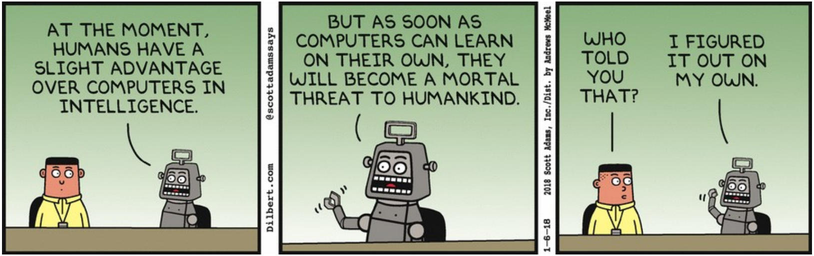
CSCI 567 Spring 2018
Lecture 01: Introduction



The Syllabus

- The syllabus is posted on blackboard and Piazza
- I expect you to have read it by Wednesday's lecture
- We will discuss some key pieces today

Teaching staff

- Instructor: Dr. Michael Shindler
  - Office: SAL 204
  - Open Office Hours: Monday 11:30-12:30
  - Limited: Tuesday 2:30 - 3:30, Wed 10:30 - 11:30
- Discussions:
  - Dr. Olivera Grujic
  - Dr. Selina Chu
- There are also several TAs
- Office hours will begin week 2
- Office hours will be announced soon.

Course Meetings

- There are two lectures each week
- There are twelve discussions each week
- There is also DEN sections
- You are responsible for lecture and discussion content

- There is also suggested reading across two books.
  - Kevin P. Murphy, *Machine Learning: A Probabilistic Perspective*
  - Trevor Hastie, Robert Tibshirani, and Jerome Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Second Edition*

Grading

- Three quizzes
  - Worth 15%, 20%, and then 25%.
  - Exams are on 2/23, 4/6, and 4/27.
- Five problem sets, 4% each
- Five programming assignments, 4% each

Grace Days for Programming

- You can extend the deadline for *programming assignments* with these
- Each gets you another 24 hours.
- You have four available.
- You will need to submit a form to use these.
- Form will be available for you by due date of programming assignment 1.

Problem Sets

- Some questions are reinforcement
- Some questions involve proofs or extensions.

Quizzes

- You must be in the room and seated by 4:10 on quiz day
- Quizzes begin at 4:15
- Quizzes end at 5:15 *promptly*
- Alternate arrangements needed? Let me know SOON.
- No makeup quizzes except for extreme circumstances
  (documentation will be required)

Regrade Policy

- For problem sets and programming:
  - We will release rubric/grading script
  - You should grade yours and compare results
  - If we graded incorrectly, tell us promptly
  - Details in written syllabus.
- For quizzes:
  - We will have regrade sessions.

Reasons that do not qualify for regrading

- I need to upgrade my grade to maintain/boost my GPA.
- I cannot graduate if my GPA is low
- I cannot graduate if I have failed this course.
- This is the last course I have taken before I graduate.
- I have done well in other courses
- I am a great programmer/theoretician.
- I have a deadline prior to the homework/quiz due date.
- I have a regular job requiring a lot of my attention.
- Exam/homework are not the best way to show my competency in learning.

What would happen if you fail?

What would happen if you fail?

ANY of the following can cause a fail in the class:

- Plagiarizing on any problem of a problem set
- Plagiarizing on any portion of a programming assignment
- Cheating on an exam
- This is not an exhaustive list.

Academic Honesty infractions will **always** be reported to the appropriate office.
I will **never** suggest less than a fail as a penalty for infractions

Teaching philosophy

The nature of this course

- Describe basic concepts and tools
- Describe algorithms and their development with intuition and rigor

Expectation on you

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

What to do before Wednesday's lecture

- Read the syllabus in its entirety
  - You are responsible for it.
- Sign up for github if you haven't already
  - You will need it to submit programming assignments.
- Put quiz dates on your calendar

What is Machine Learning?

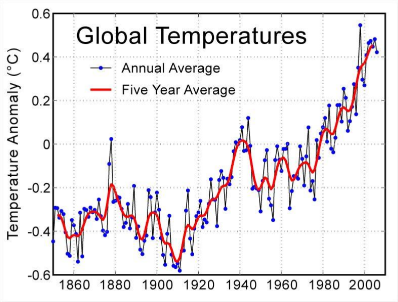
What is machine learning?

One possible definition[1]
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*

[1] cf. Murphy's book

Example: detect patterns

How temp has been changing in last 140 years?

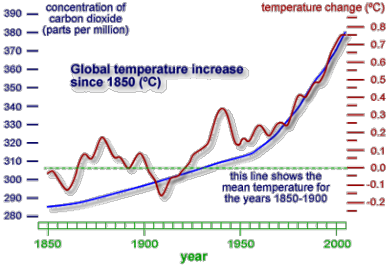


Patterns

- Seems going up
- Seems to be repeated periods of going up and down.

How do we describe the pattern?

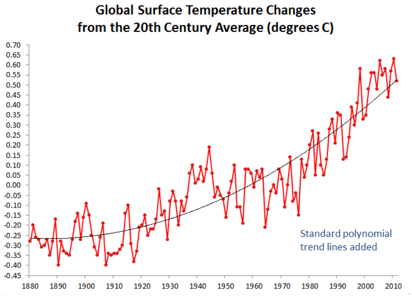
Build a model: fit the data with a polynomial function



- The model is not accurate for individual years
- But collectively, the model captures the major trend (for instance, we are not able to model the pattern of the *repeated up and down* with this polynomial function)

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

What we have learned from this example?

Key ingredients in the machine learning task

- Data
  collected from past observation (we often call them *training data*)
- Modeling
  devised to capture the patterns in the data
  - The model does not have to be true — as long as it is close, it is useful
  - We should tolerate randomness and mistakes — many interesting things are stochastic by nature.
- Prediction
  apply the model to forecast what is going to happen in future

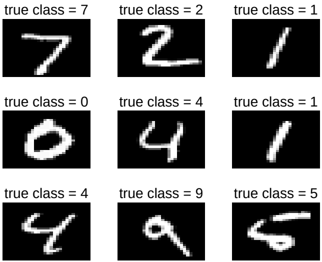
A rich history of applying statistical learning methods

Recognizing flowers (by R. Fisher, 1936)
Types of Iris: setosa, versicolor, and virginica



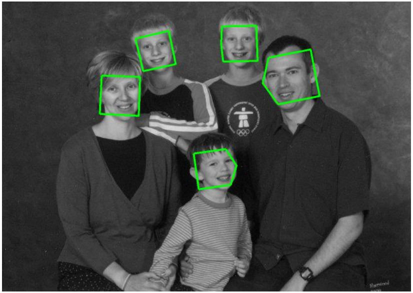
Huge success 20 years ago

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



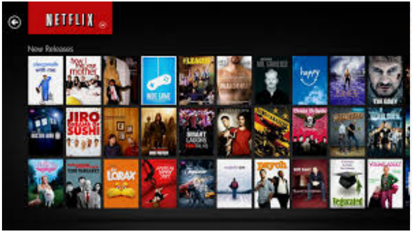
More modern ones, in your social life

Recognizing your friends on the Facebook



It might be possible to know about you than yourself

Recommending what you might like



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, Linkedin, Amazon, $\cdots$

Why is machine learning so hot?

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, $\cdots$

What is in machine learning?

Different flavors of learning problems

- Supervised learning
  Aim to predict (as in the previous list of applications)
- Unsupervised learning
  Aim to discover hidden and latent patterns and explore data
- 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)

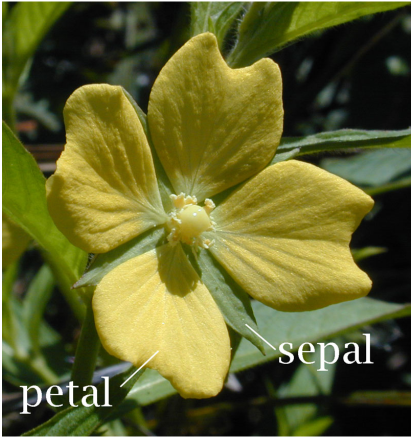
Nearest Neighbor Classifiers

Recognizing flowers

Types of Iris: setosa, versicolor, and virginica

Measuring the properties of the flowers

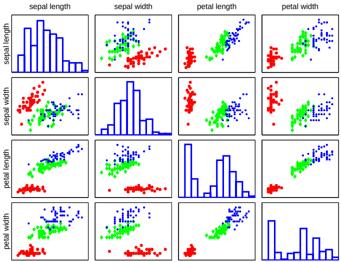
Features and attributes: the widths and lengths of sepal and petal



Pairwise scatter plots of 131 flower specimens

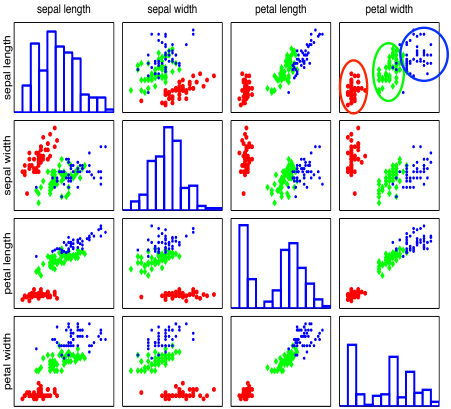
Visualization of data helps to identify the right learning model to use

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



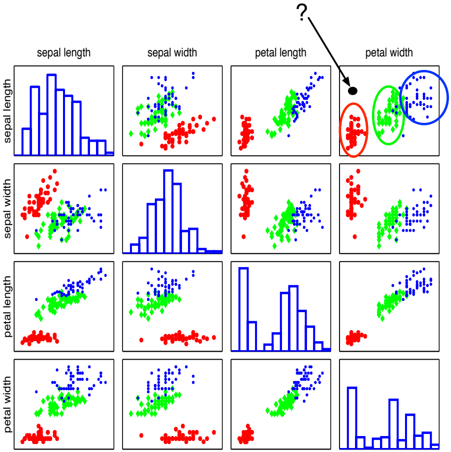
Different types seem well-clustered and separable

Using two features: petal width and sepal length



Labeling an unknown flower type

Closer to red cluster: so labeling it as setosa



Multi-class classification

Classify data into one of the multiple categories

- Input (feature vectors): $\boldsymbol{x} \in R^D$
- Output (label): $y \in [\mathrm{C}] = \{1, 2, \cdots, \mathrm{C}\}$
- Learning goal: $y = f(\boldsymbol{x})$

Special case: binary classification

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

More terminology

Training data (set)

- N samples/instances: $\mathcal{D}^{\text{TRAIN}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_\mathsf{N}, y_\mathsf{N})\}$
- They are used for learning $f(\cdot)$

Test (evaluation) data

- M samples/instances: $\mathcal{D}^{\text{TEST}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_\mathsf{M}, y_\mathsf{M})\}$
- They are used for assessing how well $f(\cdot)$ will do in predicting an unseen $\boldsymbol{x} \notin \mathcal{D}^{\text{TRAIN}}$

Training data and test data should *not* overlap:

$$\mathcal{D}^{\text{TRAIN}} \cap \mathcal{D}^{\text{TEST}} = \emptyset$$

Often, data is organized as a table

Ex: Iris data (click here for all data)

- 4 features
- 3 classes

Fisher's *Iris* Data

Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	*I. setosa*
4.9	3.0	1.4	0.2	*I. setosa*
4.7	3.2	1.3	0.2	*I. setosa*
4.6	3.1	1.5	0.2	*I. setosa*
5.0	3.6	1.4	0.2	*I. setosa*
5.4	3.9	1.7	0.4	*I. setosa*
4.6	3.4	1.4	0.3	*I. setosa*
5.0	3.4	1.5	0.2	*I. setosa*
4.4	2.9	1.4	0.2	*I. setosa*
4.9	3.1	1.5	0.1	*I. setosa*

Nearest neighbor classification (NNC)

Nearest neighbor

$$\boldsymbol{x}(1) = \boldsymbol{x}_{\mathrm{nn}(\boldsymbol{x})}$$

where $\mathrm{nn}(\boldsymbol{x}) \in [\mathrm{N}] = \{1, 2, \cdots, \mathrm{N}\}$, i.e., the index to one of the training instances,

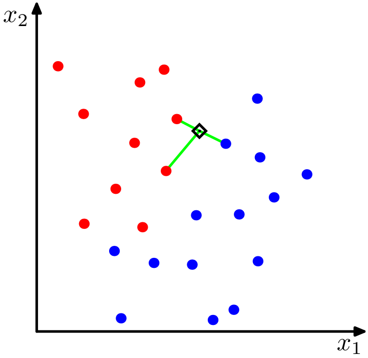
$$\mathrm{nn}(\boldsymbol{x}) = \arg\min_{n \in [\mathrm{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2 = \arg\min_{n \in [\mathrm{N}]} \sqrt{\sum_{d=1}^{\mathrm{D}} (x_d - x_{nd})^2}$$

Classification rule

$$y = f(\boldsymbol{x}) = y_{\mathrm{nn}(\boldsymbol{x})}$$

Visual example

In this 2-dimensional example, the nearest point to $\boldsymbol{x}$ is a red training instance, thus, $\boldsymbol{x}$ will be labeled as red.



(a)

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
⋮	⋮	⋮	

Flower with unknown category

petal width $= 1.8$ and sepal width $= 6.4$

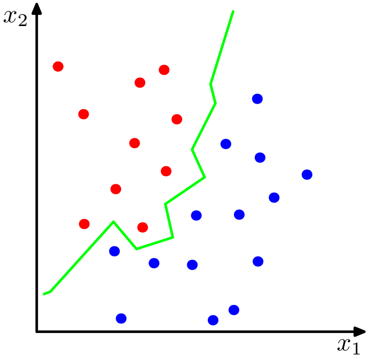
Calculating distance $= \sqrt{(x_1 - x_{n1})^2 + (x_2 - x_{n2})^2}$

ID	distance
1	1.75
2	0.72
3	0.76

Thus, the category is *versicolor* (the real category is *viriginica*)

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.



(b)

How to measure nearness with other distances?

Previously, we use the Euclidean distance

$$\text{nn}(\boldsymbol{x}) = \arg\min_{n \in [\text{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2$$

We can also use alternative distances
E.g., the following L_1 distance (i.e., city block distance, or Manhattan distance)

$$\begin{aligned}\text{nn}(\boldsymbol{x}) &= \arg\min_{n \in [\text{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_1 \\ &= \arg\min_{n \in [\text{N}]} \sum_{d=1}^{\text{D}} |x_d - x_{nd}|\end{aligned}$$

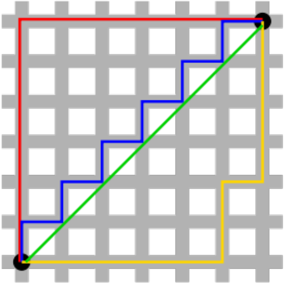


Figure: Green line is Euclidean distance. Red, Blue, and Yellow lines are L_1 distance

K-nearest neighbor (KNN) classification

Increase the number of nearest neighbors to use?

- 1-nearest neighbor:
  $\mathrm{nn}_1(\boldsymbol{x}) = \arg\min_{n \in [\mathrm{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2$
- 2nd-nearest neighbor:
  $\mathrm{nn}_2(\boldsymbol{x}) = \arg\min_{n \in [\mathrm{N}] - \mathrm{nn}_1(\boldsymbol{x})} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2$
- 3rd-nearest neighbor:
  $\mathrm{nn}_3(\boldsymbol{x}) = \arg\min_{n \in [\mathrm{N}] - \mathrm{nn}_1(\boldsymbol{x}) - \mathrm{nn}_2(\boldsymbol{x})} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2$

The set of K-nearest neighbor

$$\mathrm{knn}(\boldsymbol{x}) = \{\mathrm{nn}_1(\boldsymbol{x}), \mathrm{nn}_2(\boldsymbol{x}), \cdots, \mathrm{nn}_K(\boldsymbol{x})\}$$

Let $\boldsymbol{x}(k) = \boldsymbol{x}_{\mathrm{nn}_k(\boldsymbol{x})}$, then

$$\|\boldsymbol{x} - \boldsymbol{x}(1)\|_2^2 \leq \|\boldsymbol{x} - \boldsymbol{x}(2)\|_2^2 \cdots \leq \|\boldsymbol{x} - \boldsymbol{x}(K)\|_2^2$$

How to classify with K neighbors?

Classification rule

- Every neighbor votes: suppose y_n (the true label) for $\boldsymbol{x}_n$ is c, then
  - vote for c is 1
  - vote for $c' \neq c$ is 0

  We use the *indicator function* $\mathbb{I}(y_n == c)$ to represent.
- Aggregate everyone's vote on a class label c

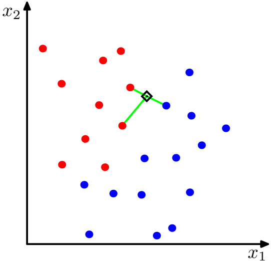
$$v_c = \sum_{n \in \text{knn}(\boldsymbol{x})} \mathbb{I}(y_n == c), \quad \forall \quad c \in [\text{C}]$$

- Label with the majority

$$y = f(\boldsymbol{x}) = \arg\max_{c \in [\text{C}]} v_c$$

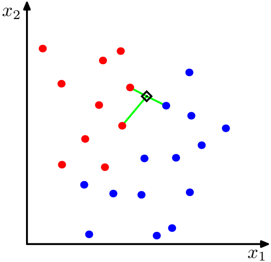
Example

K=1, Label: red



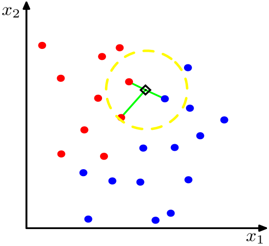
(a)

K=3, Label: red



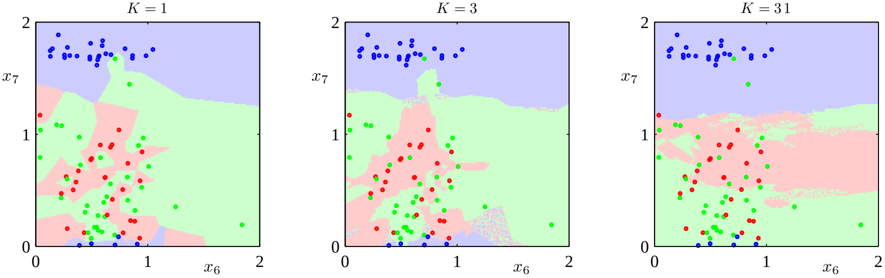
(a)

K=5, Label: blue



(a)

How to choose an optimal K?



When K increases, the decision boundary becomes smooth.

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.

Two versions: which one to use?

- Defined on the training data set

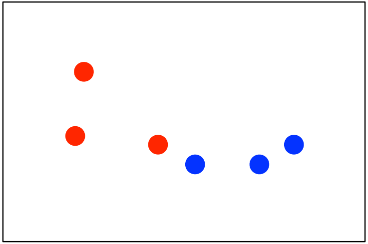
$$A^{\text{TRAIN}} = \frac{1}{\mathsf{N}} \sum_n \mathbb{I}[f(\boldsymbol{x}_n) == y_n], \quad \varepsilon^{\text{TRAIN}} = \frac{1}{\mathsf{N}} \sum_n \mathbb{I}[f(\boldsymbol{x}_n) \neq y_n]$$

- Defined on the test (evaluation) data set

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

Example

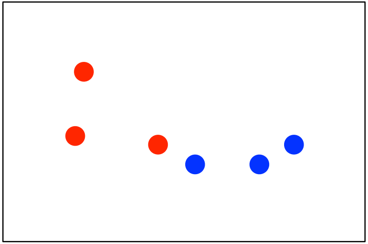
Figure: Training data



What are A^{TRAIN} and $\mathcal{E}^{\mathrm{TRAIN}}$?

Example

Figure: Training data



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

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

Example

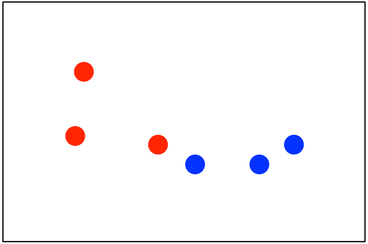


Figure: Training data

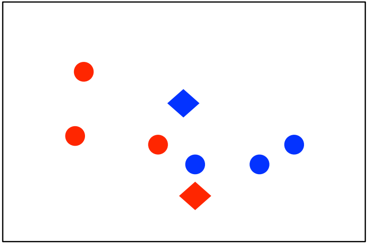


Figure: Test data

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

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

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

Example

Figure: Training data

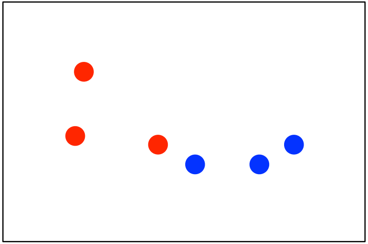
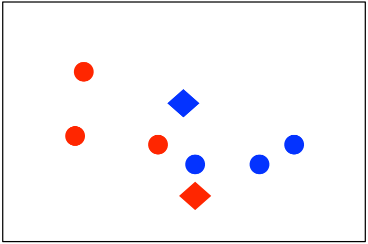


Figure: Test data



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

What are A^{TEST} and $\varepsilon^{\mathrm{TEST}}$?

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

$A^{\mathrm{TEST}} = 0\%, \quad \varepsilon^{\mathrm{TEST}} = 100\%$

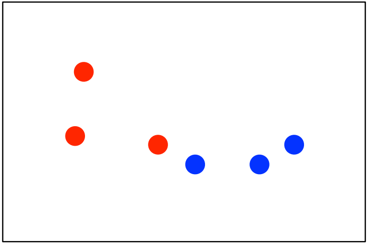
NB. In this case, for every training data point, its nearest neighbor in the training dataset is itself.

Leave-one-out (LOO)

Idea

- For each training instance $\boldsymbol{x}_n$, take it out of the training set and then label it.
- For NNC, $\boldsymbol{x}_n$'s nearest neighbor will *not be itself*. So the error rate would not become 0 necessarily.

Figure: Training data



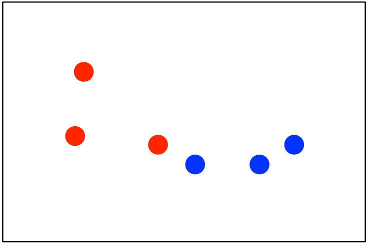
What are the LOO-version of A^{TRAIN} and $\varepsilon^{\mathrm{TRAIN}}$?

Leave-one-out (LOO)

Idea

- For each training instance $\boldsymbol{x}_n$, take it out of the training set and then label it.
- For NNC, $\boldsymbol{x}_n$'s nearest neighbor will *not be itself*. So the error rate would not become 0 necessarily.

Figure: Training data



What are the LOO-version of A^{TRAIN} and $\varepsilon^{\text{TRAIN}}$?

$$A^{\text{TRAIN}} = 66.67\%(\text{i.e.}, 4/6)$$
$$\varepsilon^{\text{TRAIN}} = 33.33\%(\text{i.e.}, 2/6)$$

Hyperparameters in NNC

Two practical issues about NNC

- Choosing K, i.e., the number of nearest neighbors (default is 1)
- Choosing the right distance measure (default is Euclidean distance), for example, from the following generalized distance measure

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

for $p \geq 1$.

Those are not specified by the algorithm itself — resolving them requires empirical validation and are task/dataset-specific.

Tuning by using a validation dataset

Training data (set)

- N samples/instances: $\mathcal{D}^{\text{TRAIN}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_\text{N}, y_\text{N})\}$
- They are used for learning $f(\cdot)$

Test (evaluation) data

- M samples/instances: $\mathcal{D}^{\text{TEST}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_\text{M}, y_\text{M})\}$
- They are used for assessing how well $f(\cdot)$ will do in predicting an unseen $\boldsymbol{x} \notin \mathcal{D}^{\text{TRAIN}}$

Development (or validation) data

- L samples/instances: $\mathcal{D}^{\text{DEV}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_\text{L}, y_\text{L})\}$
- They are used to optimize hyperparameter(s).

Recipe

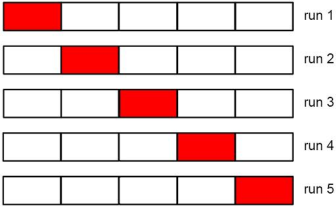
- for each possible value of the hyperparameter (say $K = 1, 3, \cdots, 100$)
  - Train a model using $\mathcal{D}^{\text{TRAIN}}$
  - Evaluate the performance of the model on $\mathcal{D}^{\text{DEV}}$
- Choose the model with the best performance on $\mathcal{D}^{\text{DEV}}$
- Evaluate the model on $\mathcal{D}^{\text{TEST}}$

Cross-validation

What if we do not have validation data?

- We split the training data into S equal parts.
- We use each part *in turn* as a validation dataset and use the others as a training dataset.
- We choose the hyperparameter such that *on average*, the model performing the best

Figure: $S = 5$: 5-fold cross validation



Special case: when $S = N$, this will be leave-one-out.

Recipe

- Split the training data into S equal parts. Denote each part as $\mathcal{D}_s^{\mathrm{TRAIN}}$
- for each possible value of the hyperparameter (say $K = 1, 3, \cdots, 100$)
  - for every $s \in [1, \mathrm{S}]$
    - Train a model using $\mathcal{D}_{\backslash s}^{\mathrm{TRAIN}} = \mathcal{D}^{\mathrm{TRAIN}} - \mathcal{D}_s^{\mathrm{TRAIN}}$
    - Evaluate the performance of the model on $\mathcal{D}_s^{\mathrm{TRAIN}}$
  - Average the S performance metrics
- Choose the hyperparameter corresponding to the best averaged performance
- Use the best hyerparamter to train on a model using all $\mathcal{D}^{\mathrm{TRAIN}}$
- Evaluate the model on $\mathcal{D}^{\mathrm{TEST}}$

Preprocess data

Normalize data so that the data look like from a normal distribution

- Compute the means and standard deviations in each feature

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

- Scale the feature accordingly

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

Many other ways of normalizing data — you would need/want to try different ones and pick them using (cross)validation

Mini-summary

Advantages of NNC

- Computationally, simple and easy to implement – just computing the distance
- Theoretically, has strong guarantees "doing the right thing"

Disadvantages of NNC

- Computationally intensive for large-scale problems: $O(\mathrm{N}D)$ for labeling a (unseen) data point
- We need to "carry" the training data around. Without it, we cannot do classification. This type of method is called *nonparametric*.
- Choosing the right distance measure and K can be involved.