Machine Learning

Prof. Victor Adamchik

U of Southern California

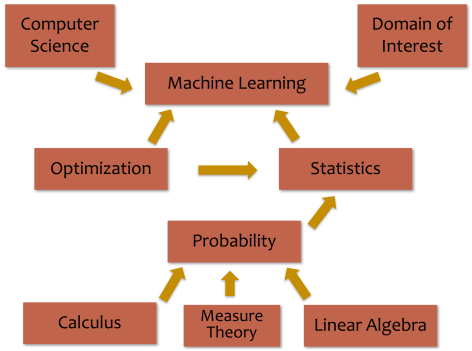
July 1, 2020

Outline

Outline

Required preparation

Students in the class are expected to have a reasonable degree of mathematical sophistication, and to be familiar with the basic knowledge of linear/matrix algebra, multivariate calculus, probability and statistics. Undergraduate classes in these subjects should be sufficient. Programming in Python using necessary packages is optional.



Slides and readings

Lectures Lecture slides will be posted before or soon after class.

Recommended Textbooks:

- Bishop = Pattern Recognition and Machine Learning by C.M. Bishop, 2006
- ESL = The Elements of Statistical Learning by T. Hastie, R. Tibshirani, and J. Friedman, 2008

Teaching philosophy

The nature of this course

- I will NOT teach you how to use TensorFlow and/or PyTorch.
- I will describe the formal properties of models and explain the practical implications.
- I will describe algorithms and their development with intuition and rigor.

Expectation on you

- Grasp abstract concepts and thinking critically to solve problems with machine learning techniques.
- Prepare for advanced machine learning techniques.

Outline

What is machine learning?

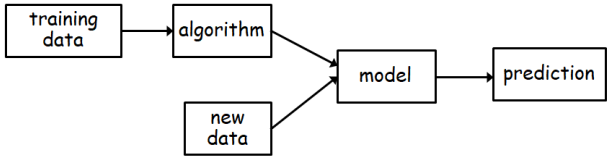
Machine Learning is an umbrella term used to describe a variety of different tools and techniques which allow a machine or a computer program to learn and improve over time. ML is the field of study that gives computers the ability to learn (the way humans do) without being explicitly programmed. We let machines to learn on their own and even to fail so they learn from the failure.

Another definition (Mitchell's book)
A computer program is said to *learn* from experience 'E', with respect to some class of tasks 'T' and performance measure 'P' if its performance at tasks in 'T' as measured by 'P' improves with experience 'E'.

One possible definition (Murphy's book)
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*

Predictive modeling

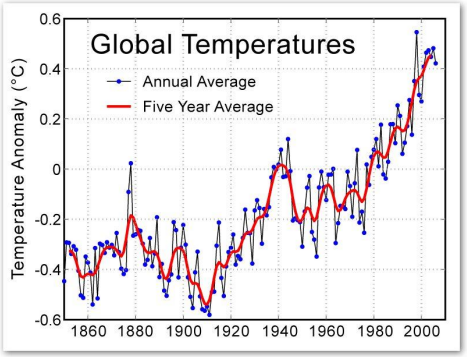
Predictive modeling (i.e. supervised learning) is a process of creating a model using data to make a prediction on new data.



Predictive modeling is a problem of finding a mapping function f from training data ($\boldsymbol{x} \in \mathbb{R}^{\mathrm{D}}$) to output variables.

Example: detect patterns

How the temperature has been changing?

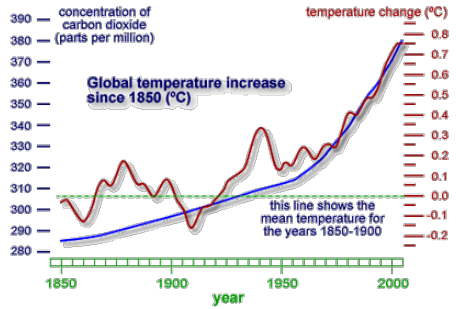


Patterns

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

How do we describe the pattern?

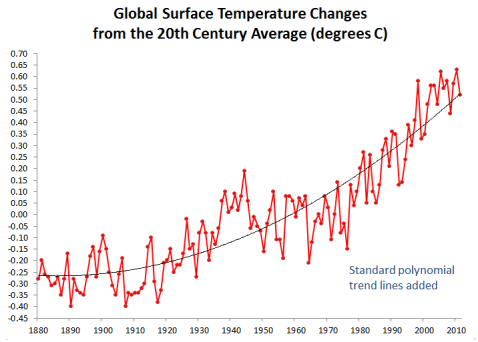
Build a model: fit the data with a polynomial function



- The model is not accurate for individual years
- But collectively, the model captures the major trend
- Still, not able to model the pattern of the *repeated up and down*

Prediction

What is temperature of 2020?



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

What we have learned from this example?

Key ingredients in machine learning

- 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 — "All models are wrong, but some are useful" by George Box.
- Prediction
  apply the model to forecast what is going to happen in future

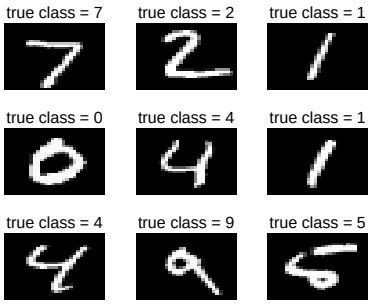
A rich history of applying statistical learning methods

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



Huge success 20 years ago

Handwritten digit recognition (AT&T Labs, late 1990s)



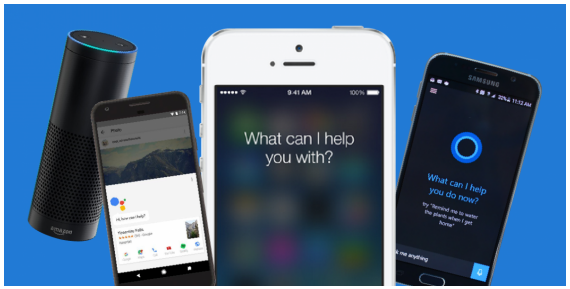
More modern ones

Face Detection and Recognition

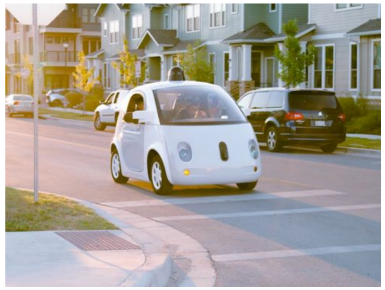
Recommendation Engines

Speech Recognitions

Amazon Alexa, Apple Siri, Microsoft Cortana, ...



Self-Driving Cars



The music streaming platform

Games

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, $\cdots$
- **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 previous examples). Trained by presenting it with inputs and desired outputs
- Unsupervised learning
  There is no feedback from the environment to indicate if the outputs are correct.
- Reinforcement learning
  Aim to act optimally under uncertainty
- Many other paradigms

The focus and goal of this course

- Supervised learning
- Unsupervised learning

Top 10 Algorithms in Machine Learning ...

- k-Nearest Neighbors
- Decision Tree and Random Forests
- Naive Bayes
- Linear and Logistic Regressions
- Artificial Neural Networks
- SVM
- Boosting
- Dimensionality Reduction Algorithms
- Clustering
- Markov Chains

We Live in Exciting Times

ACM (an international computing research society) has named

Yoshua Bengio, Geoffrey Hinton, and Yann LeCun

recipients of the 2018 ACM Turing Award for conceptual and engineering breakthroughs that have made deep neural networks a critical component of computing.

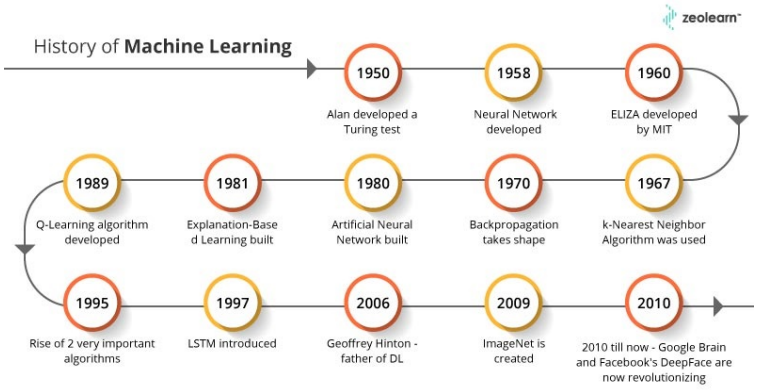
The ACM Turing Award, often referred to as the "Nobel Prize of Computing," carries a $1 million prize.

It is named for Alan M. Turing, the British mathematician who articulated the mathematical foundation and limits of computing.

Outline

How did it Evolve?

Machine Learning is not a single concept but a collection or tools and techniques which have been developed throughout the past 70 years or more.



How did it Evolve?

Year 1950: Turing Test is a game of questions and answers played by a human and a machine (a bot). The goal is to judge whether the machine is a human or otherwise.

Year 1958: Perceptron is a first single-layer Neural Network.

Year 1960: ELIZA is a first Natural Language Processing program.

Year 1967: Nearest Neighbor algorithm.

Year 1970: Backpropagation algorithm for training Neural Networks. Rediscovered in 1986.

Year 1980: First multi-layer feedforward Neural Network (ANN).

How did it Evolve?

Year 1989: Reinforcement Learning. Q-Learning algorithm.

Year 1995: Random Forest Algorithm and Support Vector Machines.

Year 1997: Recurrent Neural Network (RNN).

Year 2006: Geoffrey Hinton coined "Deep Learning"

Year 2009: ImageNet neural network for computer vision.

Year 2010: Google Brain formed.

Year 2012: AlexNet. Revolution in CNN architecture (the depth is essential).

Outline

Reading: ESL chapters 1, 2.1 - 2.3

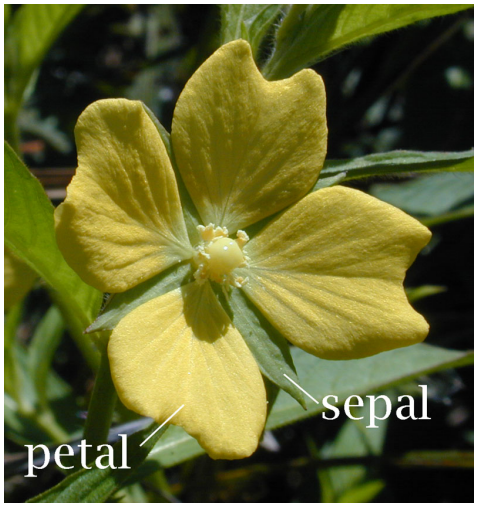
Recognizing flowers

Types of Iris: setosa, versicolor, and virginica



Measuring the properties of the flowers

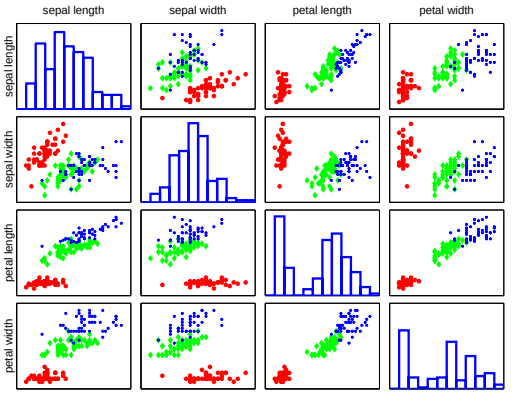
Features and attributes: the widths and lengths of sepal and petal



Pairwise scatter plots of 131 flower specimens

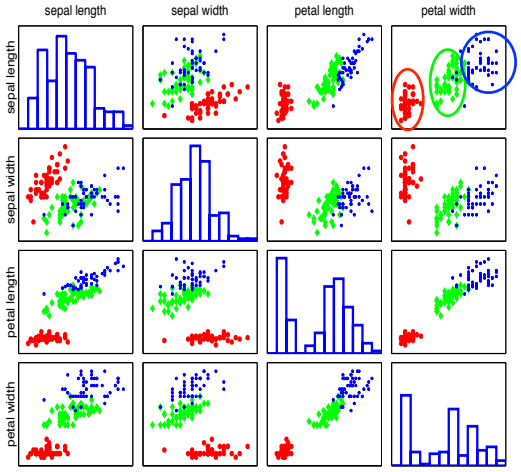
Visualization of data helps identify the right learning model to use

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



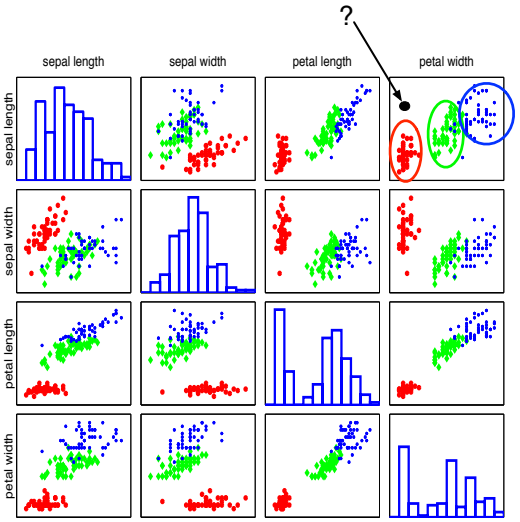
Different types seem well-clustered and separable

Using two features: petal width and sepal length



Labeling an unknown flower type

Closer to red cluster: so labeling it as setosa



General setup for multi-class classification

Training data (set)

- N samples/instances: $\mathcal{D}^{\text{TRAIN}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\mathsf{N}}, y_{\mathsf{N}})\}$
- Each $\boldsymbol{x_n} \in \mathbb{R}^{\mathsf{D}}$ is called a feature vector.
- Each $y_n \in [\mathsf{C}] = \{1, 2, \cdots, \mathsf{C}\}$ is called a label/class/category.
- Our goal is to learn a function $f : \mathbb{R}^{\mathsf{D}} \to [\mathsf{C}]$ for future prediction. Given an unseen observation $\boldsymbol{x}$, $f(\boldsymbol{x})$ can confidently predict the corresponding output y.

Special case: binary classification

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

Often, data is conveniently 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

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

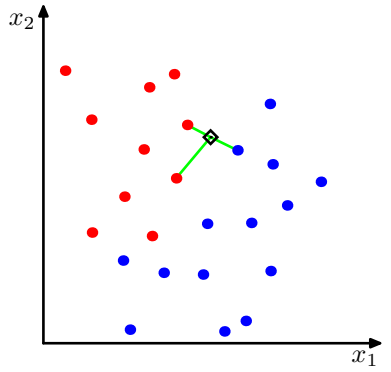
where $\mathsf{nn}(\boldsymbol{x}) \in [\mathsf{N}] = \{1, 2, \cdots, \mathsf{N}\}$, i.e., the index to one of the training instances,

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

where argmin means finding the argument (in this case n) that minimizes the function/expression, and $\|\cdot\|_2$ is the L_2/Euclidean distance between two points.

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	7.1	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 (i.e. $\boldsymbol{x} = (1.8, 6.4)$)

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

ID	distance
1	1.75
2	0.72
3	0.76

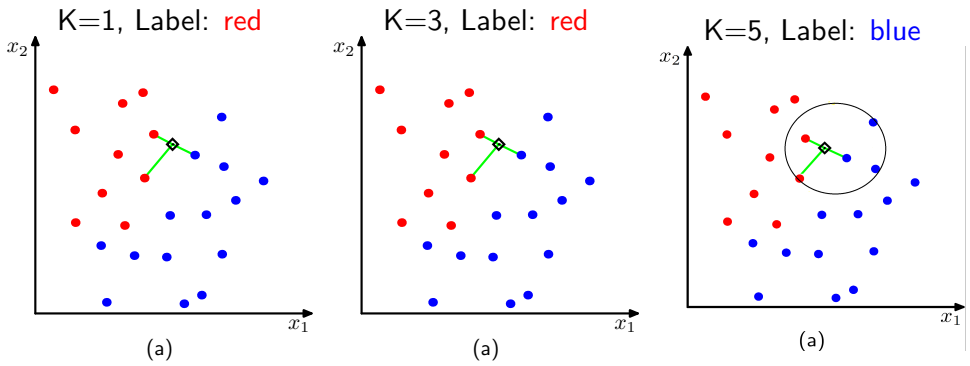
Thus, the category is *versicolor*.

KNN classifier steps

Each test sample is classified using majority vote of its neighbors by the following steps:

- Distances from the test sample to all stored training sample are calculated using a specific distance function or similarity measure.
- The K nearest neighbors of the test sample are selected, where K is a pre-defined small integer.
- The most repeated class of these K neighbors is assigned to the test sample.

Example



Is NNC doing the right thing for us?

Intuition

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

Defined on the training data set

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

where $\mathbb{I}[\cdot]$ is the indicator function: $\mathbb{I}[true] = 1$ and $\mathbb{I}[false] = 0$.

Is this the right measure?

Test Data

We should care about accuracy when predicting *unseen data*.

Test Data

- $\mathcal{D}^{\text{TEST}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\mathsf{M}}, y_{\mathsf{M}})\}$
- A fresh dataset, it does *not* overlap with Training Data.
- Test accuracy and test error

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

- Good measurement of a classifier's performance

Variant 1: measure nearness with other distances

Previously, we use the Euclidean distance

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

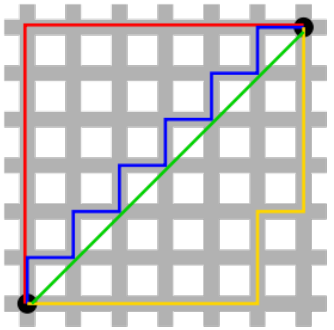
Many other alternative distances

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

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

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

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



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

The distance function

The distance function between two vectors x and y is a function $d(x, y)$ that defines the distance between both vectors as a non-negative real number. This function satisfies the following properties:

- $d(x, y) \geq 0$
- $d(x, y) = 0$ if and only if $x = y$
- $d(x, y) = d(y, x)$
- $d(x, y) \leq d(x, z) + d(z, y)$

Variant 2: K-nearest neighbor (KNN)

Increase the number of nearest neighbors to use?

- 1-nearest neighbor: $\mathsf{nn}_1(\boldsymbol{x}) = \operatorname{argmin}_{n \in [\mathsf{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2$
- 2-nearest neighbor: $\mathsf{nn}_2(\boldsymbol{x}) = \operatorname{argmin}_{n \in [\mathsf{N}] - \mathsf{nn}_1(\boldsymbol{x})} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2$
- 3-nearest neighbor: $\mathsf{nn}_3(\boldsymbol{x}) = \operatorname{argmin}_{n \in [\mathsf{N}] - \mathsf{nn}_1(\boldsymbol{x}) - \mathsf{nn}_2(\boldsymbol{x})} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2$

The set of K-nearest neighbors

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

Note: with $\boldsymbol{x}(k) = \boldsymbol{x}_{\mathsf{nn}_k(\boldsymbol{x})}$, we have

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

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

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

- Scale the feature accordingly (it looks more like the Gaussian distribution)

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

Many other ways of normalizing data.

Which variants should we use?

Hyperparameters in NNC

- The distance measure (or similarity measure)
- K (i.e. how many nearest neighbors?)
- Different ways of preprocessing

Most algorithms have hyperparameters. Tuning them is a significant part of applying an algorithm, the most time-consuming part.

Tuning hyperparameters

Training data

- 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 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 our model $f(\cdot)$ will do.

Development/Validation data

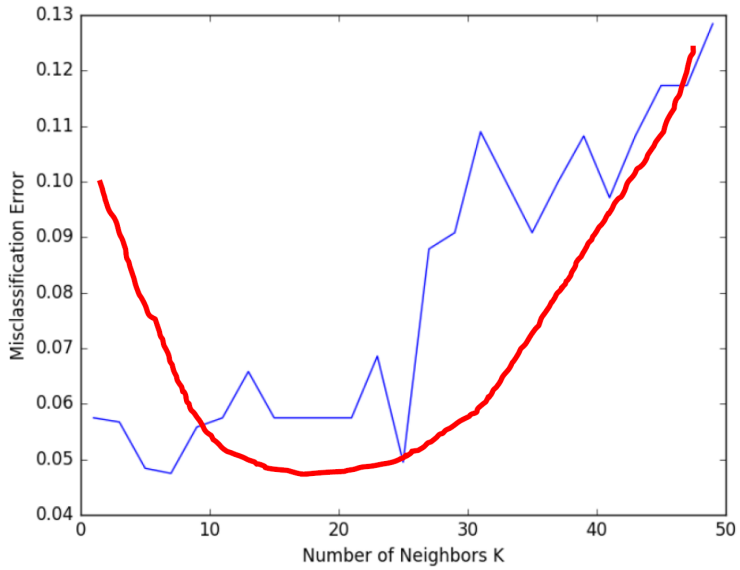
- L samples/instances: $\mathcal{D}^{\text{DEV}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\mathsf{L}}, y_{\mathsf{L}})\}$
- They are used to optimize hyperparameter(s).

These three sets should *not* overlap!

Choosing K

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

Choosing K

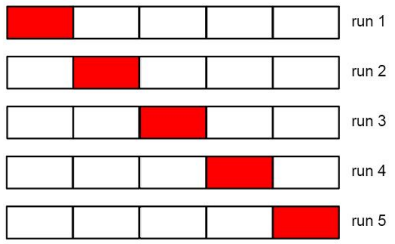


S-fold Cross-validation

What if we do not have a development set?

- Split the training data into S equal parts.
- We will run the algorithms S times.
- For each run we take the red block as a development dataset and use the others as a training dataset.
- Choose the hyperparameter with the best *average* performance.

S = 5: 5-fold cross validation



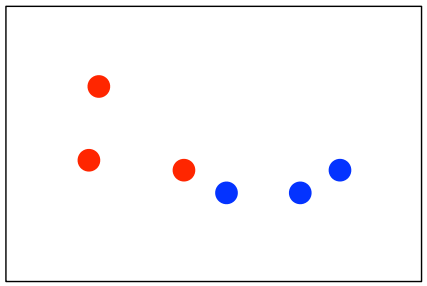
Special case: S = N, called *leave-one-out*.

Leave-one-out (LOO)

Idea

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

Training data



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

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

Underfitting and Overfitting

Underfitting means the model does not fit, in other words, does not predict, the training data very well.

Underfitting implies high bias and low variance.

Bias is the difference between the true label and our prediction.

The variance is an error from sensitivity to small fluctuations in the training set.

When we increase the model complexity, bias is reduced and variance is increased. This eventually leads to overfitting.

Overfitting means that the model predict the training data too well.

Overfitting implies low bias but high variance.

Choosing K

How do the bias and variance relate to KNN?

Consider an extreme case, $K = 1$.

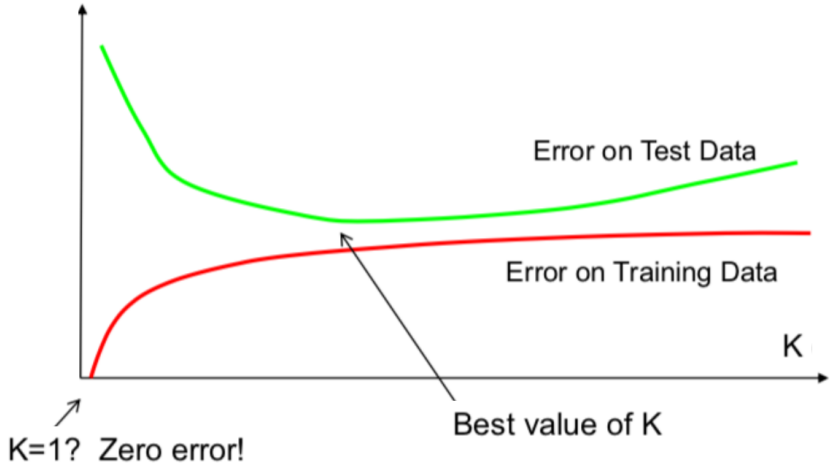
The bias is zero: the training data is perfectly predicted.

However, when it comes to new data there is a higher chance of an error.

This causes high variance.

When we increase K, training error will increase (increase bias), but the test error may decrease (decrease variance).

Choosing K



Mini-summary

- The K-nearest neighbor algorithm essentially boils down to forming a majority vote between the K most similar instances to a given "unseen" observation.
- KNN classifier performs the following two steps:
  - It runs through the whole dataset computing distances between x and each training observation.
  - It chooses K points in the training data that are closest to x.
  - It computes a majority vote.

Advantages of NNC

Simple, easy to implement, wildly used in practice.

7291 train points, 2007 test points. Error rates:

- Neural Network: 0.049
- 1-NN/Euclidean distance: 0.055
- 1-NN/tangent distance: 0.026

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 *non-parametric*.
- Choosing the right hyperparameters could be laborious.

Mini-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 hyperparameters.
- Evaluate using the test data and report performance.
- Use the model to predict future/make decisions.

Mini-summary

Supervised learning is a process of creating a model using data (labeled with desired outputs) to make a prediction on new data.