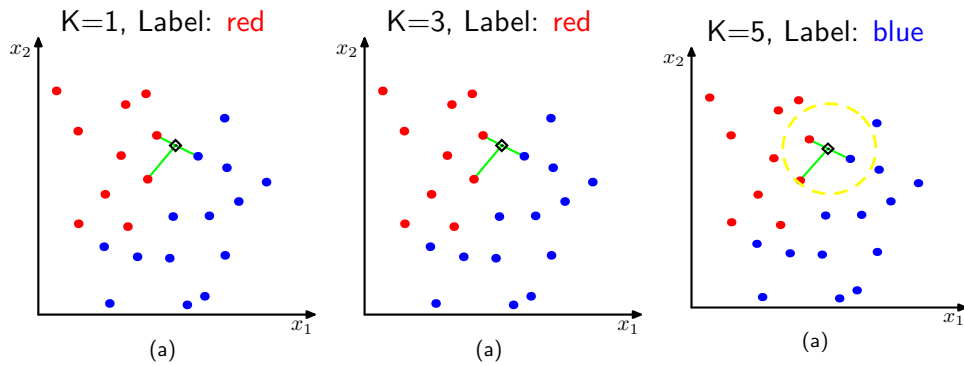
- Every neighbor votes: naturally \mathbf{x}_n votes for its label y_n .
- Aggregate everyone's vote on a class label c

$$v_c = \sum_{n \in \text{knn}(\mathbf{x})} \mathbb{I}(y_n == c), \quad \forall c \in [\mathbf{C}]$$

- Predict with the majority

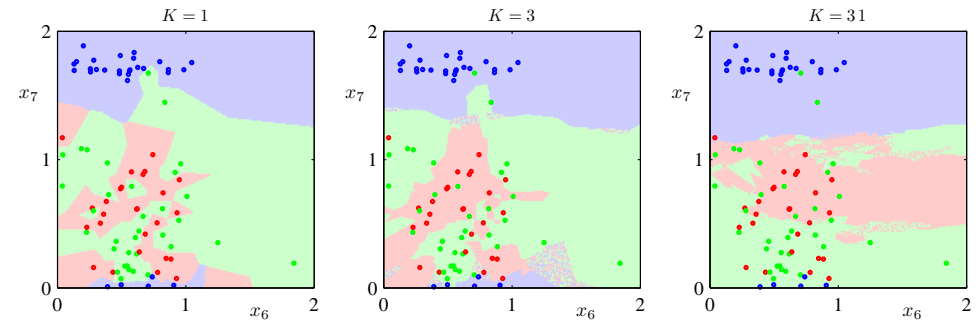
$$f(\mathbf{x}) = \underset{c \in [\mathbf{C}]}{\operatorname{argmax}} v_c$$

Example



45 / 62

Decision boundary



When K increases, the decision boundary becomes smoother.

What happens when $K = N$?

46 / 62

Variant 3: Preprocessing data

One issue of NNC: *distances depend on units of the features!*

One solution: preprocess data so it looks more “normalized”.

Example:

- compute the means and standard deviations in each feature

$$\bar{x}_d = \frac{1}{N} \sum_n x_{nd}, \quad s_d^2 = \frac{1}{N-1} \sum_n (x_{nd} - \bar{x}_d)^2$$

- Scale the feature accordingly

$$x_{nd} \leftarrow \frac{x_{nd} - \bar{x}_d}{s_d}$$

Many other ways of normalizing data.

47 / 62

Which variants should we use?

Hyper-parameters in NNC

- The distance measure (e.g. the parameter p for L_p norm)
- K (i.e. how many nearest neighbor?)
- Different ways of preprocessing

Most algorithms have hyper-parameters. Tuning them is a significant part of applying an algorithm.

48 / 62

Tuning via a development dataset

Training data

- N samples/instances: $\mathcal{D}^{\text{TRAIN}} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$
- They are used to learn $f(\cdot)$

Test data

- M samples/instances: $\mathcal{D}^{\text{TEST}} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_M, y_M)\}$
- They are used to evaluate how well $f(\cdot)$ will do.

Development/Validation data

- L samples/instances: $\mathcal{D}^{\text{DEV}} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_L, y_L)\}$
- They are used to optimize hyper-parameter(s).

These three sets should *not* overlap!

49 / 62

Recipe

- For each possible value of the hyperparameter (e.g. $K = 1, 3, \dots$)
 - Train a model using $\mathcal{D}^{\text{TRAIN}}$
 - Evaluate the performance of the model on \mathcal{D}^{DEV}
- Choose the model with the best performance on \mathcal{D}^{DEV}
- Evaluate the model on $\mathcal{D}^{\text{TEST}}$

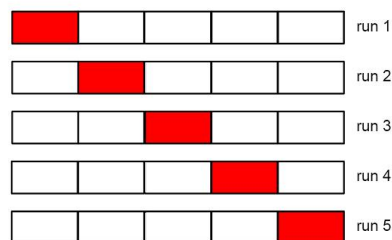
50 / 62

S-fold Cross-validation

What if we do not have a development set?

- Split the training data into S equal parts.
- Use each part *in turn* as a development dataset and use the others as a training dataset.
- Choose the hyper-parameter leading to best *average* performance.

S = 5: 5-fold cross validation



Special case: $S = N$, called leave-one-out.

51 / 62

Cross-validation recipe

- Split the training data into S equal parts. Denote each part as $\mathcal{D}_s^{\text{TRAIN}}$.
- For each possible value of the hyper-parameter (e.g. $K = 1, 3, \dots$)
 - For every $s \in [S]$
 - Train a model using $\mathcal{D}_{\setminus s}^{\text{TRAIN}} = \mathcal{D}^{\text{TRAIN}} - \mathcal{D}_s^{\text{TRAIN}}$
 - Evaluate the performance of the model on $\mathcal{D}_s^{\text{TRAIN}}$
 - Average the S performance metrics
- Choose the hyper-parameter with the best averaged performance
- **Use the best hyper-parameter to train a model using all $\mathcal{D}^{\text{train}}$**
- Evaluate the model on $\mathcal{D}^{\text{TEST}}$

52 / 62

Summary

Advantages of NNC

- Simple, easy to implement (wildly used in practice)

Disadvantages of NNC

- Computationally intensive for large-scale problems: $O(ND)$ for each prediction *naively*.
- Need to “*carry*” the training data around. This type of method is called *nonparametric*.
- Choosing the right hyper-parameters can be involved.

53 / 62

Outline

- 1 About this course
- 2 Overview of machine learning
- 3 Classification and Nearest Neighbor Classifier (NNC)
- 4 Some theory on NNC
 - Step 1: Expected risk
 - Step 2: The ideal classifier

55 / 62

Summary

Typical steps of developing a machine learning system:

- Collect data, split into training, development, and test sets.
- Train a model with a machine learning algorithm. Most often we apply cross-validation to tune hyper-parameters.
- Evaluate using the test data and report performance.
- Use the model to predict future/make decisions.

54 / 62

How good is NNC really?

To answer this question, we proceed in 3 steps

- 1 Define *more carefully* a performance metric for a classifier.
- 2 Hypothesize an ideal classifier - *the best possible one*.
- 3 Compare NNC to the ideal one.

56 / 62

Why does test error make sense?

Test error makes sense only when training set and test set are correlated.

Most standard assumption: every data point (\mathbf{x}, y) (from $\mathcal{D}^{\text{TRAIN}}$, \mathcal{D}^{DEV} , or $\mathcal{D}^{\text{TEST}}$) is an *independent and identically distributed (i.i.d.)* sample of an unknown joint distribution \mathcal{P} .

- often written as $(\mathbf{x}, y) \stackrel{i.i.d.}{\sim} \mathcal{P}$

Test error of a fixed classifier is therefore a *random variable*.

Need a more “certain” measure of performance (so it’s easy to compare different classifiers for example).

57 / 62

Expected error

What about the **expectation** of this random variable?

$$\mathbb{E}[\epsilon^{\text{TEST}}] = \frac{1}{M} \sum_{m=1}^M \mathbb{E}_{(\mathbf{x}_m, y_m) \sim \mathcal{P}} \mathbb{I}[f(\mathbf{x}_m) \neq y_m] = \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{P}} \mathbb{I}[f(\mathbf{x}) \neq y]$$

- i.e. the expected error/mistake of f

Test error is a proxy of expected error. *The larger the test set, the better the approximation.*

What about the expectation of training error? Is training error a good proxy of expected error?

58 / 62

Expected risk

More generally, for a loss function $L(y', y)$,

- e.g. $L(y', y) = \mathbb{I}[y' \neq y]$, called *0-1 loss*. **Default**
- many more other losses as we will see.

the *expected risk* of f is defined as

$$R(f) = \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{P}} L(f(\mathbf{x}), y)$$

59 / 62

Bayes optimal classifier

What should we predict for \mathbf{x} , *knowing* $\mathcal{P}(y|\mathbf{x})$?

Bayes optimal classifier: $f^*(\mathbf{x}) = \operatorname{argmax}_{c \in [C]} \mathcal{P}(c|\mathbf{x})$.

The optimal risk: $R(f^*) = \mathbb{E}_{\mathbf{x} \sim \mathcal{P}_{\mathbf{x}}} [1 - \max_{c \in [C]} \mathcal{P}(c|\mathbf{x})]$ where $\mathcal{P}_{\mathbf{x}}$ is the marginal distribution of \mathbf{x} .

It is easy to show $R(f^*) \leq R(f)$ for any f .

For special case $C = 2$, let $\eta(\mathbf{x}) = \mathcal{P}(0|\mathbf{x})$, then

$$R(f^*) = \mathbb{E}_{\mathbf{x} \sim \mathcal{P}_{\mathbf{x}}} [\min\{\eta(\mathbf{x}), 1 - \eta(\mathbf{x})\}].$$

60 / 62

Comparing NNC to Bayes optimal classifier

Come back to the question: how good is NNC?

Theorem (Cover and Hart, 1967)

Let f_N be the 1-nearest neighbor binary classifier using N training data points, we have (under mild conditions)

$$R(f^*) \leq \lim_{N \rightarrow \infty} \mathbb{E}[R(f_N)] \leq 2R(f^*)$$

i.e., expected risk of NNC in the limit is at most twice of the best possible.

A pretty strong guarantee.

In particular, $R(f^*) = 0$ implies $\mathbb{E}[R(f_N)] \rightarrow 0$.

Proof sketch

Fact: $x_{nn(x)} \rightarrow x$ as $N \rightarrow \infty$ with probability 1

$$\begin{aligned} \mathbb{E}[R(f_N)] &= \mathbb{E}[\mathbb{E}_{(x,y) \sim \mathcal{P}} \mathbb{I}[f_N(x) \neq y]] \\ &\rightarrow \mathbb{E}_{x \sim \mathcal{P}_x} \mathbb{E}_{y, y' \stackrel{i.i.d.}{\sim} \mathcal{P}(\cdot|x)} [\mathbb{I}[y' \neq y]] \\ &= \mathbb{E}_{x \sim \mathcal{P}_x} \mathbb{E}_{y, y' \stackrel{i.i.d.}{\sim} \mathcal{P}(\cdot|x)} [\mathbb{I}[y' = 0 \text{ and } y = 1] + \mathbb{I}[y' = 1 \text{ and } y = 0]] \\ &= \mathbb{E}_{x \sim \mathcal{P}_x} [\eta(x)(1 - \eta(x)) + (1 - \eta(x))\eta(x)] \\ &= 2\mathbb{E}_{x \sim \mathcal{P}_x} [\eta(x)(1 - \eta(x))] \\ &\leq 2\mathbb{E}_{x \sim \mathcal{P}_x} [\min\{\eta(x), (1 - \eta(x))\}] \\ &= 2R(f^*) \end{aligned}$$