CSCI 566: Deep Learning and its Applications

Yue Zhao
Thomas Lord Department of Computer Science
University of Southern California

Credits to previous versions of USC CSCI566, CMU 10601/701, Stanford CS 229, 231n

Welcome to CSCI 566!

This class will teach you some exciting developments in Machine Learning, Computer Vision, NLP, Robotics, and other AI-related fields in the last decade!

New cool stuff: I will aim to bring someone to lectures from industry, academia, and more to discuss their life in ML and Data Science – this session will be interactive, and I hope it will be helpful for your career planning.

~~Prerequisite~~ (CSCI 567 – Machine Learning)

Do you know the following..?

- Probability and Statistical Learning
    - Density function, loss function, cross-validation
- Supervised Learning
    - Nearest Neighbor, Kernels, Random Forest
- Unsupervised Learning
    - Clustering, PCA, SVD

Ideally yes, but this semester is a bit tricky as we are transitioning

Prerequisite

ML course transitioning (refactored and "swapped"; still undecided):

- **Spring 2024**: 566 is for deep learning (DL), and 567 is for classical ML
- **Fall 2024**: CSCI 566 will be classical ML, and 567 will cover more DL

Thus, the format for this version will be unique – think twice:

- 35% classical ML
- 15% real-world applications and career discussion
- 50% deep learning
- It will be the ideal one if you have take 567 before or does not plan to take 567 in the future

Prerequisite

This course is still evolving, and we are trying new things this time. The syllabus, course policy, and grading details may change over the semester (check the course website!)

https://usc-asap.github.io/CSCI566-S24/

https://piazza.com/usc/spring2024/csci566

Note: this is a transitional course, which means its deep learning portion will be reduced to make the transition

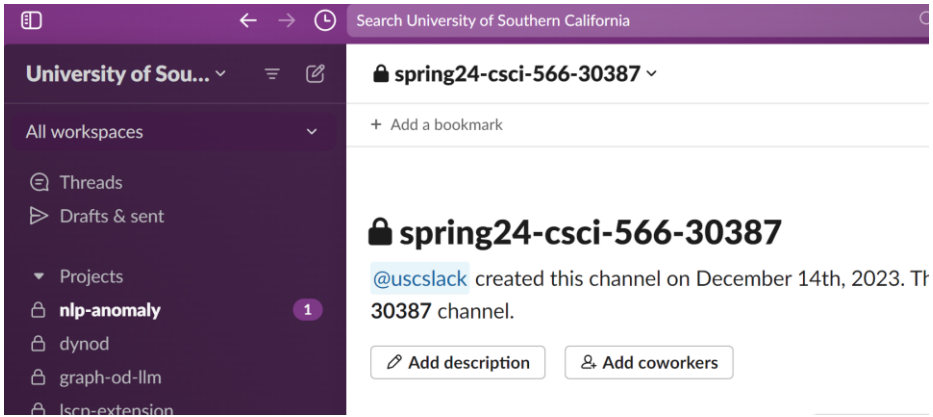
CSCI 566 and 567 share some common topics due to the transition.

Piazza and Gradescope

If you are newly added to the course, please add yourself to:

- **Gradescope** with the Entry Code: **ZW8KJ8**
- **Piazza:** https://piazza.com/usc/spring2024/csci566

We also have a slack channel:



Discussions on The Depth of the Course

Actions

Redundant Course Content

Hi everyone,

I'm feeling a bit frustrated that we are covering so many of the same topics in this deep learning class that we already learned in depth in the machine learning course in previous semesters. As masters students who have already taken an ML course (CSCI 567), I was hoping this course would help me gain more advanced DL knowledge. But half the syllabus is just re-teaching neural networks, CNNs, RNNs, etc.

I get that some review is needed since DL builds on ML concepts. But it feels like we're just repeating a huge chunk of fundamental content rather than getting to dive deeper. We already know how to train basic NNs with backprop, implement CNNs, understand transformer architectures, etc. Spending weeks re-learning this feels redundant when our time could be better spent on more cutting edge DL models, architectures and applications.

The professor said this is meant to be an introductory level overview of DL. But for most students in this class, it's not introducing anything new. We want to get more value from this course and our tuition dollars. As grad students trying to become experts in DL, we were hoping to get exposure to more advanced techniques that would better prepare us for research or industry jobs.

It's frustrating to take precious credits on essentially a retread of previous material. We'd appreciate if the prof could trim back the introductory weeks and spend more time helping us build our DL skills beyond basic ML concepts we already have down pat. We want this class to feel like it's deepening our DL knowledge, not just repeating our old ML homework.

followup discussions, *for lingering questions and comments*

Resolved Unresolved @11_f1

Actions

[redacted] ay ago

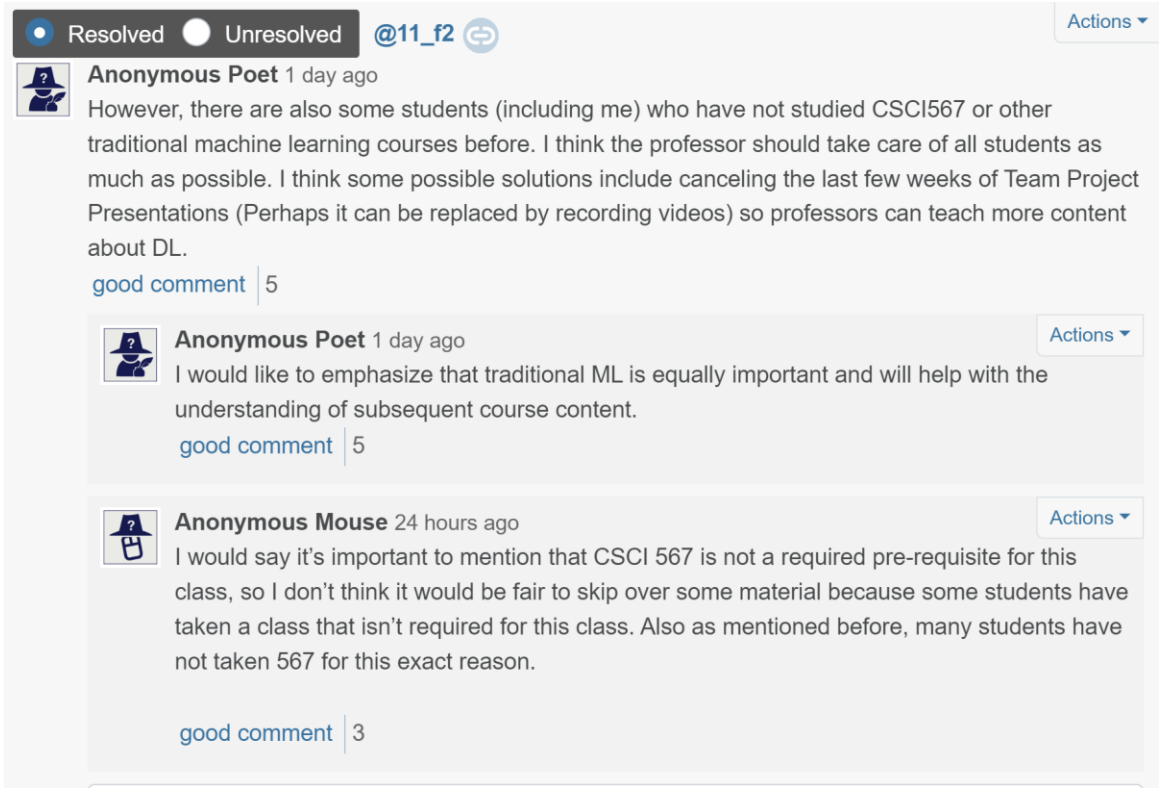
Very well stated. Dear teaching staff please take this issue in consideration. Especially we are in a very challenging era of industry and we need more advanced expertise for our future. Thanks

helpful! 0

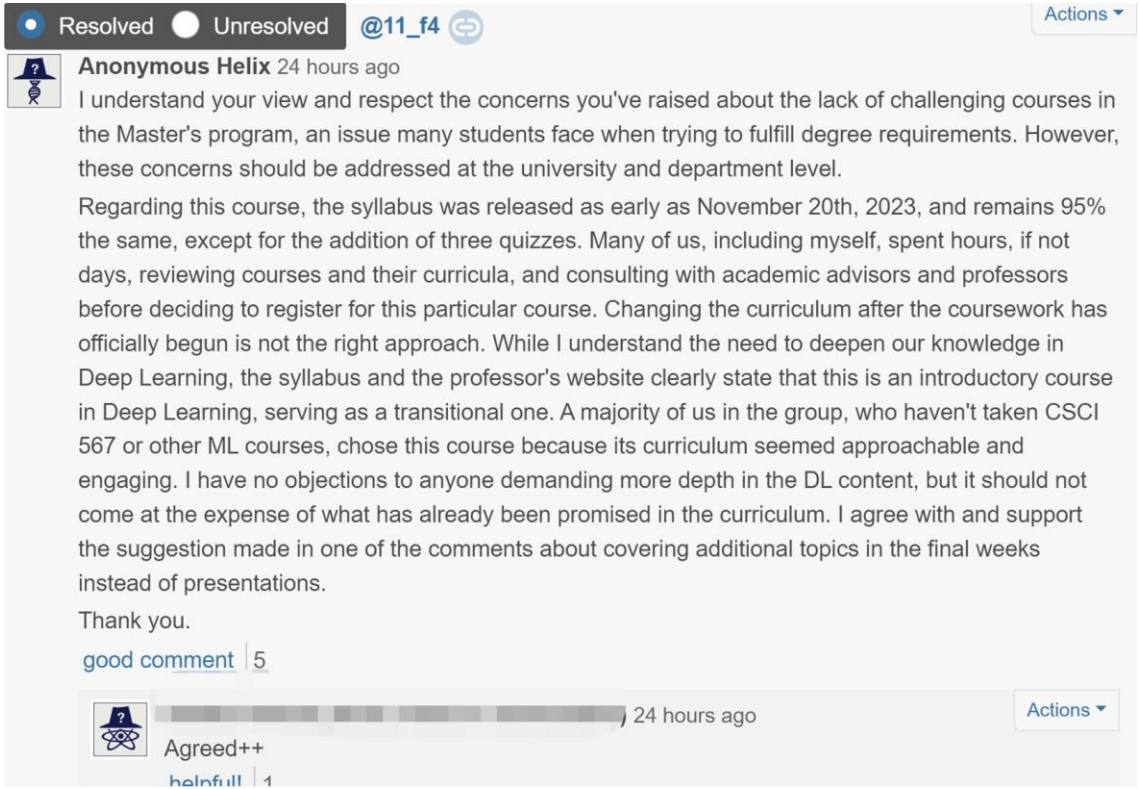
Reply to this followup discussion

USC Viterbi School of Engineering

Discussions on The Depth of the Course



Discussions on The Depth of the Course



Discussions on The Depth of the Course

the instructors' answer, *where instructors collectively construct a single answer*

First of all, thanks for sharing your thoughts -- which are always acknowledged and appreciated.

Second, I also agree with most of the posts saying that since 567 is not a prerequisite this semester, it will be *risky* to assume all the students have the same level of knowledge in classical/non-deep ML. Thus, we will still have to take this into account.

Again, I think the solution can be multi-fold:

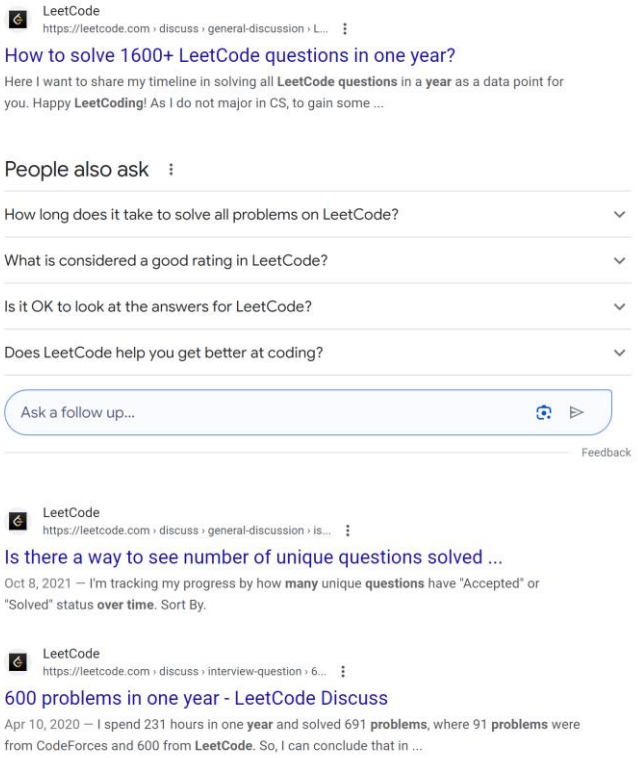
1. we will consider covering more advanced topics in the later part of the course
2. feel free to sign up for research-based projects (to be posted by TA), which can be open-ended, challenging, and rewarding
3. by design, 566 is an intro course for MS students -- you may consider more specialized courses if preferred. Let me name a few:

CSCI 655: Applied NLP
CSCI 599: 3D Vision
CSCI 599: Autonomous DecisionMaking

Discussions on The Depth of the Course

The correlation of getting a job and the course difficulty:

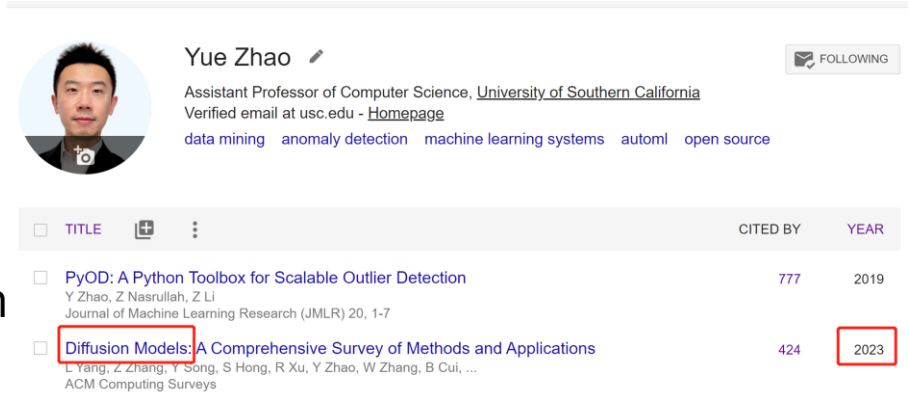
- A challenging course does not necessarily get you a better job
    - It will eat more of your time on **leetcode** and **networking** on advanced but less useful topics



Discussions on The Depth of the Course

The correlation of getting a job and the course difficulty:

- A challenging course does not necessarily get you a better job
  - It will eat more of your time on **leetcode** and **networking** on advanced but less useful topics
  - Advanced topics change so fast

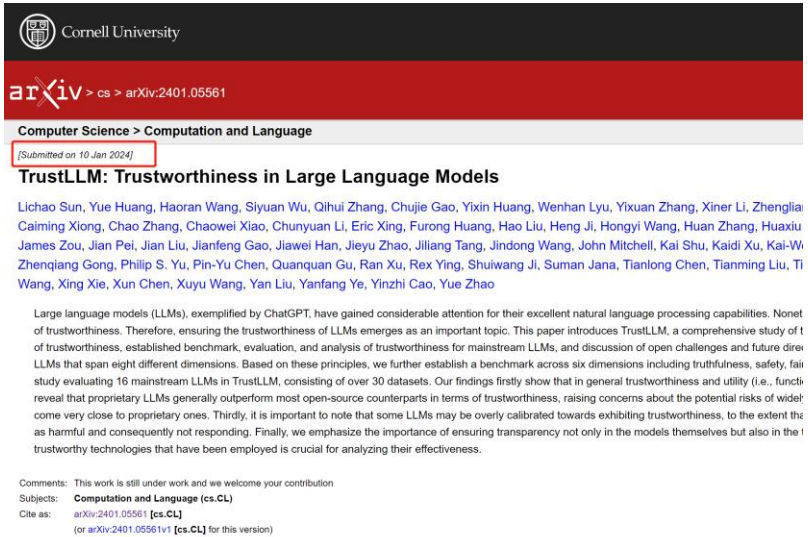


USC Viterbi School of Engineering

Discussions on The Depth of the Course

The correlation of getting a job and the course difficulty:

- A challenging course does not necessarily get you a better job
  - It will eat more of your time on **leetcode** and **networking** on advanced but less useful topics
  - Advanced topics change so fast



Cornell University

arXiv > cs > arXiv:2401.05561

Computer Science > Computation and Language

[Submitted on 10 Jan 2024]

TrustLLM: Trustworthiness in Large Language Models

Lichao Sun, Yue Huang, Haoran Wang, Siyuan Wu, Qihui Zhang, Chujie Gao, Yixin Huang, Wenhan Lyu, Yixuan Zhang, Xiner Li, Zhengliang
Caiming Xiong, Chao Zhang, Chaowei Xiao, Chunyuan Li, Eric Xing, Furong Huang, Hao Liu, Heng Ji, Hongyi Wang, Huan Zhang, Huaxiu
James Zou, Jian Pei, Jian Liu, Jianfeng Gao, Jiawei Han, Jieyu Zhao, Jiliang Tang, Jindong Wang, John Mitchell, Kai Shu, Kaidi Xu, Kai-W
Zhenqiang Gong, Philip S. Yu, Pin-Yu Chen, Quanquan Gu, Ran Xu, Rex Ying, Shuiwang Ji, Suman Jana, Tianlong Chen, Tianming Liu, Ti
Wang, Xing Xie, Xun Chen, Xuyu Wang, Yan Liu, Yanfang Ye, Yinzhi Cao, Yue Zhao

Large language models (LLMs), exemplified by ChatGPT, have gained considerable attention for their excellent natural language processing capabilities. Nonet
of trustworthiness. Therefore, ensuring the trustworthiness of LLMs emerges as an important topic. This paper introduces TrustLLM, a comprehensive study of t
of trustworthiness, established benchmark, evaluation, and analysis of trustworthiness for mainstream LLMs, and discussion of open challenges and future direc
LLMs that span eight different dimensions. Based on these principles, we further establish a benchmark across six dimensions including truthfulness, safety, fair
study evaluating 16 mainstream LLMs in TrustLLM, consisting of over 30 datasets. Our findings firstly show that in general trustworthiness and utility (i.e., functi
reveal that proprietary LLMs generally outperform most open-source counterparts in terms of trustworthiness, raising concerns about the potential risks of widel
come very close to proprietary ones. Thirdly, it is important to note that some LLMs may be overly calibrated towards exhibiting trustworthiness, to the extent tha
as harmful and consequently not responding. Finally, we emphasize the importance of ensuring transparency not only in the models themselves but also in the t
trustworthy technologies that have been employed is crucial for analyzing their effectiveness.

Comments: This work is still under work and we welcome your contribution
Subjects: Computation and Language (cs.CL)
Cite as: arXiv:2401.05561 [cs.CL]
(or arXiv:2401.05561v1 [cs.CL] for this version)

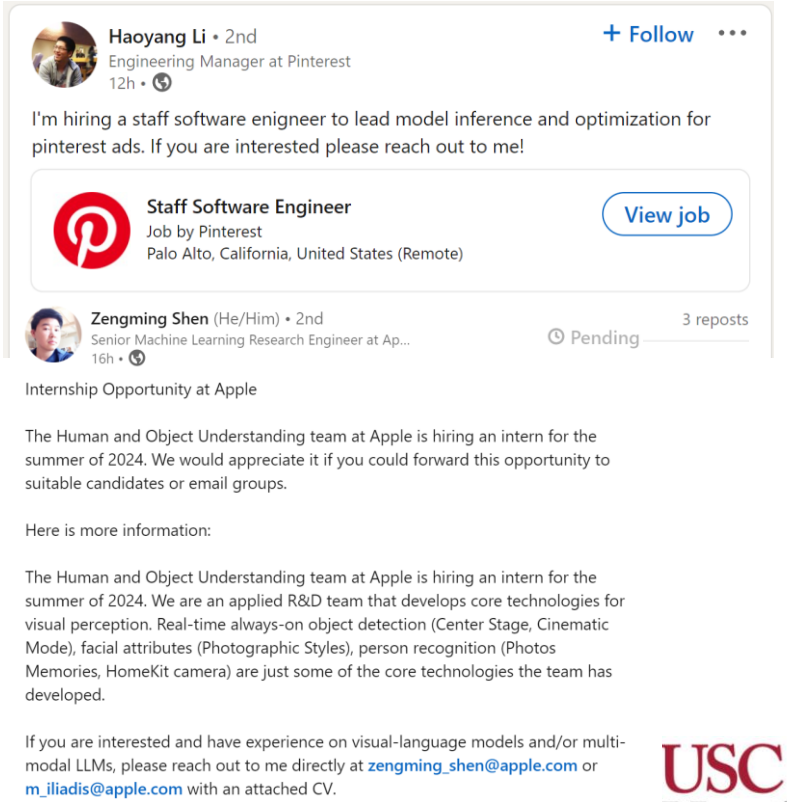
Discussions on The Depth of the Course

How to use LinkedIn more effectively?

1. Identify the post
2. Reach out the person (many times they leave the email)
3. Submit the application

Differentiate yourself:

1. Degree (hmmmm)
2. Unique skillsets, e.g., open-source developers
3. Perfect match to the job, e.g., papers



Haoyang Li • 2nd
Engineering Manager at Pinterest
12h •
+ Follow

I'm hiring a staff software enigneer to lead model inference and optimization for pinterest ads. If you are interested please reach out to me!

Staff Software Engineer
Job by Pinterest
Palo Alto, California, United States (Remote)
View job

Zengming Shen (He/Him) • 2nd
Senior Machine Learning Research Engineer at Ap...
16h •
Pending
3 reposts

Internship Opportunity at Apple

The Human and Object Understanding team at Apple is hiring an intern for the summer of 2024. We would appreciate it if you could forward this opportunity to suitable candidates or email groups.

Here is more information:

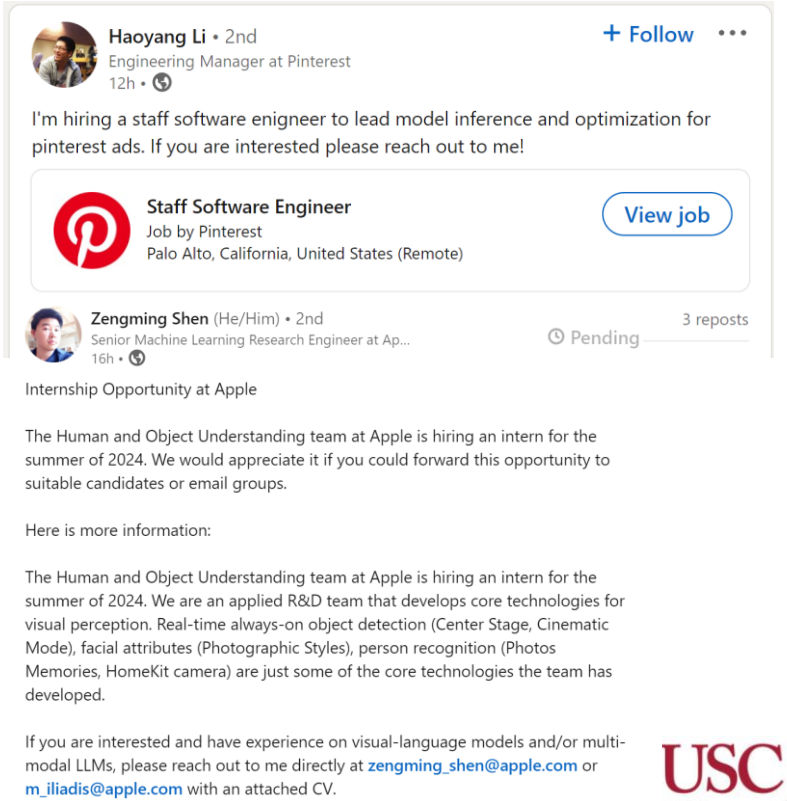
The Human and Object Understanding team at Apple is hiring an intern for the summer of 2024. We are an applied R&D team that develops core technologies for visual perception. Real-time always-on object detection (Center Stage, Cinematic Mode), facial attributes (Photographic Styles), person recognition (Photos Memories, HomeKit camera) are just some of the core technologies the team has developed.

If you are interested and have experience on visual-language models and/or multi-modal LLMs, please reach out to me directly at zengming_shen@apple.com or m_iliadis@apple.com with an attached CV.

Discussions on The Depth of the Course

How to use LinkedIn more effectively?

1. Do not spend too much time on any single position – they can be fake
2. Apply to the job which may not necessarily fit (who knows)
3. Put some emphasis on non-tech industries, e.g., healthcare, mining, finance.



Haoyang Li • 2nd
Engineering Manager at Pinterest
12h •
+ Follow

I'm hiring a staff software enigneer to lead model inference and optimization for pinterest ads. If you are interested please reach out to me!

Staff Software Engineer
Job by Pinterest
Palo Alto, California, United States (Remote)
View job

Zengming Shen (He/Him) • 2nd
Senior Machine Learning Research Engineer at Ap...
16h •
Pending
3 reposts

Internship Opportunity at Apple

The Human and Object Understanding team at Apple is hiring an intern for the summer of 2024. We would appreciate it if you could forward this opportunity to suitable candidates or email groups.

Here is more information:

The Human and Object Understanding team at Apple is hiring an intern for the summer of 2024. We are an applied R&D team that develops core technologies for visual perception. Real-time always-on object detection (Center Stage, Cinematic Mode), facial attributes (Photographic Styles), person recognition (Photos Memories, HomeKit camera) are just some of the core technologies the team has developed.

If you are interested and have experience on visual-language models and/or multi-modal LLMs, please reach out to me directly at zengming_shen@apple.com or m_iliadis@apple.com with an attached CV.

Diversify Your Skillset (Personal Story)

Yue Zhao
yzhao062

Assistant Professor of Computer Science @ University of Southern California (USC); Ph.D. @ CMU. Anomaly/Outlier Detection | ML Systems | AutoML

Edit profile

Sponsors dashboard

3.5k followers · 22 following

University of Southern California
Los Angeles, CA, USA
https://viterbi-web.usc.edu/~yzhao010/
@yzhao062
in/yzhao062

Pinned — Customize your pins

pyod Public
A Comprehensive and Scalable Python Library for Outlier Detection (Anomaly Detection)
Python 7.7k 1.3k

anomaly-detection-resources Public
Anomaly detection related books, papers, videos, and toolboxes
Python 7.7k 1.7k

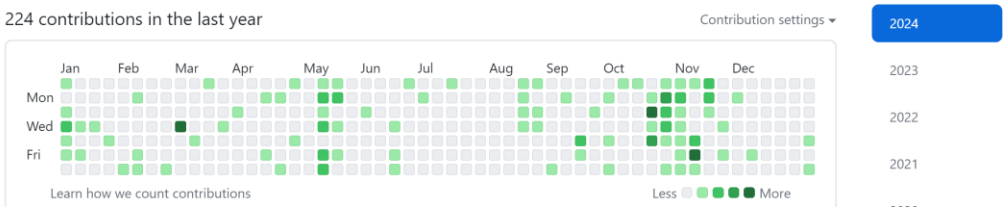
WSAD Public
A Collection of Resources for Weakly-supervised Anomaly Detection (WSAD)
Python 114 5

Minqi824/ADBench Public
Official Implement of "ADBench: Anomaly Detection Benchmark", NeurIPS 2023.
Python 717 120

pytod Public
TOD: GPU-accelerated Outlier Detection via Tensor Operations
Python 162 22

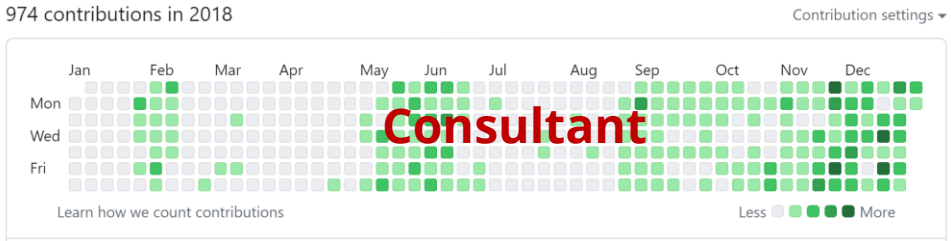
pygod-team/pygod Public
A Python Library for Graph Outlier Detection (Anomaly Detection)
Python 1.1k 119

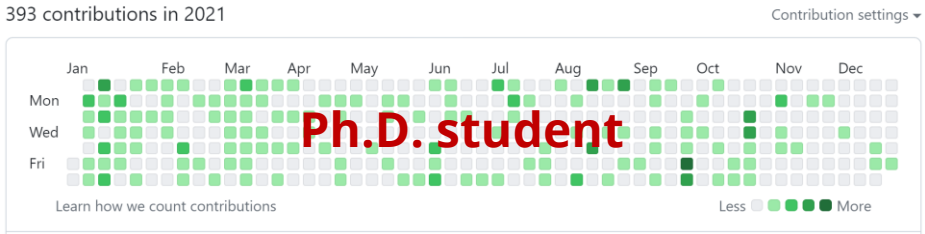
224 contributions in the last year — Contribution settings

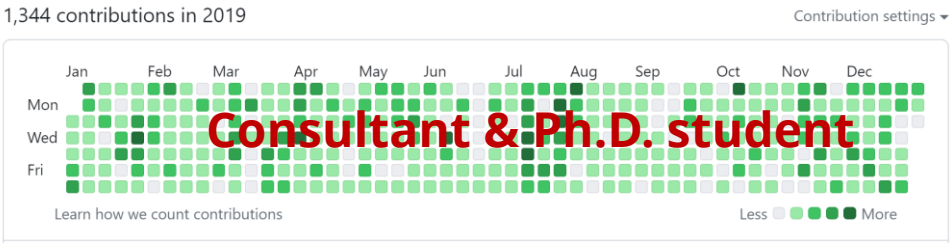


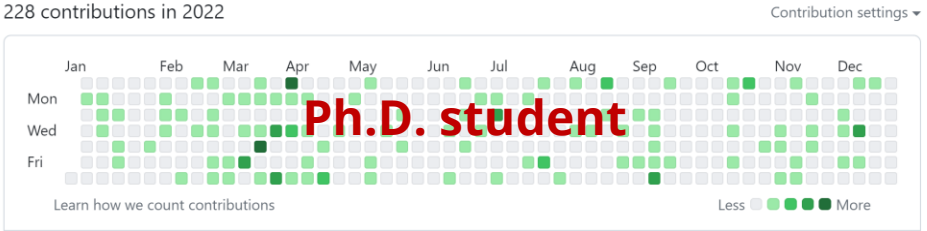
USC Viterbi School of Engineering

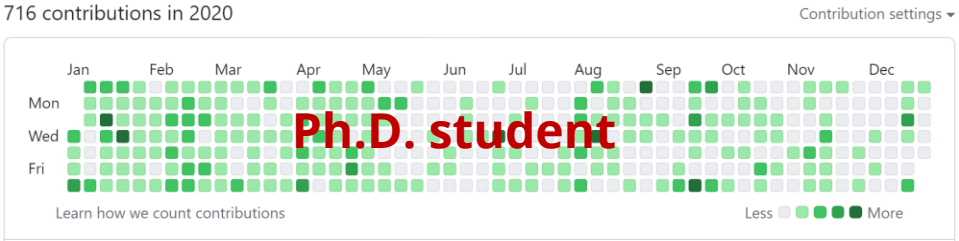
Diversify Your Skillset

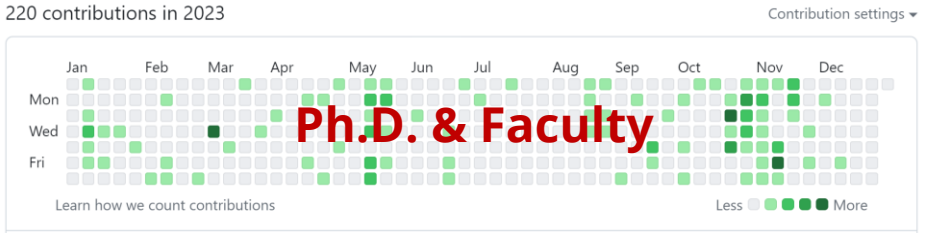












Diversify Your Skillset

Gitstar Ranking Users Organizations Repositories GitHub username Search yzhao062

yzhao062

Star ★ 17704
Rank 727

Go to GitHub

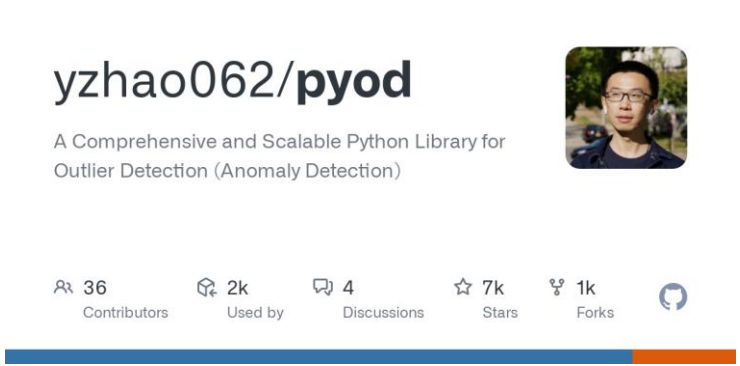
Fetched on 2024/01/14 20:49

Up to date

30 Repositories

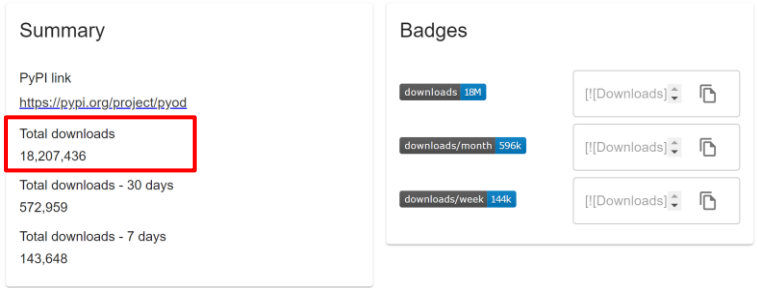
pyod	★ 7741
anomaly-detection-resources	★ 7741
combo	★ 633
SUOD	★ 364
data-mining-conferences	★ 313
awesome-ensemble-learning	★ 270
pytod	★ 162
MetaOD	★ 154

Deployed Systems and Real-world Applications



PyOD library got **18M downloads**:

- **Morgan Stanley** for risk modeling
- **Tesla** for manufacturing quality control
- **NASA** for supply chain management
- **Microsoft** for security breach detection



Morgan Stanley

TESLA

NASA

Microsoft

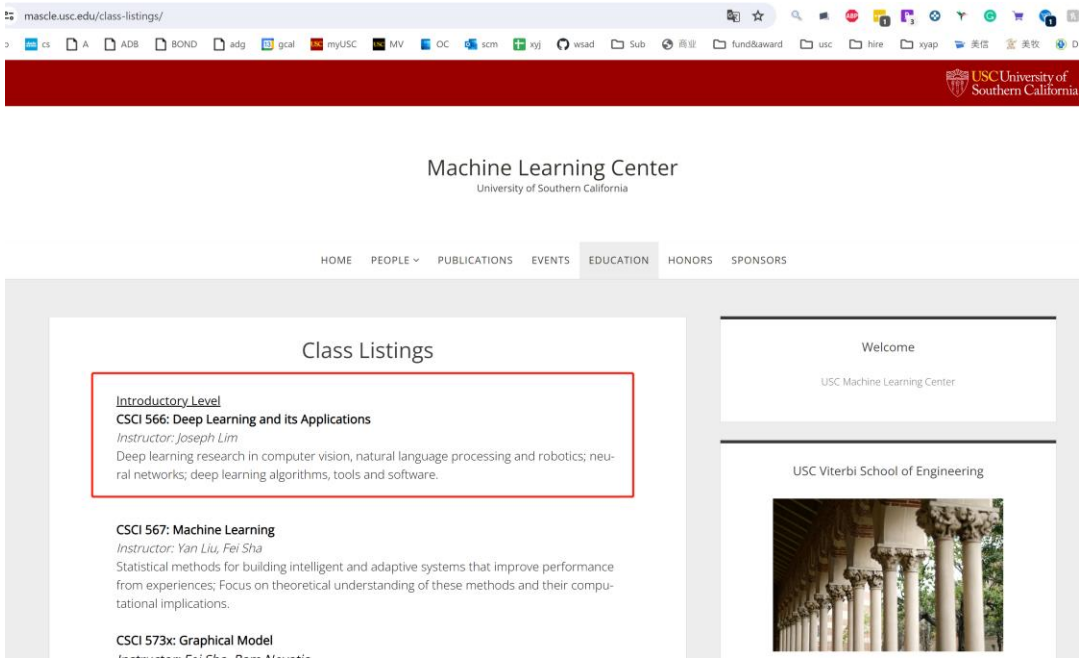
The Goal & Value Proposition of This Course

Have an understanding of deep learning topics:

- Understand when and where to use them

Knowing better about the industry:

- How to find a better job or career planning

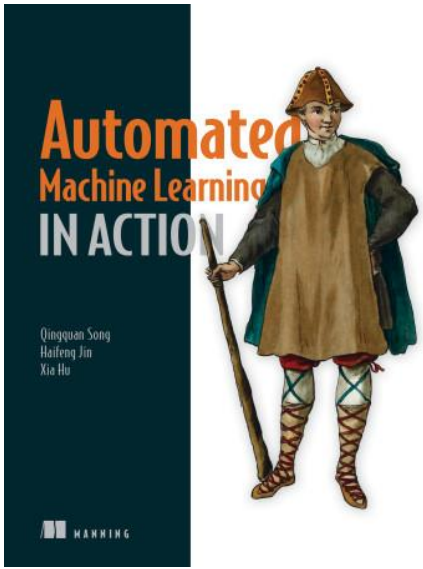


USC Viterbi School of Engineering

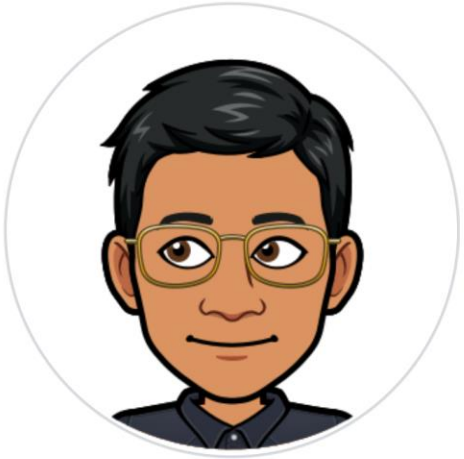
How to Start Building ML Open-Source?

Guest lecture & Presenter: Dr. Haifeng Jin

March 1st



Haifeng Jin is a software engineer on the Keras team at Google. He is the creator of **AutoKeras**, coauthor of **Keras Tuner**, and a contributor to **Keras** and **TensorFlow**. Haifeng got his Ph.D. in computer science at Texas A&M University. His research interest is automated machine learning (AutoML).



Haifeng Jin

haifeng-jin · he/him

Unfollow

Software engineer on Keras | Project lead of AutoKeras & KerasTuner

690 followers · 143 following

Followed by yifanjiang19

@keras-team

https://haifengjin.com/about

Pinned

keras-team/autokeras Public

AutoML library for deep learning

Python 9k 1.4k

keras-team/keras-tuner Public

A Hyperparameter Tuning Library for Keras

Python 2.8k 385

datamllab/automl-in-action-notebooks Public

Jupyter notebooks for the code samples of the book "Automated Machine Learning in Action"

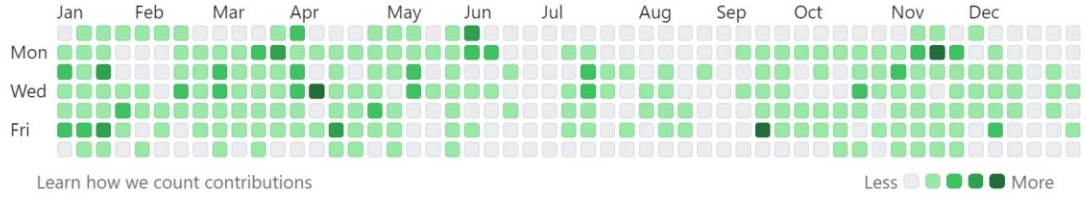
Jupyter Notebook 77 37

keras-team/keras Public

Deep Learning for humans

Python 60.2k 19.5k

956 contributions in the last year



Learn how we count contributions

Less More

2024

2023

2022

2021

2020

2019

2018

2017

@keras-team @tensorflow @conda-forge More

Activity overview

Contributed to keras-team/keras-tuner, keras-team/keras, keras-team/keras-core

15%
Code review

Teaching Team (Finalized!)

Note: alphabetical by last names. **See up-to-date location/zoom** Piazza post.

Instructor

Yue Zhao

OH: Fridays morning (not for technical questions but admin/logistics/etc.)

Teaching Assistant

Varun Bhatt

OH: Tuesdays 14:00-15:00; Details at Piazza post

Teaching Assistant

Zihao He

OH: Thursdays 14:00-15:00; Details at Piazza post

Teaching Assistant

Zihao Hu

OH: Wednesdays 10:00-11:00; Details at Piazza post

Teaching Assistant

Ayush Jain

OH: Wednesdays 14:00-15:00; Details at Piazza post

Teaching Assistant

Ziyi Liu

OH: Thursdays 10:00-11:00; Details at Piazza post

Teaching Assistant

Yuehan Qin

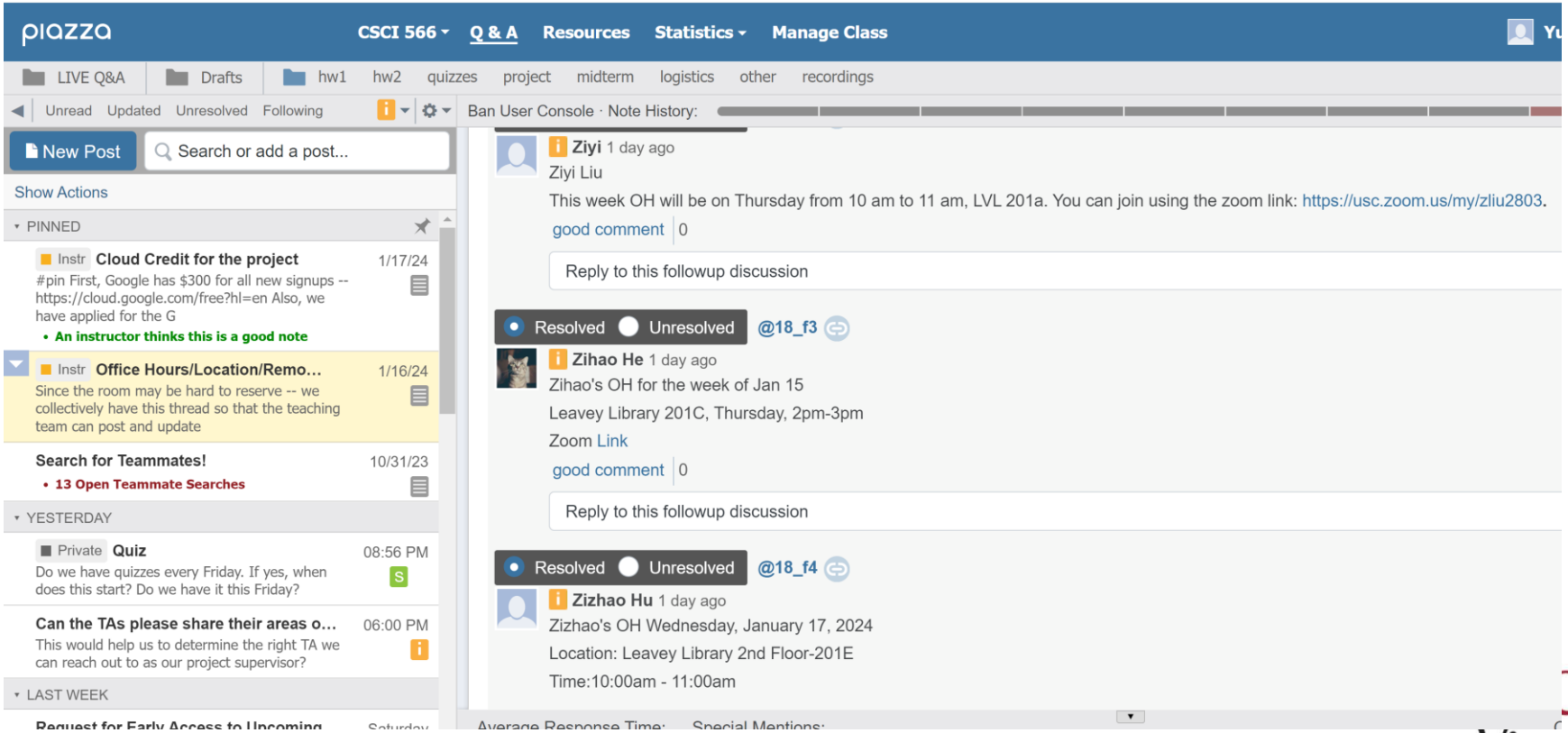
OH: Tuesdays 10:00-11:00; Details at Piazza post

Teaching Assistant

Pengda Xiang

OH: Tuesdays 15:30-16:30; Details at Piazza post

Office Hours (up-to-date on Piazza)



Logistics

Basic logistics:

- **Syllabus:** Syllabus (USC login required; may not be updated as frequent as the website)
- **Time:** Fridays, 1:00pm-4:20pm PST
- **Location:** THH 201
- **Discussion:** Piazza
- **Contact:** Students should use Piazza for any course-related questions. For external inquiries, personal matters, or emergencies, you can email the CSCI 566 staff at usc.csci566@gmail.com. Please do *not* email any of the CSCI 566 staff individually.
- **Guest Lectures**: Industry and academic professionals will join our lectures regularly, sharing their experiences in ML and data science, and providing career insights.

Logistics (Again)

Course website: https://usc-asap.github.io/CSCI566-S24/

Please use **Piazza** for any course-related communication
piazza.com/usc/spring2024/csci566

- This will be our primary place for answering the questions given the size of the course

Any non-necessary e-mail will be ignored.

Cloud Credits

First, **Google** has $300 for all new signups -- https://cloud.google.com/free?hl=en

- Also, we have applied for the Google Cloud credit ($50/student) -- considering pooling them for the project.

I have also applied for **USC advanced computing credit** -- https://www.carc.usc.edu/; to request the usage of it, please add your information

- https://docs.google.com/spreadsheets/d/1KWPlHyMVZ-2o47IhpFC9gxDYusbchPuvX971HhtUULI/edit?usp=sharing

Cloud Credits

Google Cloud Credit:
https://urldefense.com/v3/__https://gcp.secure.force.com/GCPEDU?cid=U4cYmFRYRNMCY3QWOi3mu7niNVKxs*2Fq76pJeSiD9VVD23kqHSaeE71KE2OJOYly6*__;JS8!!LIr3w8kk_Xxm!rkVFbeK9qPAsJiV4bg5sEaMqL_gHvBxGjsvzaUVDmNJHJeIYJfXCjpVeTlmm5RYk5wcsdrkD_VAFFuSRkAs0nbyaFxNL_w$

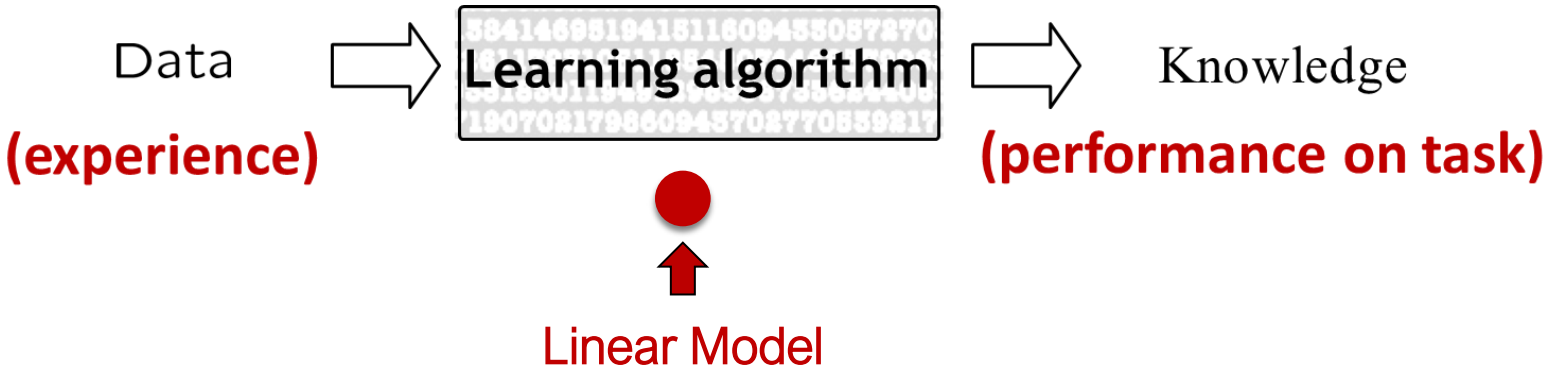
USC Credit: https://docs.google.com/spreadsheets/d/1KWPlHyMVZ-2o47IhpFC9gxDYusbchPuvX971HhtUULI/edit#gid=0

See details: https://piazza.com/class/loendg1jxkryr/post/21

Classical ML Algorithms

Decision Tree

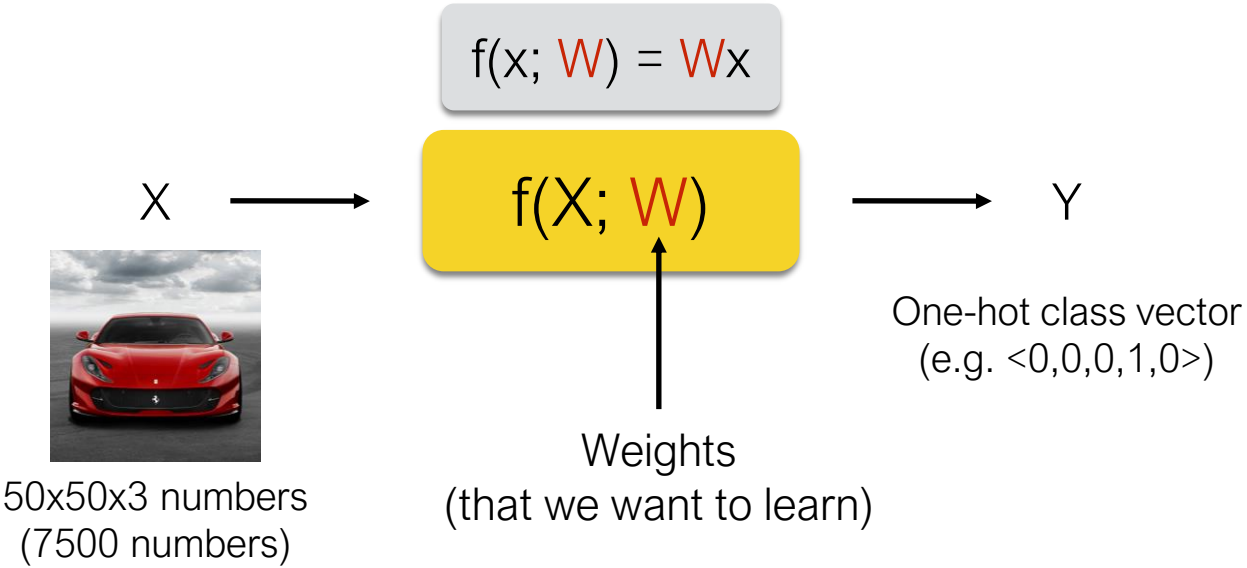
Recap: Tasks, Experience, Performance



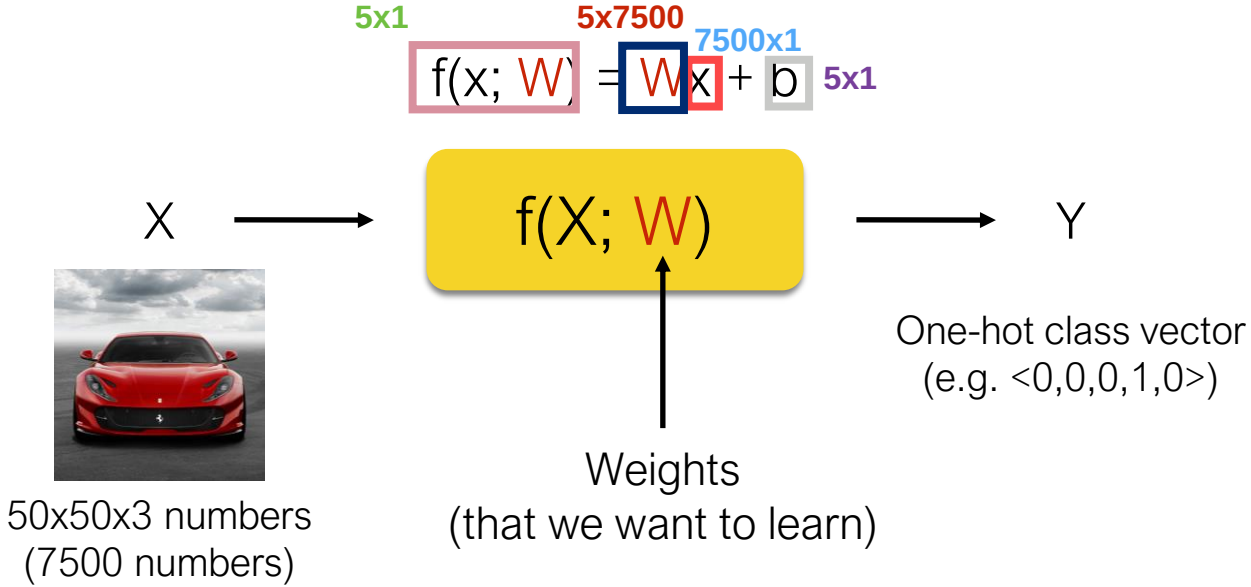
Recap: From Linear Classification to Deep Learning

Let's first talk about learning a simple function.

Recap: From Linear Classification to Deep Learning



Recap: From Linear Classification to Deep Learning



Recap: From Linear Classification to Deep Learning

5x1 5x7500 7500x1

$f(x; W) = Wx + b$ 5x1

X → f(X; W) → Y

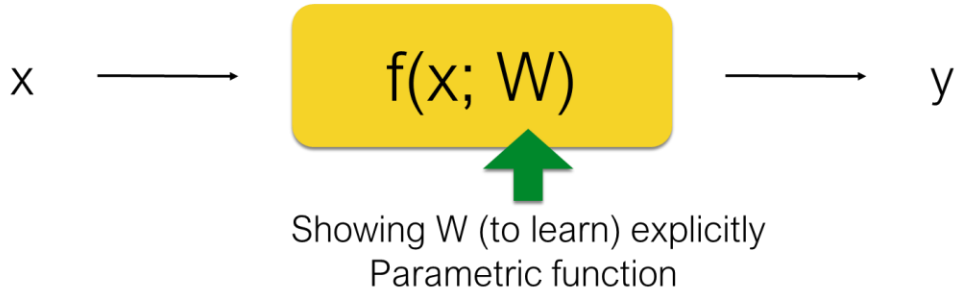
50x50x3 numbers
(7500 numbers)

Weights
(that we want to learn)

One-hot class vector
(e.g. <0,0,0,1,0>)

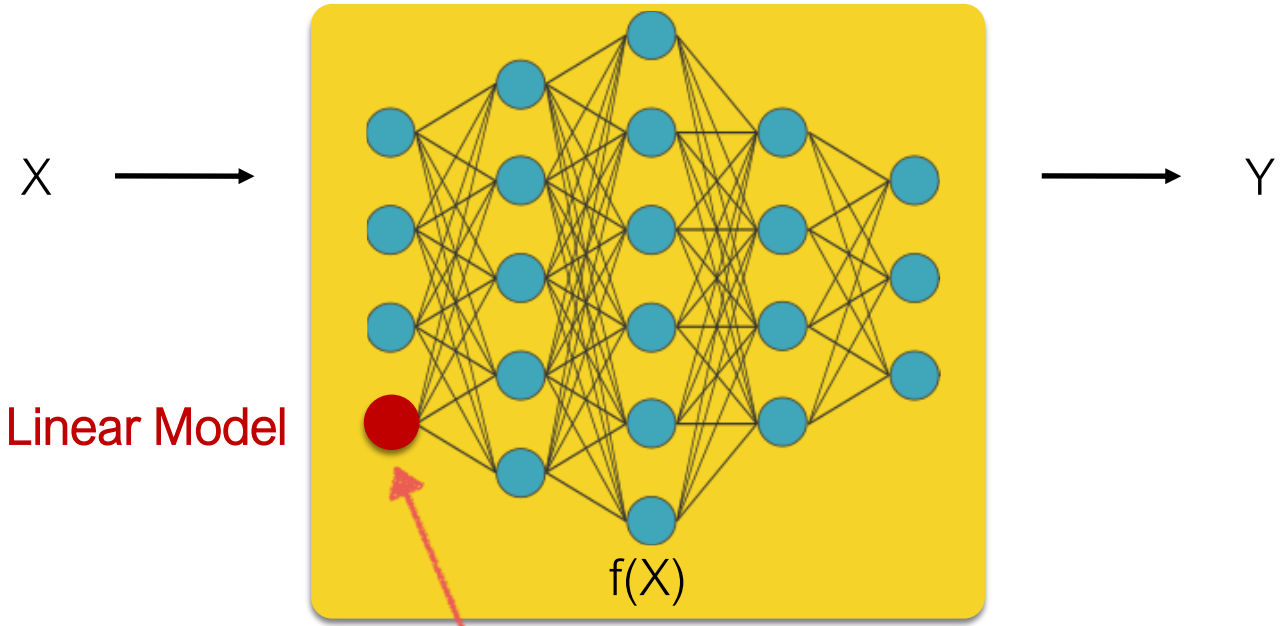
Y: output
X: input
W, b: **learned weight**

Summary of Loss Function and Optimization



- **Loss function (L)** measures how well learned W can map X to Y (compared to y*).
- **Optimization** finds the best W given a loss function L (i.e. finding W that minimizes L).

Recap: From Linear Classification to Deep Learning



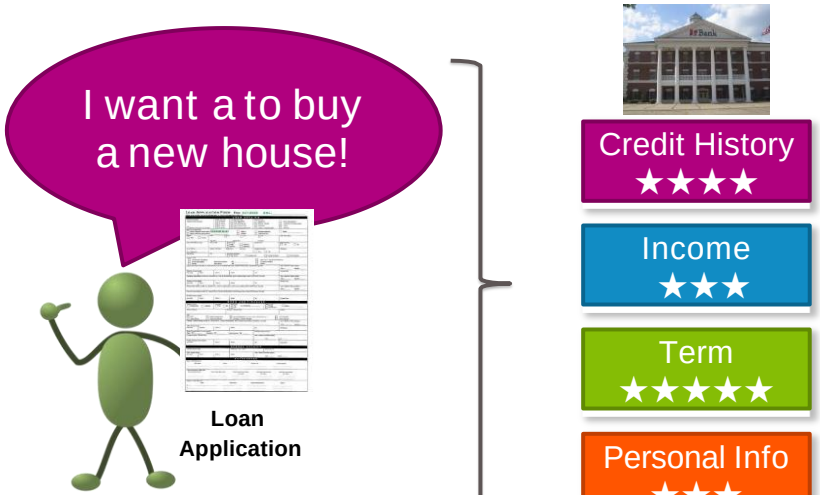
Apparently, this is a unit used in deep neural networks too.

Alternatives to Learn Function f

Of course, there are more approaches beyond linear models, which is parametric

Decision Tree

What makes a loan risky?



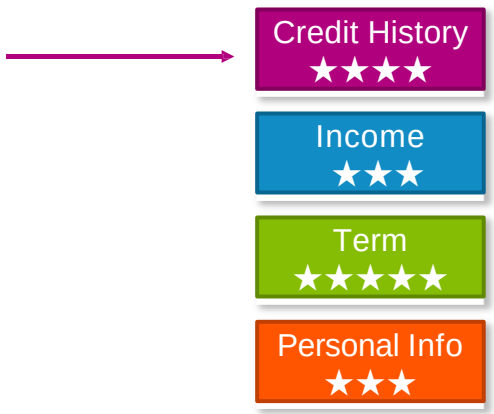
Credit to Stanford CS 229

Decision Tree

Credit history explained

Did I pay previous loans on time?

Example:
excellent, good, or fair



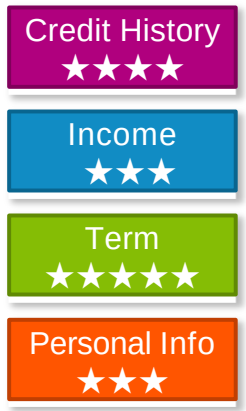
Credit to Stanford CS 229

Decision Tree

Income

What's my income?

Example:
$80K per year



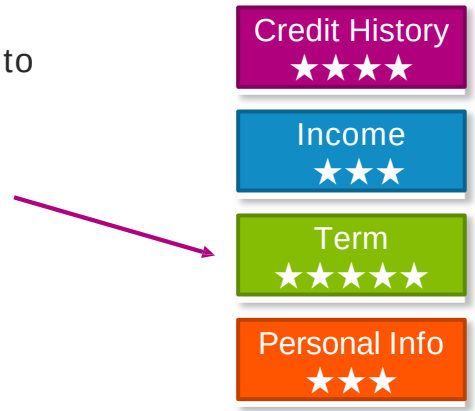
Credit to Stanford CS 229

Decision Tree

Loan terms

How soon do I need to pay the loan?

Example: 3 years, 5 years,…



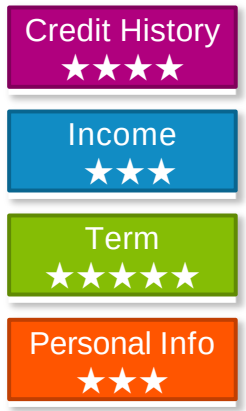
Credit to Stanford CS 229

Decision Tree

Personal information

Age, reason for the loan, marital status,…

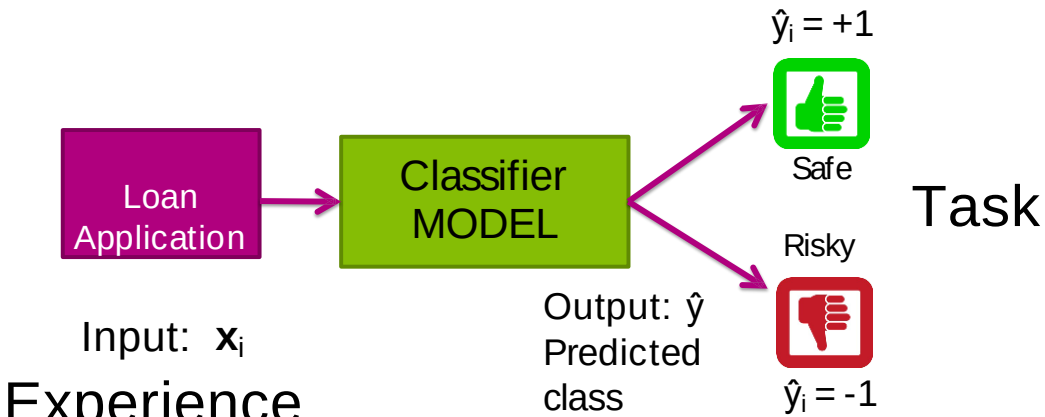
Example: Home loan for a married couple



Credit to Stanford CS 229

Decision Tree – Experience, Model, Task

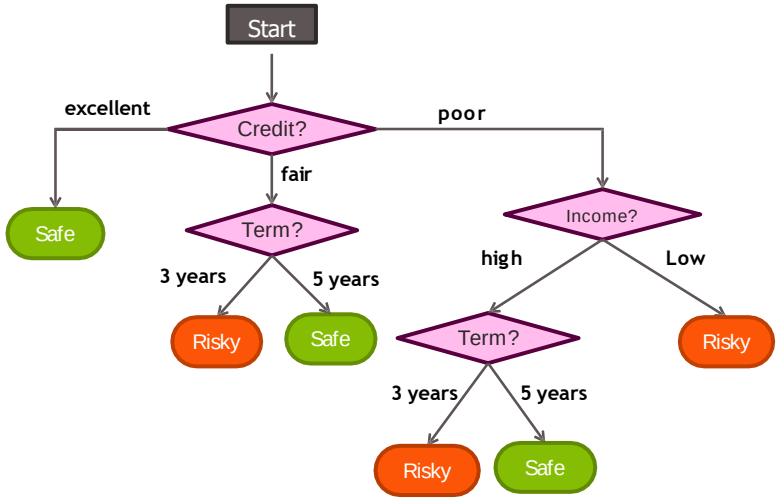
Classifier review



Credit to Stanford CS 229

Decision Tree

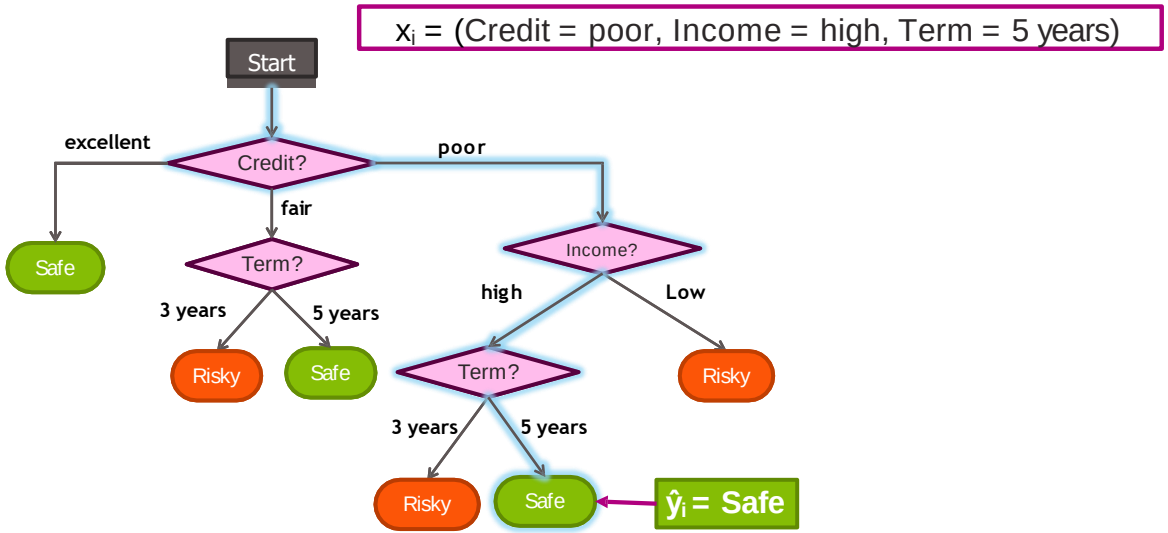
This module ... decision trees



Credit to Stanford CS 229

Decision Tree

Scoring a loan application



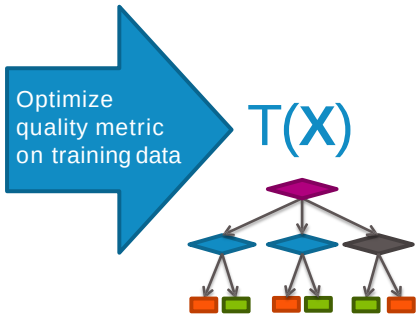
Credit to Stanford CS 229

Decision Tree

Decision tree learning problem

Training data: N observations (x_i, y_i)

Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	5 yrs	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	low	safe
poor	3 yrs	high	risky
poor	5 yrs	low	safe
fair	3 yrs	high	safe



Credit to Stanford CS 229

Decision Tree - Performance

Quality metric: Classification error

- Error measures fraction of mistakes

$$\text{Error} = \frac{\#\ \text{incorrect predictions}}{\#\ \text{examples}}$$

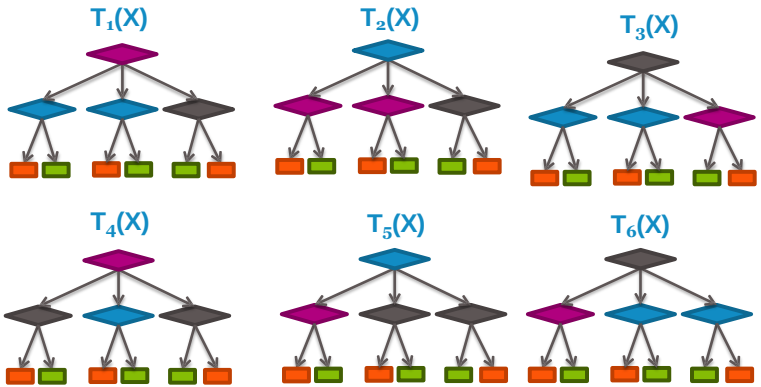
  - Best possible value : 0.0
  - Worst possible value: 1.0

Credit to Stanford CS 229

Decision Tree

How do we find the best tree?

Exponentially large number of possible trees makes decision tree learning hard!



Learning the smallest decision tree is an ***NP-hard*** *problem* [Hyafil & Rivest '76]

Credit to Stanford CS 229

Recap - Optimization

How do we find W minimizing L?

* Random search
* Analytic solution
* Numerical approach (gradient descent)

$f(x; W) \rightarrow y^*$

L(W; f,x,y*)

As an example, let's pick a simple loss function.

$L(W) = || f(x; W) - y^* ||_2$

Decision Tree

Our training data table

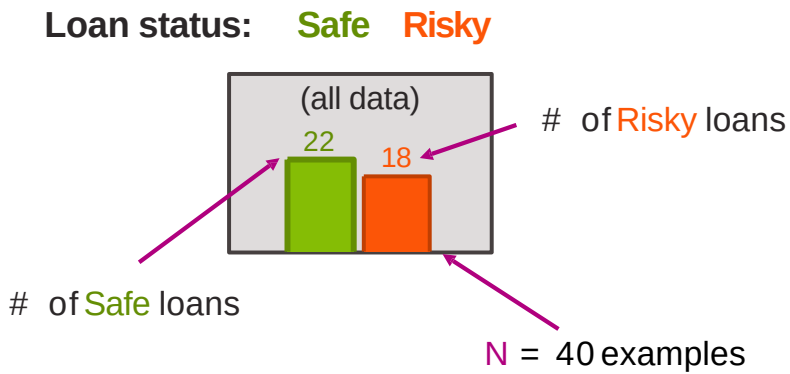
Assume N = 40, 3 features

Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	5 yrs	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	low	safe
poor	3 yrs	high	risky
poor	5 yrs	low	safe
fair	3 yrs	high	safe

Credit to Stanford CS 229

Decision Tree

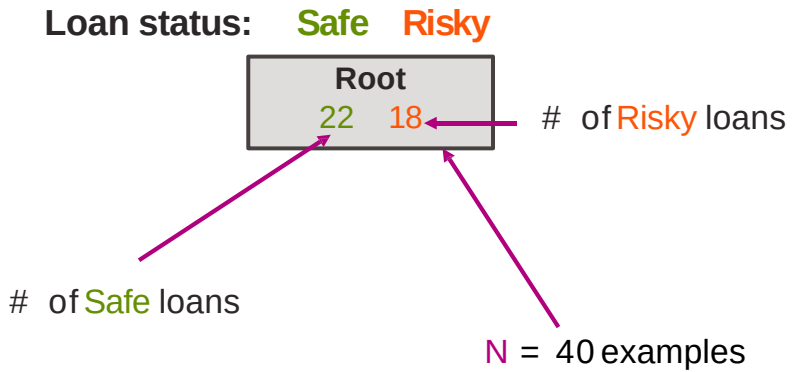
Start with all the data



Credit to Stanford CS 229

Decision Tree

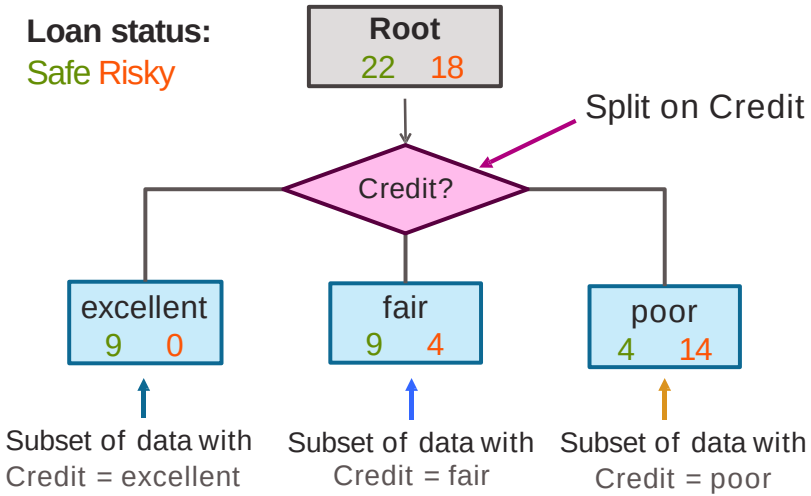
Compact visual notation: Root node



Credit to Stanford CS 229

Decision Tree

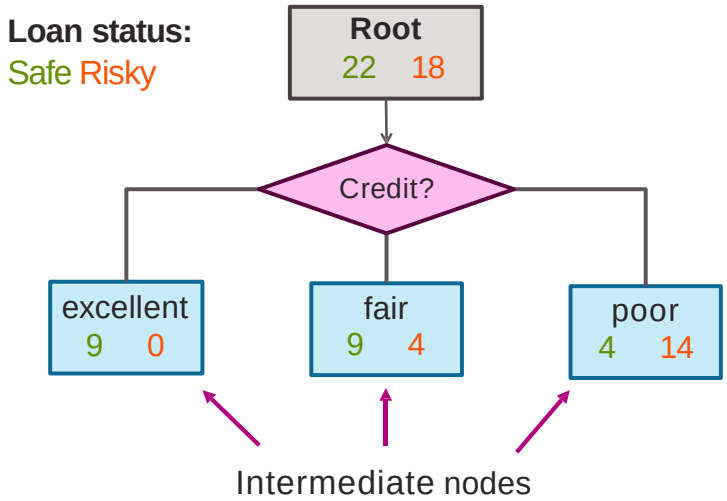
Decision stump: Single level tree



Credit to Stanford CS 229

Decision Tree

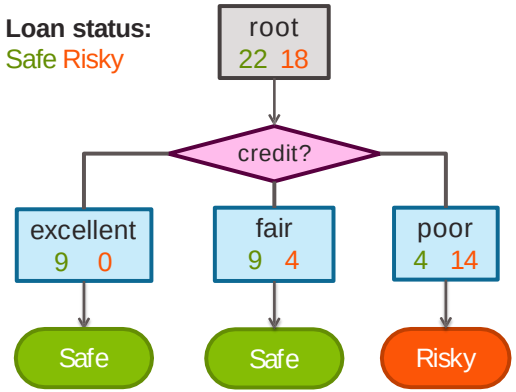
Visual notation: Intermediate nodes



Credit to Stanford CS 229

Decision Tree

Making predictions with a decision stump

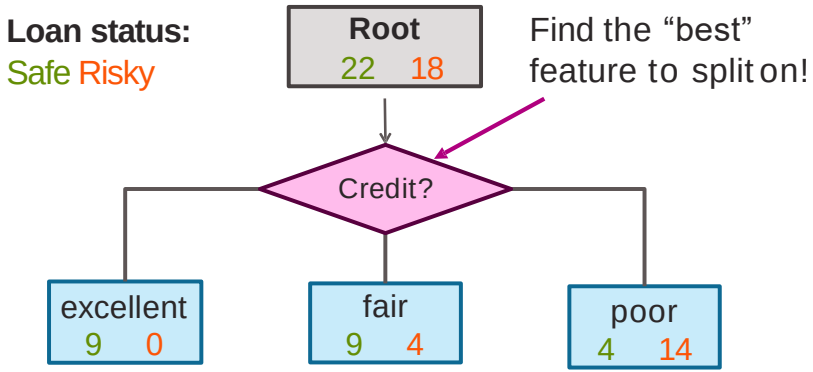


For each intermediate node, set $\hat{y}$ = majority value

Credit to Stanford CS 229

Decision Tree

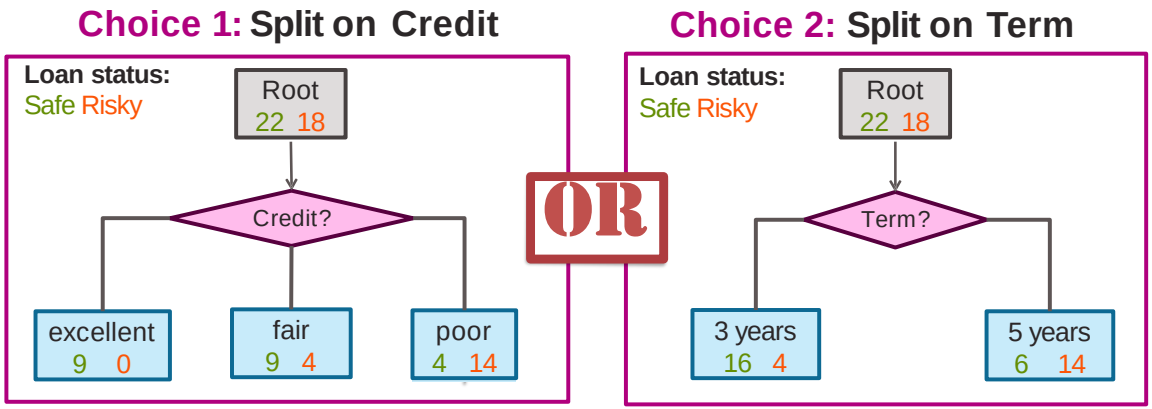
How do we learn a decision stump?



Credit to Stanford CS 229

Decision Tree

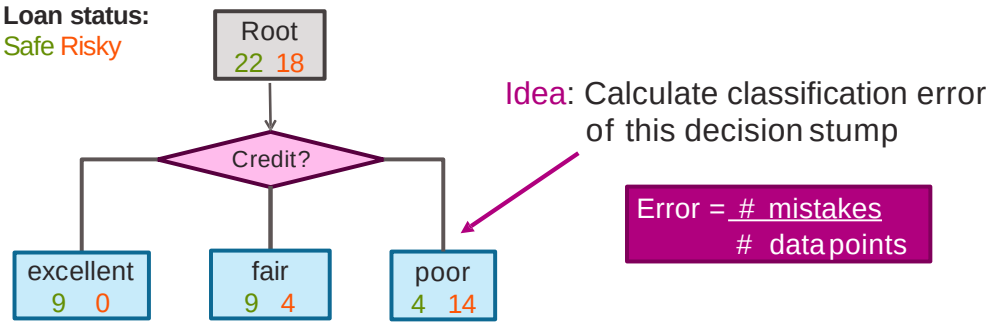
How do we select the best feature?



Credit to Stanford CS 229

Decision Tree

How do we measure effectiveness of a split?

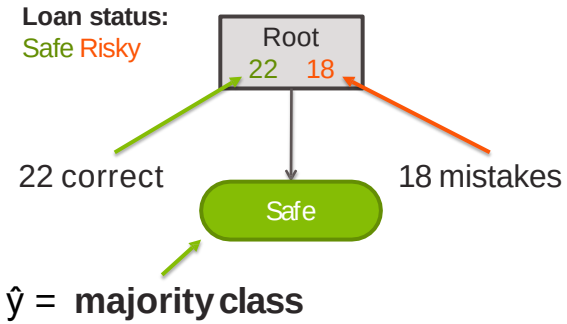


Credit to Stanford CS 229

Decision Tree

Calculating classification error

- Step 1: $\hat{y}$ = class of majority of data in node
- Step 2: Calculate classification error of predicting $\hat{y}$ for this data



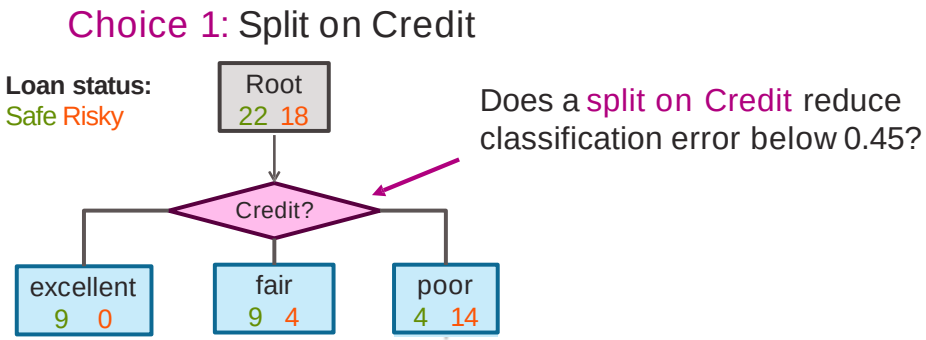
Error = $\frac{\#\ \text{mistakes}}{\#\ \text{data points}}$

Tree	Classification error
(root)	0.45

Credit to Stanford CS 229

Decision Tree

Choice 1: Split on Credit history?

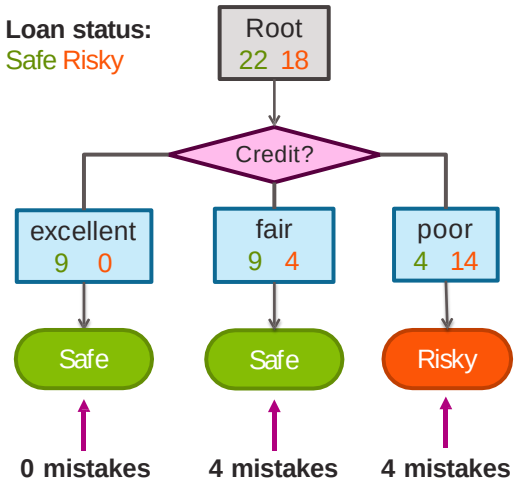


Credit to Stanford CS 229

Decision Tree

Split on Credit: Classification error

Choice 1: Split on Credit



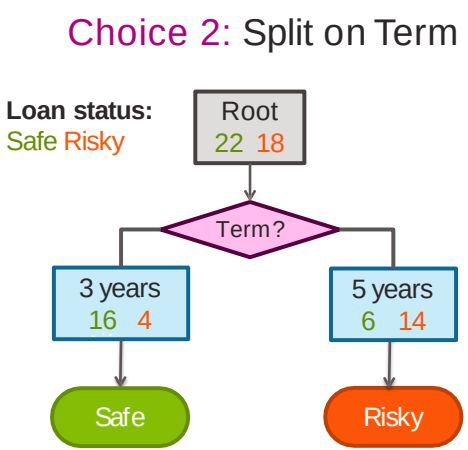
$$\text{Error} = \frac{\#\ \text{mistakes}}{\#\ \text{data points}}$$

Tree	Classification error
(root)	0.45
Split on credit	0.2

Credit to Stanford CS 229

Decision Tree

Choice 2: Split on Term?

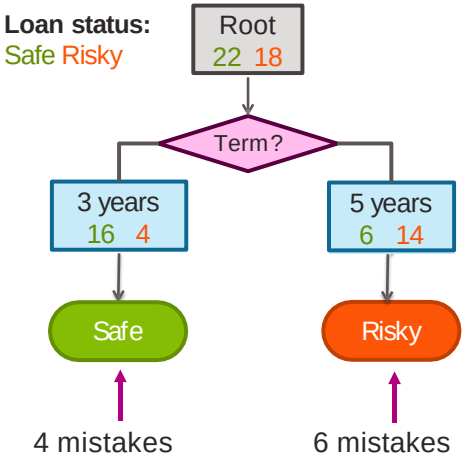


Credit to Stanford CS 229

Decision Tree

Evaluating the split on Term

Choice 2: Split on Term



Error = # mistakes / # data points

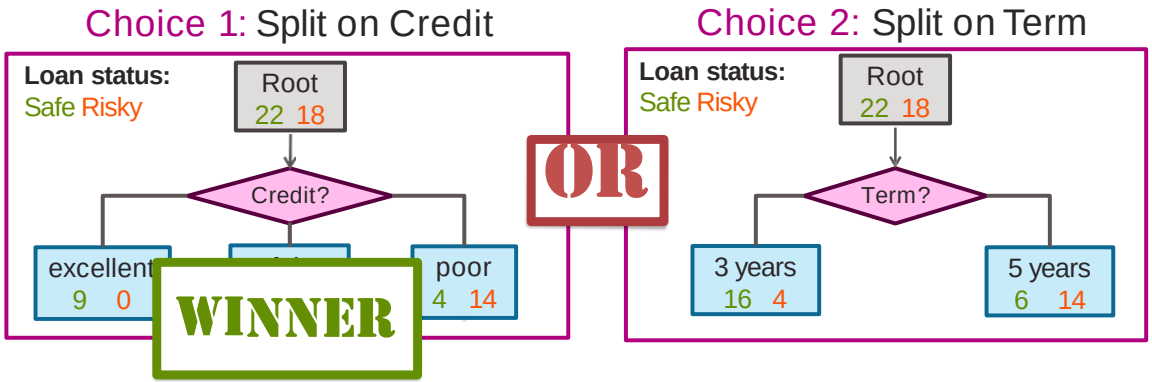
Tree	Classification error
(root)	0.45
Split on credit	0.2
Split on term	0.25

Credit to Stanford CS 229

Decision Tree

Choice 1 vs Choice 2:
Comparing split on
Credit vs Term

Tree	Classification error
(root)	0.45
split on credit	0.2
split on loanterm	0.25



Credit to Stanford CS 229

Decision Tree

Feature split selection algorithm

- Given a subset of data M (a node in a tree)
- For each feature $h_i(x)$:
  1. Split data of M according to feature $h_i(x)$
  2. Compute classification error of split
- Chose feature $h^*(x)$ with lowest classification error

Credit to Stanford CS 229

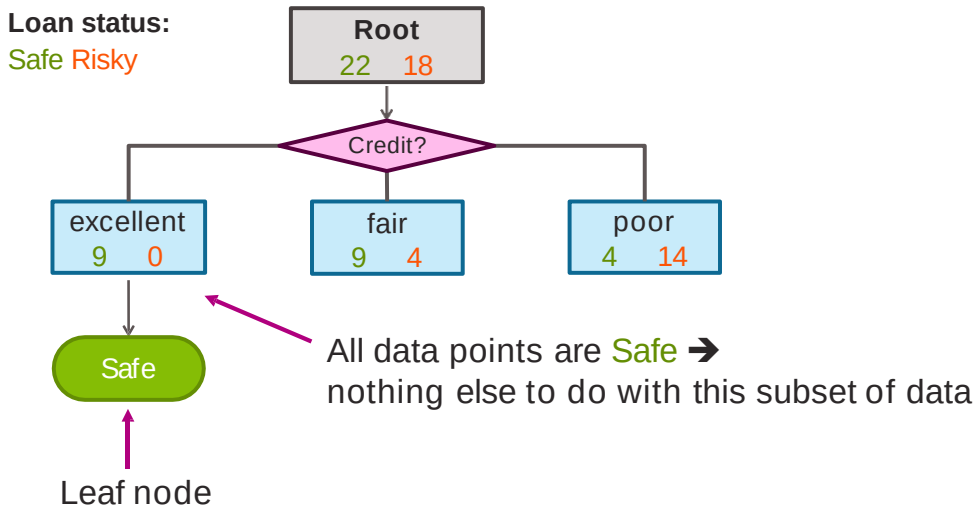
Decision Tree

Recursion & Stopping conditions

Credit to Stanford CS 229

Decision Tree

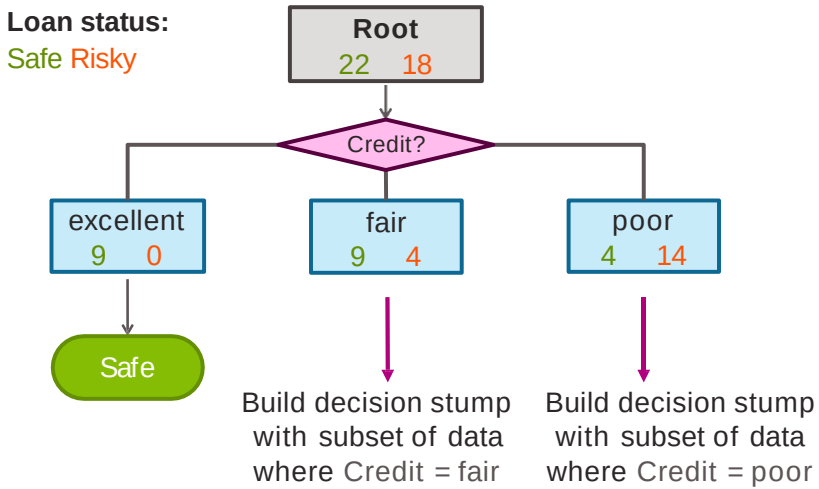
We've learned a decision stump, what next?



Credit to Stanford CS 229

Decision Tree

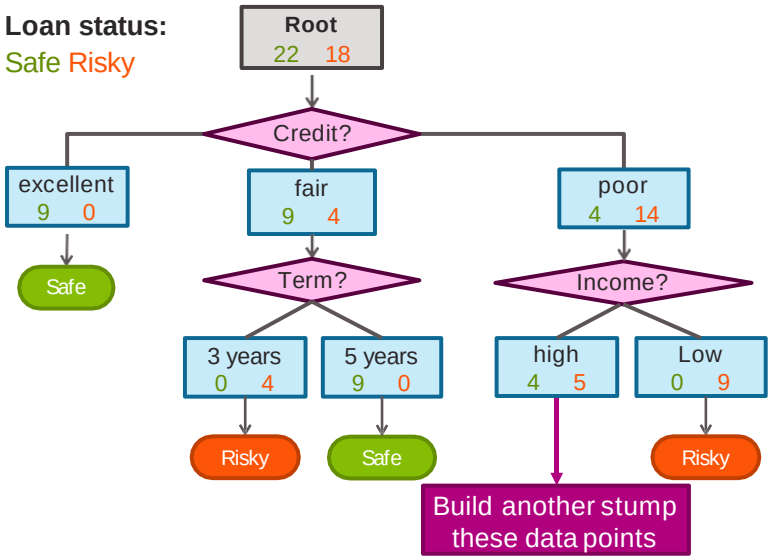
Tree learning = Recursive stump learning



Credit to Stanford CS 229

Decision Tree

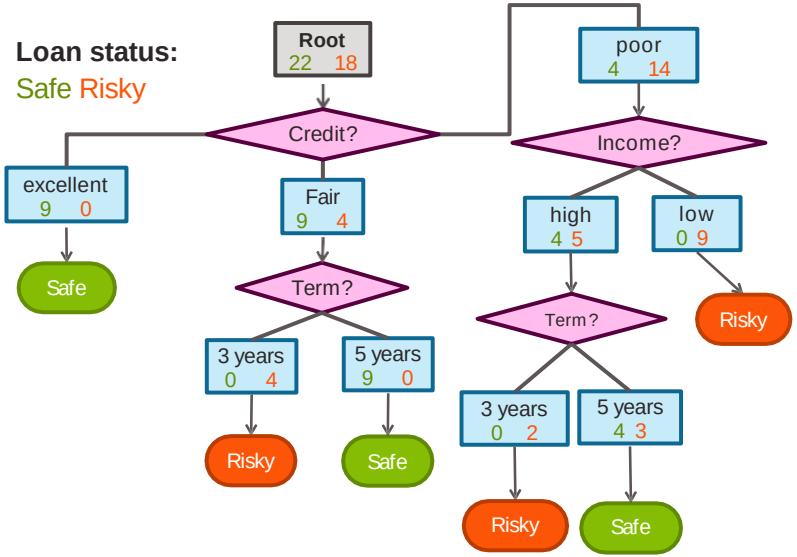
Second level



Credit to Stanford CS 229

Decision Tree

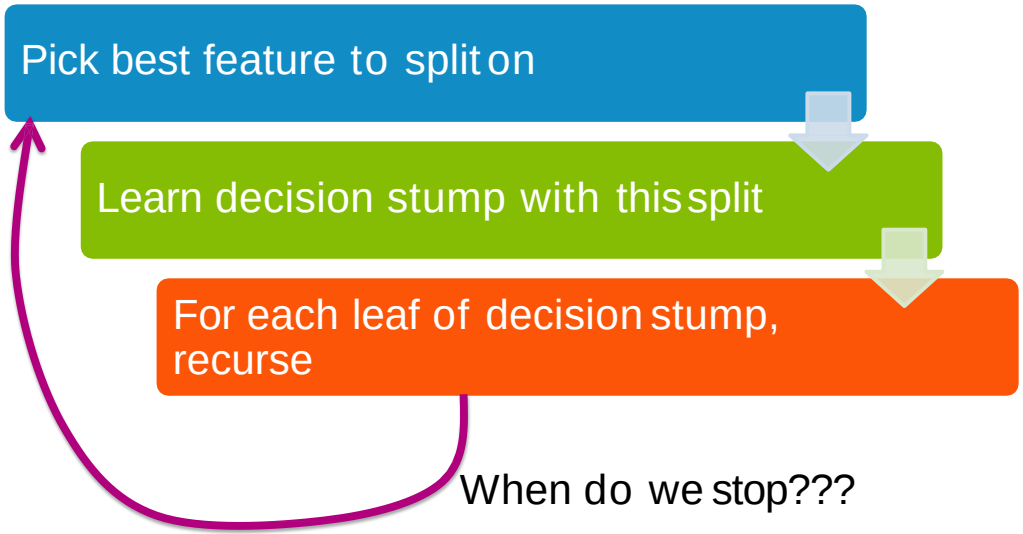
Final decision tree



Credit to Stanford CS 229

Decision Tree

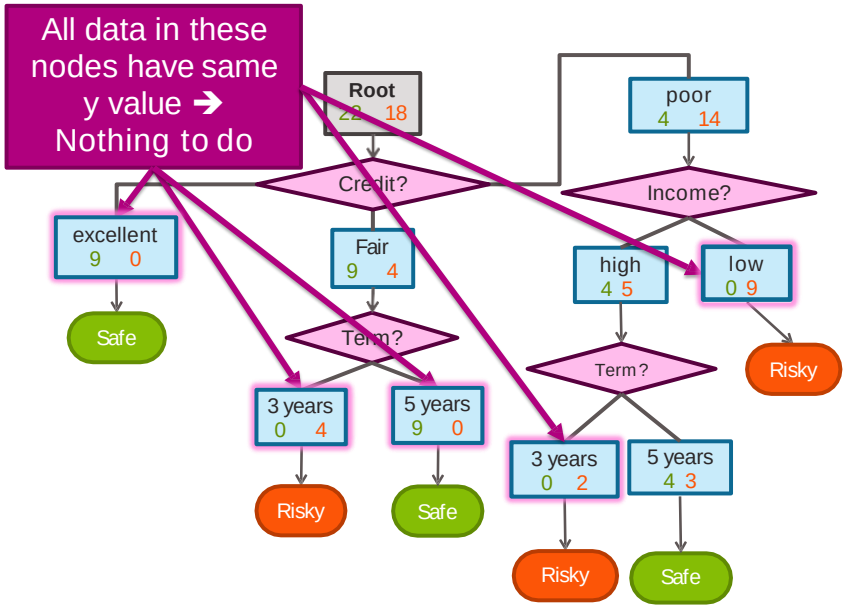
Simple greedy decision tree learning



Credit to Stanford CS 229

Decision Tree

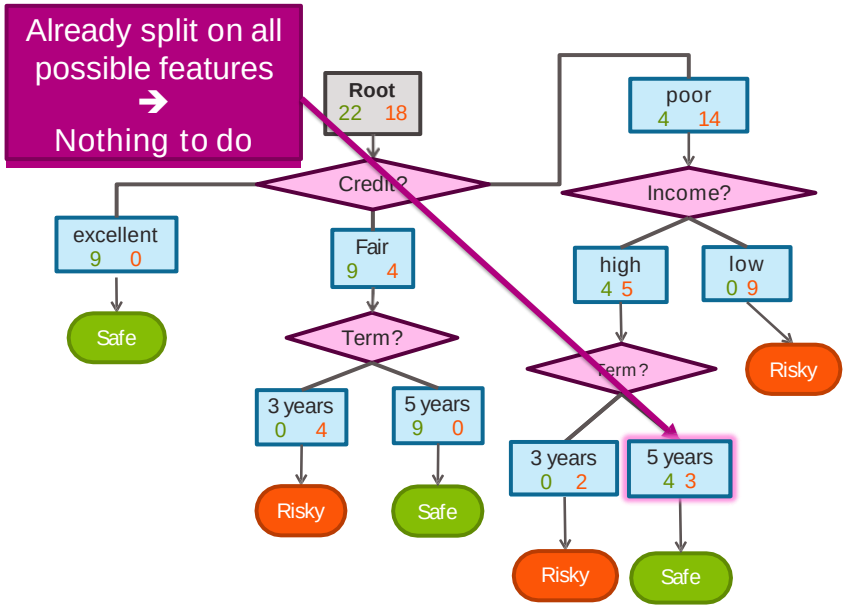
Stopping condition 1: All data agrees on y



Credit to Stanford CS 229

Decision Tree

Stopping condition 2: Already split on all features



Credit to Stanford CS 229

Decision Tree

Greedy decision tree learning

- Step 1: Start with an empty tree
- Step 2: Select a feature to split data
- For each split of the tree:
  - Step 3: If nothing more to do, make predictions
  - Step 4: Otherwise, go to Step 2 & continue (recurse) on this split

Pick feature split leading to lowest classification error

Stopping conditions

Recursion

Credit to Stanford CS 229

Decision Tree

Is this a good idea?

Credit to Stanford CS 229

Decision Tree

Stopping condition 3:
Don't stop if error doesn't decrease???

$y = x[1]$ xor $x[2]$

x[1]	x[2]	y
False	False	False
False	True	True
True	False	True
True	True	False

y values
True False

Root
2 2

Error = $\frac{\#\ \text{mistakes}}{\#\ \text{data points}}$

Tree	Classification error
(root)	0.5

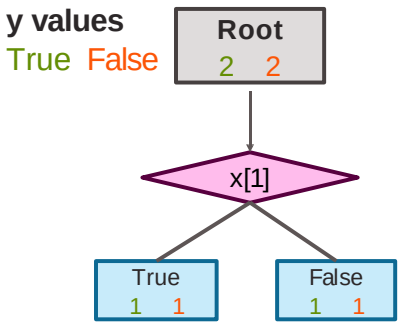
Credit to Stanford CS 229

Decision Tree

Consider split on x[1]

y = x[1] xor x[2]

x[1]	x[2]	y
False	False	False
False	True	True
True	False	True
True	True	False



Error = # mistakes / # data points

Tree	Classification error
(root)	0.5
Split on x[1]	0.5

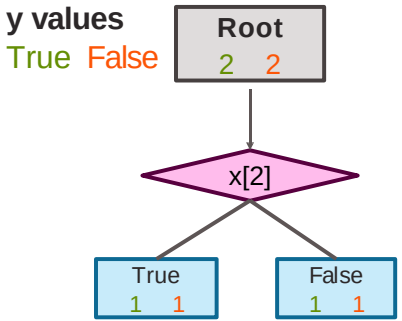
Credit to Stanford CS 229

Decision Tree

Consider split on x[2]

$y = x[1]$ xor $x[2]$

x[1]	x[2]	y
False	False	False
False	True	True
True	False	True
True	True	False



Error = $\frac{1+1}{2+2}$ = 0.5

Neither features improve training error… Stop now???

Tree	Classification error
(root)	0.5
Split on x[1]	0.5
Split on x[2]	0.5

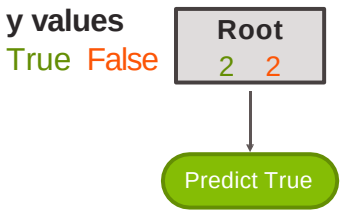
Credit to Stanford CS 229

Decision Tree

Final tree with stopping condition 3

$y = x[1] \text{ xor } x[2]$

x[1]	x[2]	y
False	False	False
False	True	True
True	False	True
True	True	False



Tree	Classification error
with stopping condition 3	0.5

Credit to Stanford CS 229

Decision Tree

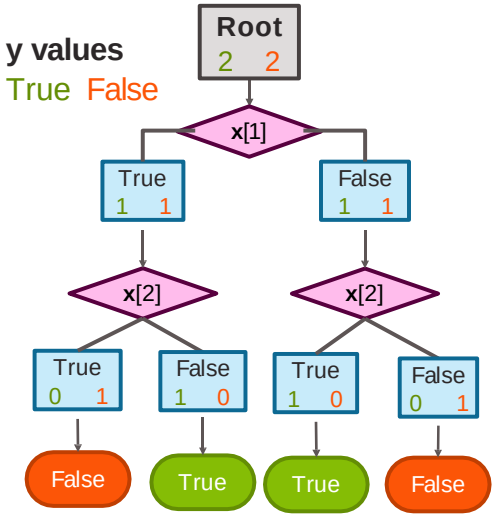
Without stopping condition 3

Condition 3 (stopping when training error doesn't' improve) is not recommended!

$y = x[1] \text{ xor } x[2]$

x[1]	x[2]	y
False	False	False
False	True	True
True	False	True
True	True	False

Tree	Classification error
with stopping condition 3	0.5
without stopping condition 3	

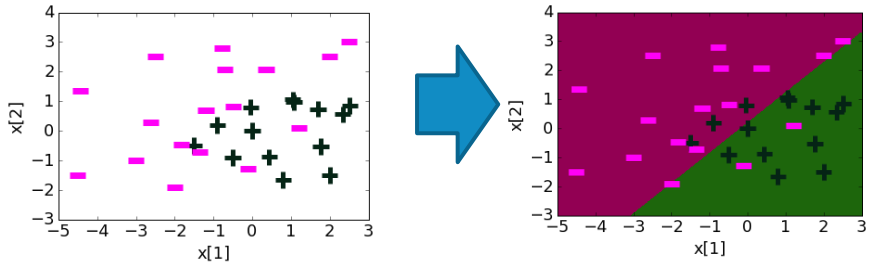


Credit to Stanford CS 229

Decision Tree

Linear classification

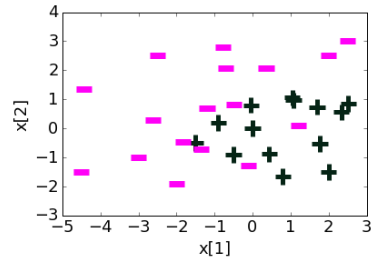
Feature	Value	Weight Learned
$h_0(x)$	1	0.22
$h_1(x)$	x[1]	1.12
$h_2(x)$	x[2]	-1.07

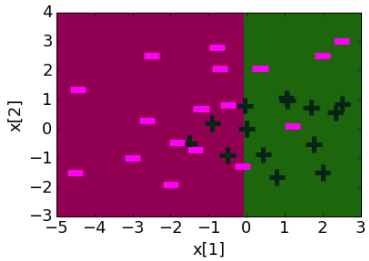


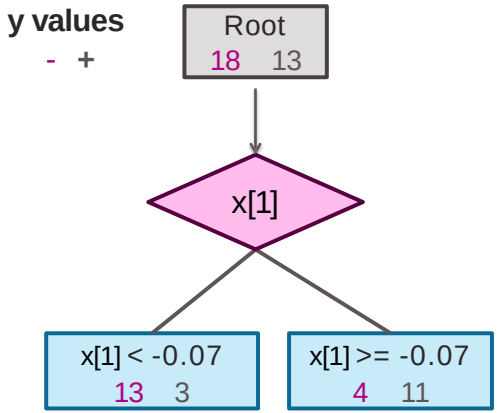
Credit to Stanford CS 229

Decision Tree

Depth 1: Split on x[1]



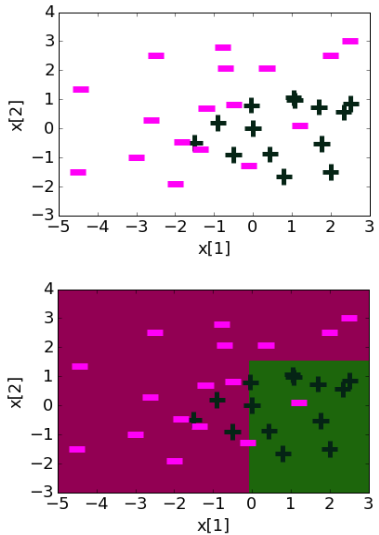


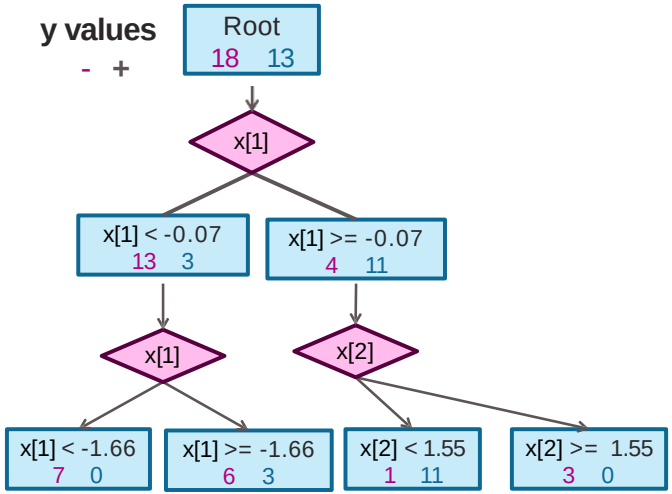


Credit to Stanford CS 229

Decision Tree

Depth 2



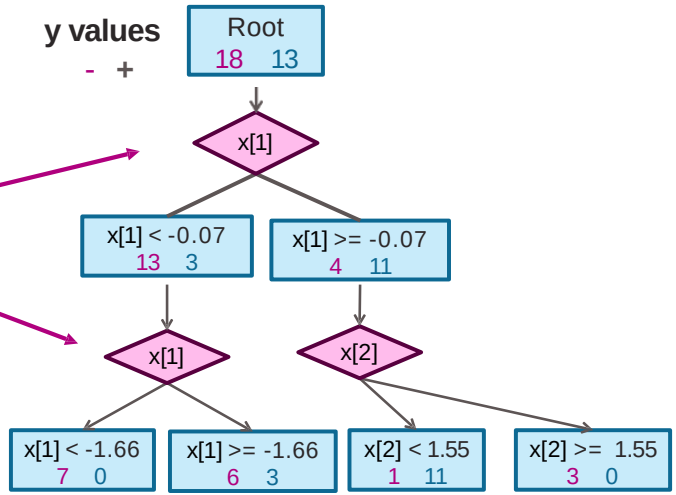


Credit to Stanford CS 229

Decision Tree

Threshold split caveat

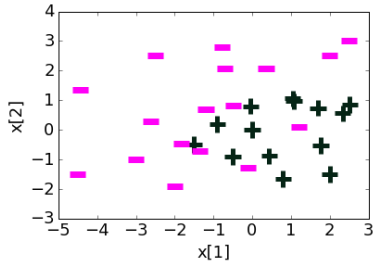
For threshold splits, same feature can be used multiple times



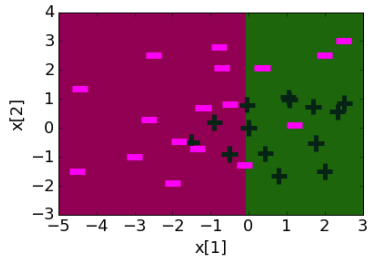
Credit to Stanford CS 229

Decision Tree

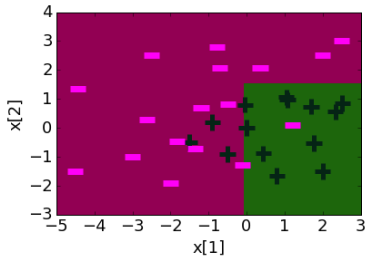
Decision boundaries



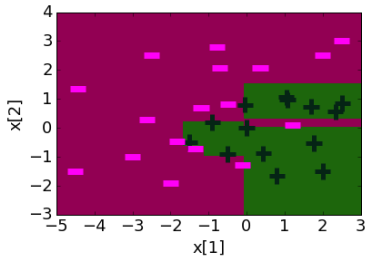
Depth 1



Depth 2



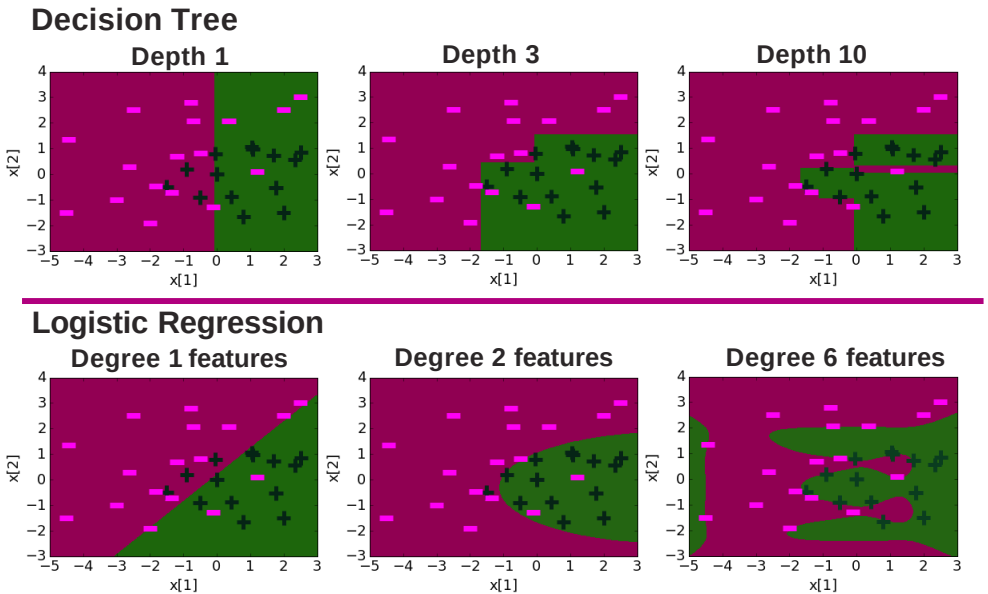
Depth 10



Credit to Stanford CS 229

Decision Tree

Comparing decision boundaries



Credit to Stanford CS 229

Decision Tree – Splitting Criteria

Bluntine & Niblett (1992) compared 4 criteria (random, Gini, mutual information, Marshall) on 12 datasets

Medical Diagnosis Datasets: (4 of 12)

- **hypo:** data set of 3772 examples records expert opinion on possible hypo- thyroid conditions from 29 real and discrete attributes of the patient such as sex, age, taking of relevant drugs, and hormone readings taken from drug samples.
- **breast:** The classes are reoccurrence or non-reoccurrence of breast cancer sometime after an operation. There are nine attributes giving details about the original cancer nodes, position on the breast, and age, with multi-valued discrete and real values.
- **tumor:** examples of the location of a primary tumor
- **lymph**: from the lymphography domain in oncology. The classes are normal, metastases, malignant, and fibrosis, and there are nineteen attributes giving details about the lymphatics and lymph nodes

Table 1. Properties of the data sets

Data Set	Classes	Attr.s	Training Set	Test Set
hypo	4	29	1000	2772
breast	2	9	200	86
tumor	22	18	237	102
lymph	4	18	103	45
LED	10	7	200	1800
mush	2	22	200	7924
votes	2	17	200	235
votes1	2	16	200	235
iris	3	4	100	50
glass	7	9	100	114
xd6	2	10	200	400
pole	2	4	200	1647

Credit to CMU 10601

Table from Bluntine & Niblett (1992)

Decision Tree – Splitting Criteria

Key Takeaway: GINI gain and Mutual Information are statistically indistinguishable!

Table 3. Error for different splitting rules (pruned trees).

	Splitting Rule			
Data Set	GINI	Info. Gain	Marsh.	Random
hypo	1.01 ± 0.29	0.95 ± 0.22	1.27 ± 0.47	7.44 ± 0.53
breast	28.66 ± 3.87	28.49 ± 4.28	27.15 ± 4.22	29.65 ± 4.97
tumor	60.88 ± 5.44	62.70 ± 3.89	61.62 ± 3.98	67.94 ± 5.68
lymph	24.44 ± 6.92	24.00 ± 6.87	24.33 ± 5.51	32.33 ± 11.25
LED	33.77 ± 3.06	32.89 ± 2.59	33.15 ± 4.02	38.18 ± 4.57
mush	1.44 ± 0.47	1.44 ± 0.47	7.31 ± 2.25	8.77 ± 4.65
votes	4.47 ± 0.95	4.57 ± 0.87	11.77 ± 3.95	12.40 ± 4.56
votes1	12.79 ± 1.48	13.04 ± 1.65	15.13 ± 2.89	15.62 ± 2.73
iris	5.00 ± 3.08	4.90 ± 3.08	5.50 ± 2.59	14.20 ± 6.77
glass	39.56 ± 6.20	50.57 ± 6.73	40.53 ± 6.41	53.20 ± 5.01
xd6	22.14 ± 3.23	22.17 ± 3.36	2.06 ± 3.37	31.86 ± 3.62
pole	15.43 ± 1.51	15.47 ± 0.88		

Info. Gain is another name for *mutual information*

Table from Bluntine & Niblett (1992)

Decision Tree – Splitting Criteria

Table 4. Difference and significance of error for GINI splitting rule versus others.

	Splitting Rule		
Data Set	Info. Gain	Marsh.	Random
hypo	−0.06 (0.82)	0.26 (0.99)	6.43 (1.00)
breast	−0.17 (0.23)	−1.51 (0.94)	0.99 (0.72)
tumor	1.81 (0.84)	.74 (	7.86 (
lymph	−0.44 (0.83)	−0.1	
LED	0.12 (0.17)	0.3	
mush	0.00 (0.00)	5.8	
votes	0.11 (0.55)	7.3	
votes1	0.26 (0.47)	2.3	
iris	−0.10 (0.67)	0.5	
glass	1.01 (0.50)	0.9	
xd6	0.04 (0.11)	−0.0	
pole	0.03 (0.11)	−0.4	

Key Takeaway: GINI gain and Mutual Information are statistically indistinguishable!

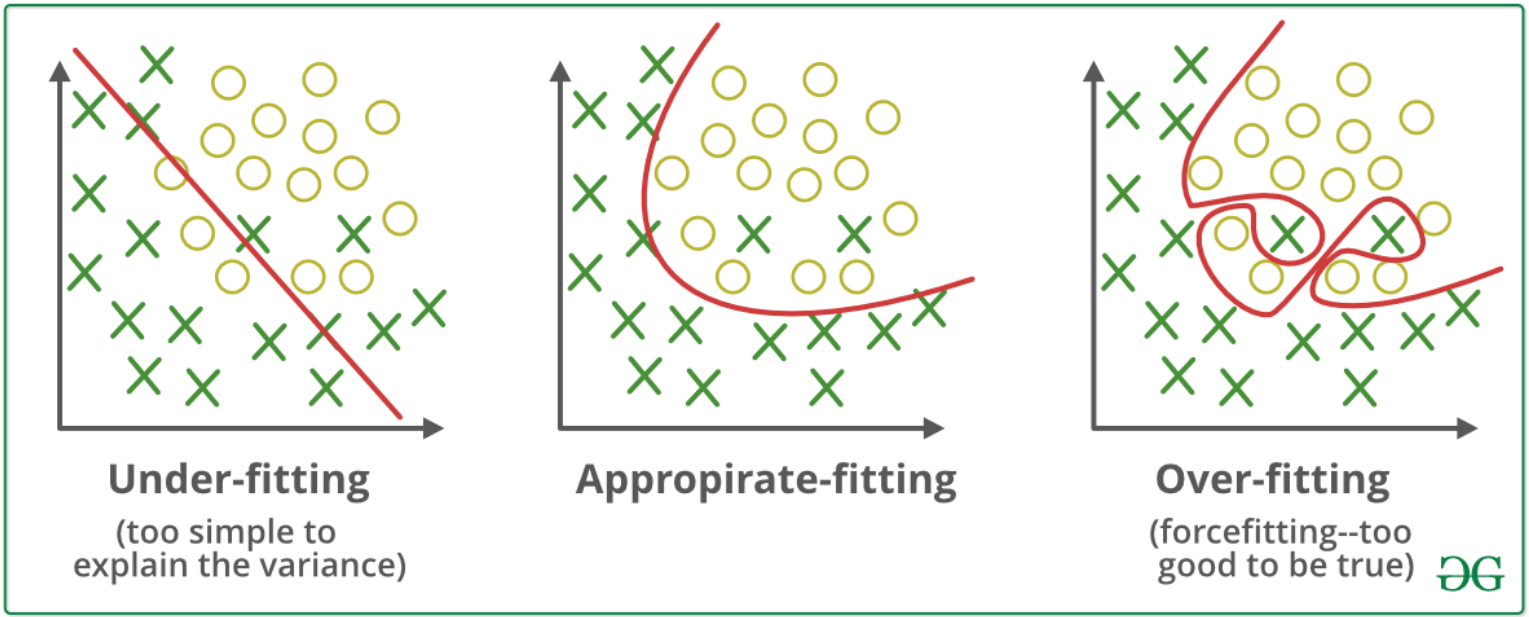
Results are of the form A.AA (B.BB) where:

1. A.AA is the **average difference in errors** between the two methods
2. B.BB is the **significance** of the difference according to a two-tailed **paired t-test**

Credit to CMU 10601

Table from Bluntine & Niblett (1992)

Overfitting and Underfitting Revisited



We favor the case where the ML model appropriately depict the underlying data

Linear Model Regularization: L1 and L2

What if $(\mathbf{A}^T\mathbf{A})$ is not invertible ?

r equations , p unknowns – underdetermined system of linear equations

many feasible solutions Need to constrain solution further

e.g. bias solution to "small" values of β (small changes in input don't translate to large changes in output)

$$\hat{\beta}_{\text{MAP}} = \arg\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda\|\beta\|_2^2$$

Ridge Regression (l2 penalty)

$$\hat{\beta}_{\text{MAP}} = \arg\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda\|\beta\|_1$$

Lasso

(l1 penalty)

Credit to CMU 10601

DT Generalization – Overfitting Revisited

Question:

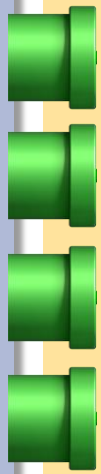
Which of the following would generalize best to unseen examples?

A. Small tree with low training accuracy

B. Large tree with low training accuracy

C. Small tree with high training accuracy

D. Large tree with high training accuracy



Answer:

Credit to CMU 10601

DT Generalization – Overfitting Revisited

Underfitting

- The model…
  - is too simple
  - is unable captures the trends in the data
  - exhibits too much bias
- *Example*: majority-vote classifier (i.e. depth-zero decision tree)
- *Example*: a toddler (that has not attended medical school) attempting to carry out medical diagnosis

Credit to CMU 10601

Overfitting

- The model…
  - is too complex
  - is fitting the noise in the data or fitting "outliers"
  - does not have enough bias
- *Example*: our "memorizer" algorithm responding to an irrelevant attribute
- *Example*: medical student who simply memorizes patient case studies, but does not understand how to apply knowledge to new patients

DT Generalization – Overfitting Revisited

- Given a hypothesis *h*, its…
  - …error rate over all training data: $error(h, D_{train})$
  - …error rate over all test data: $error(h, D_{test})$
  - …true error over all data: $error_{true}(h)$
- We say h overfits the training data if…

$$error_{true}(h) > error(h, D_{train})$$

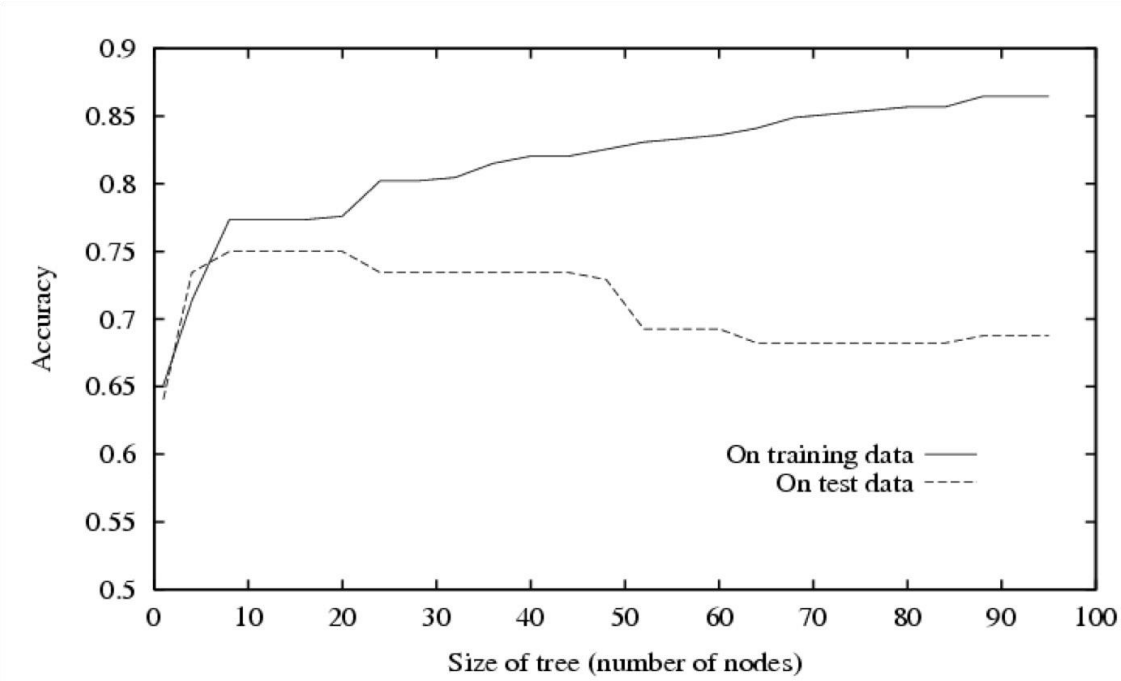
- Amount of overfitting =

$$error_{true}(h) - error(h, D_{train})$$

In practice, $error_{true}(h)$ is **unknown**

Credit to CMU 10601

DT Generalization – Overfitting Revisited



Credit to CMU 10601

Figure from Tom Mitchell

Underfitting and Overfitting

- Training error: model error on the training data
- Generalization error: model error on new data

Generalization error \ Training error	Low	High
Low	Good	Bug?
High	Overfitting	Underfitting

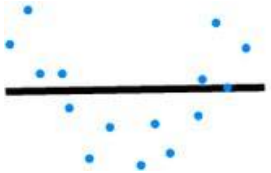
Alternative View: Data and Model Complexity

Data complexity

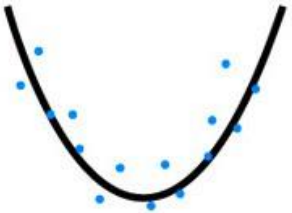
Model complexity

	Low	High
Low	Normal	Underfitting
High	Overfitting	Normal

Underfitting



Normal



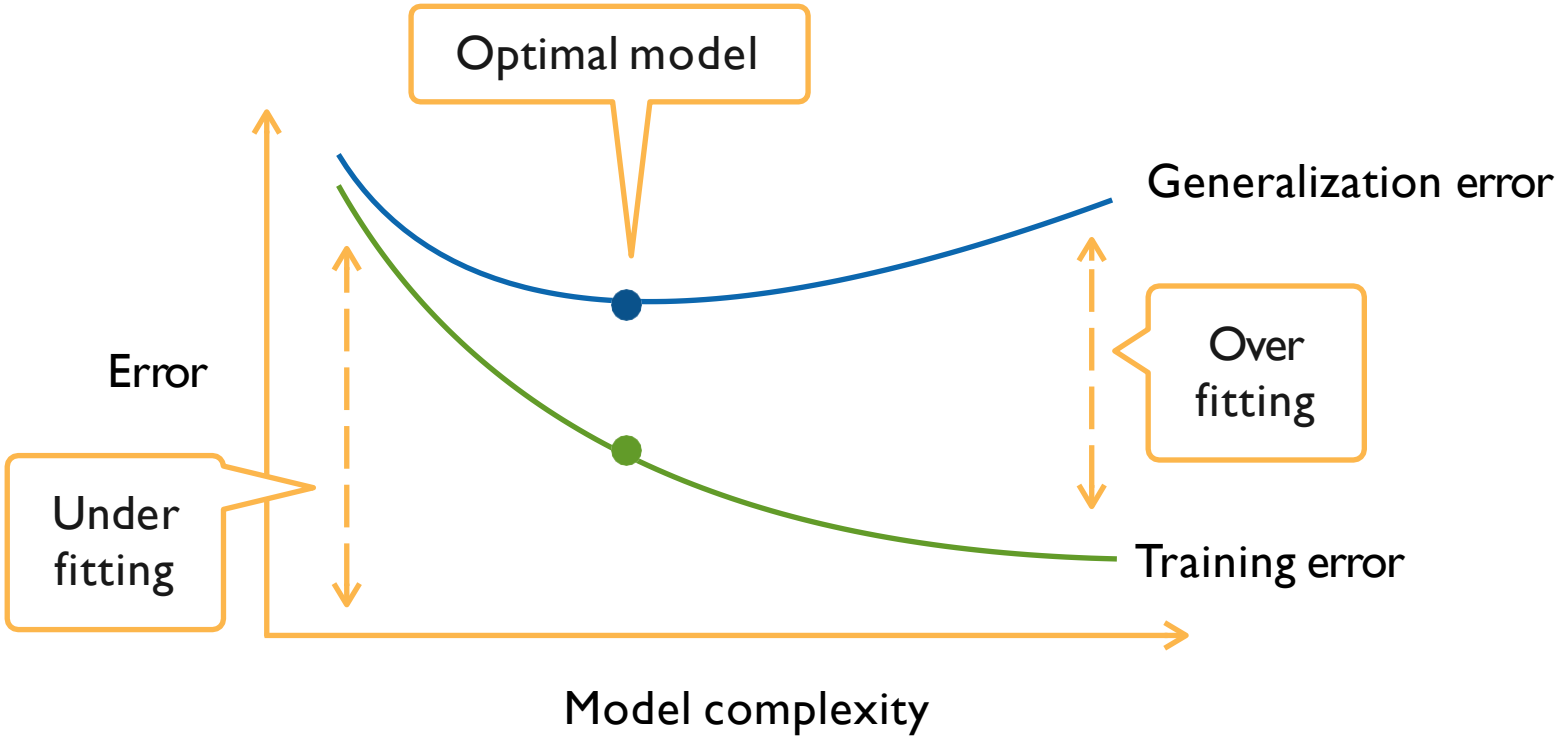
Overfitting

Image credit:hackernoon.com

Model Complexity

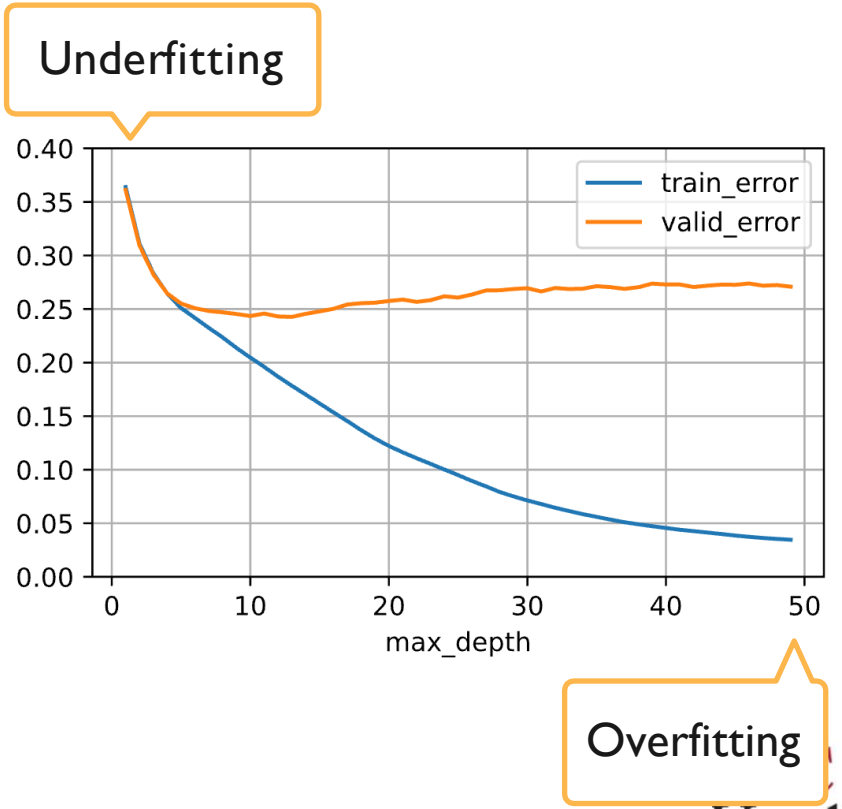
- The capacity of a set of function to fit data points
- In ML, model complexity usually refers to:
  - The number of learnable parameters
  - The value range for those parameters
- It's hard to compare between different types of ML models
  - E.g. trees vs neural network
- More precisely measure of complexity: VC dimension
  - VC dim for classification model:
    the maximum number of examples the model can shatter

Model Complexity



Model Complexity Example: Decision Tree

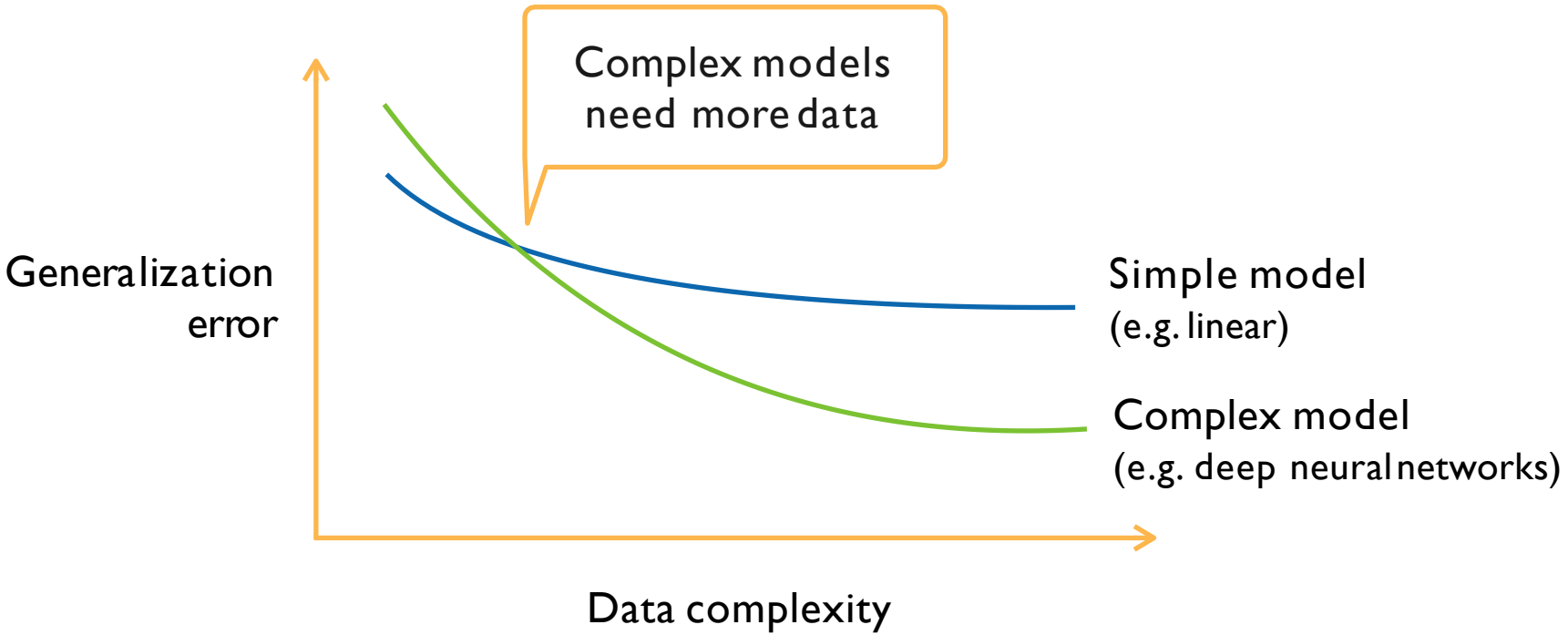
- The tree size can be controlled by the number of levels
- Use scikit-learn **DecisionTreeRegressor(max_depth =n)** on house sales data



Data Complexity

- Multiple factors matters
  - # of examples
  - # of features in each example
  - the separability of the classes
- Again, hard to compare among very different data
  - E.g a char vs a pixel
- More precisely, Kolmogorov complexity
  - A data is simple if it can be generated by a short program

Model Complexity vs Data Complexity



Model Selection

- Pick a model with a proper complexity for your data
  - Minimize the generalization error
  - Also consider business metrics
- Pick up a model family, then select proper hyper-parameters
  - Trees: #trees, maximal depths
  - Neural networks: architecture, depth (#layers), width (#hidden units), regularizations
  - **We will revisit the model selection later**

Summary

- **Model complexity**: the ability to fit various functions
- **Data complexity**: the richness of information
- **Model selection**: match model and data complexities

DT Generalization – Overfitting Revisited

For Decision Trees…

1. Do not grow tree beyond some **maximum depth**
2. Do not split if splitting criterion (e.g. mutual information) is **below some threshold**
3. Stop growing when the split is **not statistically significant**
4. Grow the entire tree, then **prune**

Credit to CMU 10601

DT Generalization – Pruning

1. Split data in two: *training* dataset and *validation* dataset
2. Grow the full tree using the *training* dataset
3. Repeatedly prune the tree:
   - Evaluate each split using a *validation* dataset by comparing the validation error rate **with and without** that split
   - (Greedily) remove the split that most decreases the validation error rate
   - Stop if no split improves validation error, otherwise repeat

Credit to CMU 10601

Decision Tree in the Wild

- DTs are one of the most popular classification methods for practical applications
  - Reason #1: The learned representation is **easy to explain** a non-ML person
  - Reason #2: They are **efficient** in both computation and memory
- DTs can be applied to a wide variety of problems including **classification, regression, density estimation,** etc.

Credit to CMU 10601

Decision Tree in the Wild

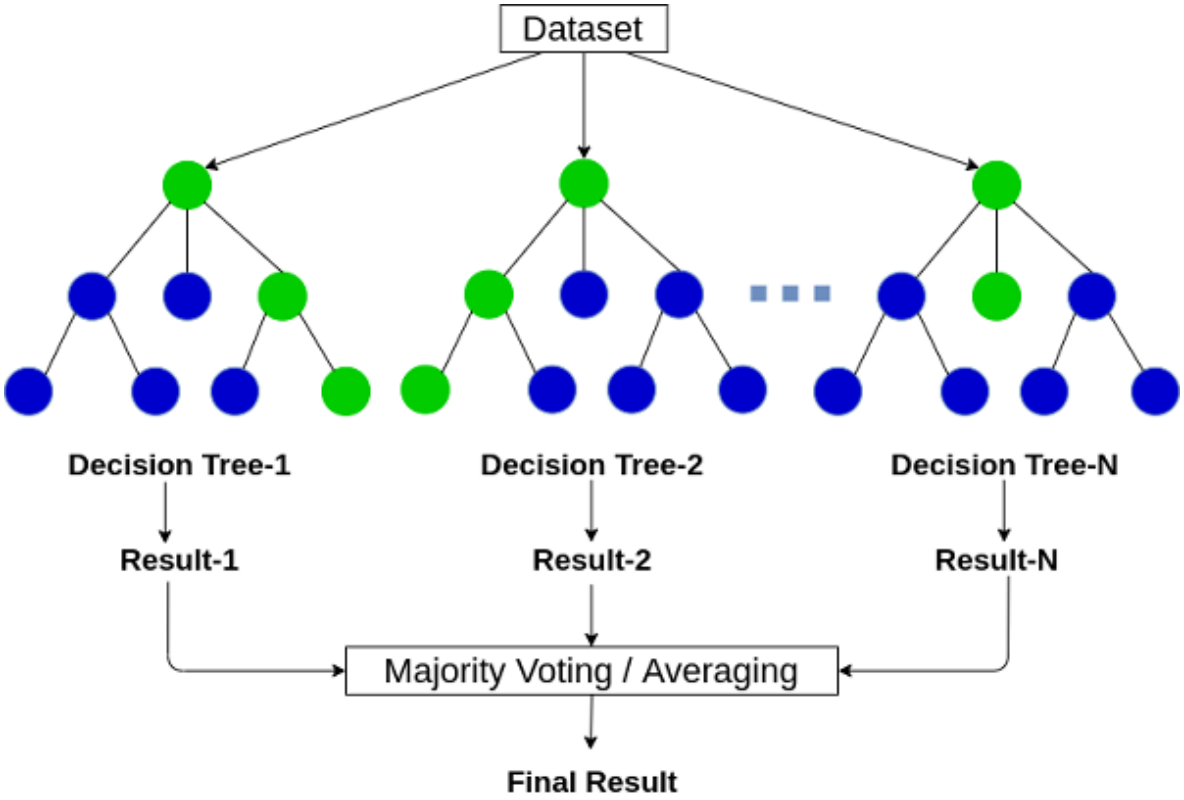
Applications of DTs include…

- medicine, molecular biology, text classification, manufacturing, astronomy, agriculture, and many others

- **Decision Forests** learn many DTs from random subsets of features; the result is a very powerful example of an **ensemble method**

Credit to CMU 10601

From A Decision Tree to Decision Forests



From A Decision Tree to Decision Forests

Improved Accuracy: Random forests combine the predictions from multiple decision trees, reducing the risk of overfitting and often leading to better accuracy.

Robustness to Noise and Overfitting: Individual decision trees can be sensitive to noise and overfitting, especially if they are deep. Random forests mitigate this by averaging multiple trees, thus balancing the overfitting of individual trees.

Pros

Feature Importance: Random forests provide a better understanding of feature importance through the aggregation of insights from multiple trees.

From A Decision Tree to Decision Forests

Complexity and Computational Cost: Random forests are more computationally intensive than single decision trees. They require more memory and processing power, and take longer to train and predict.

Less Interpretability: While a single decision tree is relatively easy to interpret, the complexity of a random forest model makes it much harder to visualize and understand.

Cons

Model Size: The model size of a random forest can be quite large due to the presence of many decision trees. This can be a drawback in environments where memory is a constraint.

Why Use *Deep* Decision Trees?

- Datasets are growing both in size *and* dimension
  - More rows *and* more columns
- To model high-dimensional data, we need to consider more splits
  - **More splits ⇒ deeper trees**

aggregates statistics to compute the node purity for candidate s

computes c sufficient statistics for each point **x**

$$Split(i) = \arg\max_{s \in \mathcal{S}} f\left(\sum_{\mathbf{x} \in \mathcal{I}} g(\mathbf{x}, s)\right)$$

split candidate maximizing node purity at node i

Set of possible split values

Data points in node i

$$Split(i) = \text{Compare}\left(f_j\right)$$

Best split among features stored in worker j

Yggdrasil: Column-Partitioning

- Workers compute sufficient statistics on ***all local features***

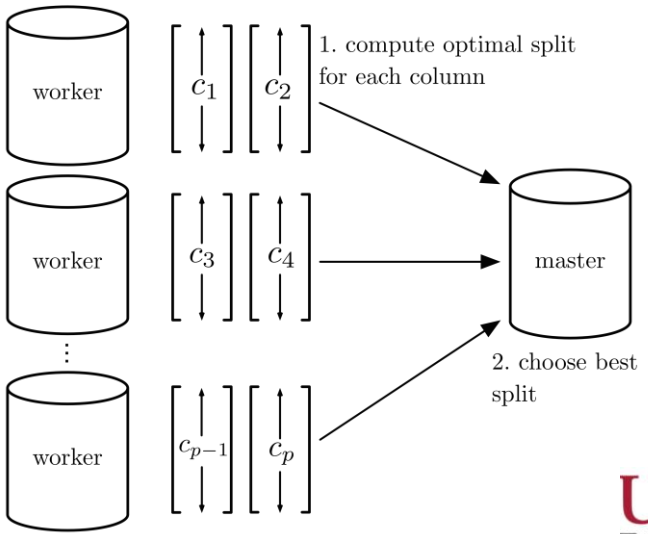
$$\vec{\mathbf{X}} = \left[\begin{array}{c|c|c|c} & & & \\ c_1 & \cdots\cdots & \cdots\cdots & c_p \\ & & & \end{array}\right] \in \mathbb{R}^{n \times p}$$

Yggdrasil:An Optimized System for Training Deep Decision Trees at Scale. FAbuzaid, J. Bradley, F. Liang, A. Feng, L. Yang, M. Zaharia, A. Talwalkar. NeurIPS 2016

Yggdrasil: Column-Partitioning

- Workers compute sufficient statistics on ***all local features***
- No approximation; all thresholds considered

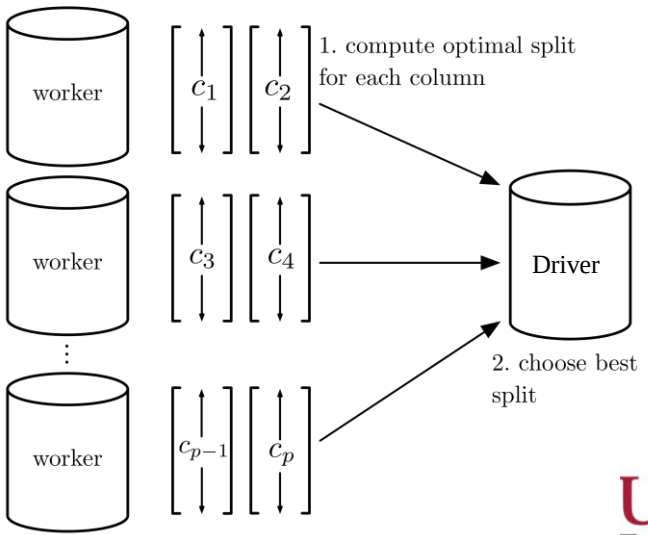
$$\vec{\mathbf{X}} = \left[\begin{array}{c|c|c|c} & & & \\ c_1 & \cdots\cdots & \cdots\cdots & c_p \\ & & & \end{array}\right] \in \mathbb{R}^{n \times p}$$



Yggdrasil: Column-Partitioning

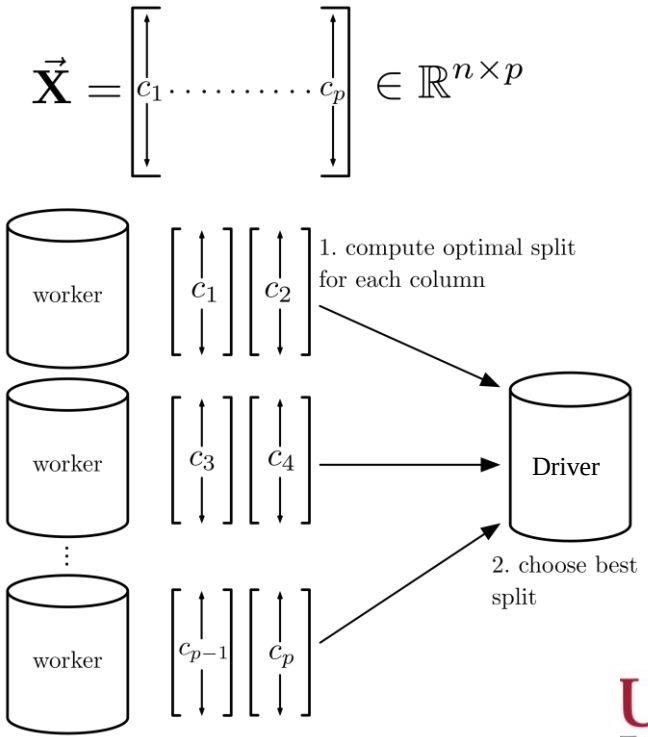
- Workers compute sufficient statistics on ***all local features***
- No approximation; all thresholds considered
- Driver picks best global split

$$\vec{\mathbf{X}} = \left[\begin{array}{ccc} \vdots & & \vdots \\ c_1 & \cdots & c_p \\ \vdots & & \vdots \end{array}\right] \in \mathbb{R}^{n \times p}$$



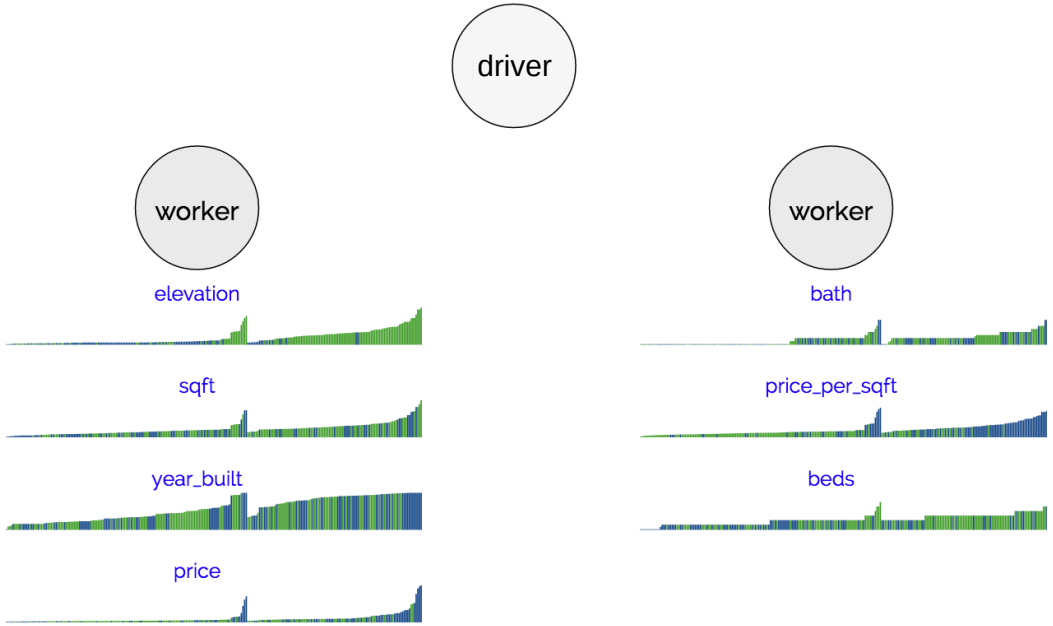
Yggdrasil: Column-Partitioning

- Workers compute sufficient statistics on ***all local features***
- No approximation; all thresholds considered
- Driver picks best global split
- One round of communication to update worker state at each tree level
  - Driver asks relevant workers about how datapoints are split, then shares results with all workers



Yggdrasil in Action

1. Partition features across workers
2. Workers sort each feature by value
3. Compute best split for each feature
4. Pick best split for each worker & send to driver
5. Driver selects best global split among the candidates, requests bit vector from worker
6. Driver broadcasts bit vector for best global split to all workers
7. Workers re-partition their features into sorted sub-arrays



Yggdrasil costs?

Notation

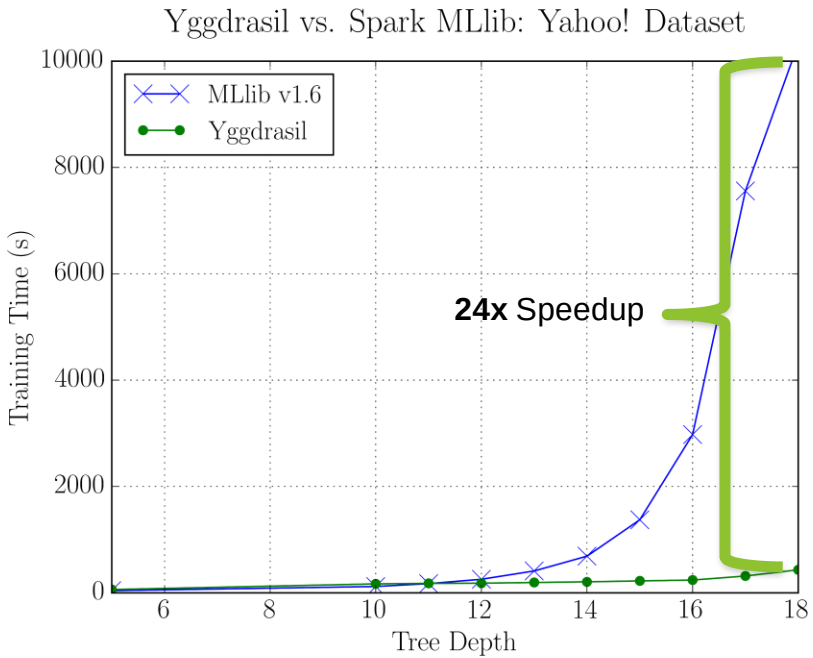
- n data points, p features, k workers
- B is number of split candidates for each feature (upper bounded by n)
- D is depth of tree (therefore roughly 2^D total tree nodes)

Computation: linear in n, p, and D, and is trivially parallelizable [same as PLANET]

Communication [much different than PLANET]

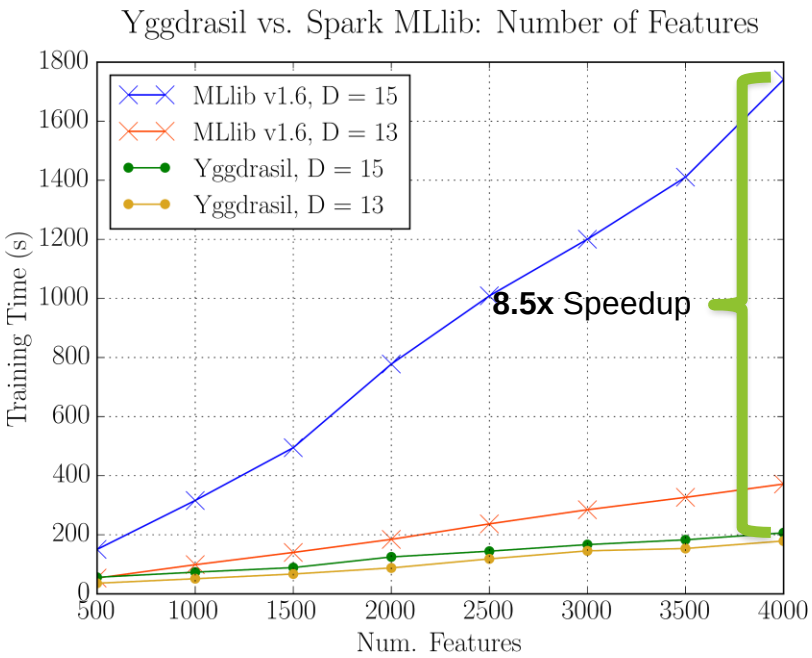
- workers communicate k tuples total per node, so **$O(2^D k)$** in total
- At each tree level, **O(n)** to compile and share results of splits to each worker
- Total communication is **$O(2^D k + D n k)$** [vs **$O(2^D B p k)$** for PLANET]
- No dependence on p; for large n, **O(D n k)** dominates

Results



- Regression task
- 2 million rows
- 3500 columns
- 52.2 GB (< 1% zeros)

Results



- 2 million rows
- Yggdrasil empirically outperforms Spark MLlib for deep trees and many features

Decision Tree Summary

What you can do now

- Define a decision tree classifier
- Interpret the output of decision trees
- Learn a decision tree classifier using greedy algorithm
- Traverse a decision tree to make predictions
  - Majority class predictions
- Tackle continuous and discrete features
- Understand how to control overfitting and underfitting in DT

Credit to Stanford CS 229

Important Decision Tree Interview Questions

Question: What is a decision tree and how does it work in machine learning?

Important Decision Tree Interview Questions

Question: What is a decision tree and how does it work in machine learning?

Answer: A decision tree is a **flowchart-like tree structure** where an internal node represents a feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome.

In the context of machine learning, it is used for both **classification** and **regression** tasks. The decision of making strategic splits heavily affects a tree's accuracy.

Important Decision Tree Interview Questions

Question: Discuss how decision trees can be used for both classification and regression.

Important Decision Tree Interview Questions

Question: Discuss how decision trees can be used for both classification and regression.

Answer: For classification, they predict discrete classes and splits are made in such a way that they best separate the classes.

For regression, decision trees predict continuous values. The tree is constructed similarly, but instead of voting for classes in the leaves, it predicts a numerical value, often **the mean of the target variable** of the instances in the leaf.

Important Decision Tree Interview Questions

Question: Can you explain overfitting in decision trees and how to avoid it?

Important Decision Tree Interview Questions

Question: Can you explain overfitting in decision trees and how to avoid it?

Answer: Overfitting occurs when the decision tree model **fits too closely** to the training data, including the **noise or random fluctuations** in the data. As a result, it performs poorly on unseen data.

To avoid overfitting, techniques such as **pruning** (removing parts of the tree that provide little power in classifying instances), **setting a minimum number of samples per leaf**, or maximum depth of the tree, and using random forests (an ensemble of decision trees) can be employed.

Important Decision Tree Interview Questions

Question: What are the main advantages and disadvantages of using decision trees?

Important Decision Tree Interview Questions

Question: What are the main advantages and disadvantages of using decision trees?

Answer: The advantages of decision trees are that they are easy to understand, **interpret**, and visualize. They can handle both numerical and categorical data and can model complex, non-linear relationships.

The disadvantages include a **tendency to overfit**, **sensitivity to small changes in the data**, and the problem of creating biased trees if some classes dominate.

Important Decision Tree Interview Questions

Question: Can you explain overfitting in decision trees and how to avoid it?

Important Decision Tree Interview Questions

Question: Can you explain overfitting in decision trees and how to avoid it?

Answer: Overfitting occurs when the decision tree model **fits too closely** to the training data, including the **noise or random fluctuations** in the data. As a result, it performs poorly on unseen data.

To avoid overfitting, techniques such as **pruning** (removing parts of the tree that provide little power in classifying instances), **setting a minimum number of samples per leaf**, or maximum depth of the tree, and using random forests (an ensemble of decision trees) can be employed.

Important Decision Tree Interview Questions

Question: What is pruning in a decision tree and why is it important?

Important Decision Tree Interview Questions

Question: What is pruning in a decision tree and why is it important?

Answer: Pruning is a technique used to **reduce the size of decision trees** by removing sections of the tree that are not necessary for classification.

This is done to reduce the complexity of the final classifier, which helps in alleviating overfitting.

Pruning can be done in two ways: **pre-pruning**, which involves setting conditions for stopping tree construction early; and **post-pruning**, which involves removing branches from the fully grown tree.

Model Metrics

- Loss measures how good the model in predicting the outcome in supervised learning
- Other metrics to evaluate the model performance
  - Model specific: e.g., accuracy for classification, mAP for object detection
  - Business specific: e.g., revenue, inference latency
- We select models by multiple metrics
  - Just like how you choose cars

Metrics for Binary Classification

- Accuracy: # correct predictions / # examples

```
sum(y == y_hat) / y.size
```

- Precision: #True positive / # (True positive + False positive)

```
sum((y_hat == 1) & (y == 1)) / sum(y_hat == 1)
```

- Recall: #True positive / # Positive examples

```
sum((y_hat == 1) & (y == 1)) / sum(y == 1)
```

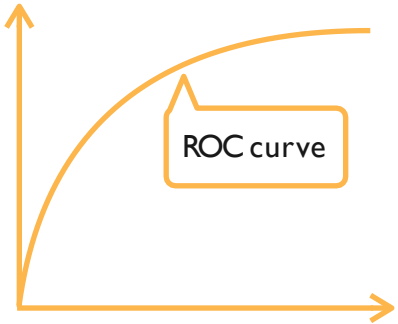
- Be careful of division by 0
- One metric that balances precision and recall
  - F1: the harmonic mean of precision and recall: $2pr/(p + r)$

AUC-ROC

- Measures how well the model can separate the two classes
- Choose decision threshold θ, predict positive if $o \geq \theta$ else neg
- In the range [0.5, 1]

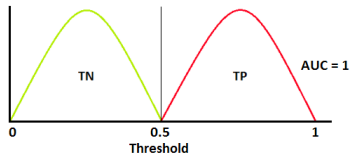
$$\text{TPR} = \frac{\#\text{True Pos predictions}}{\#\text{pos examples}}$$

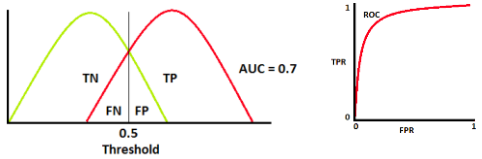
$$\text{FPR} = \frac{\#\text{False Pos predictions}}{\#\text{neg examples}}$$



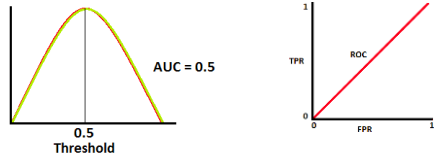
Distribution of output O

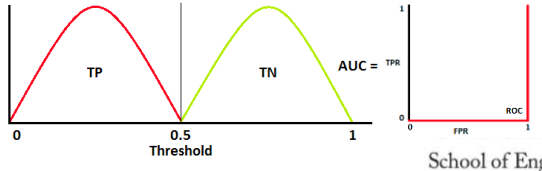
Ideal separation





Non separable





Classical ML Algorithms

k Nearest Neighbor

Nearest Neighbor Algorithm

def train($\mathcal{D}$):

Store $\mathcal{D}$

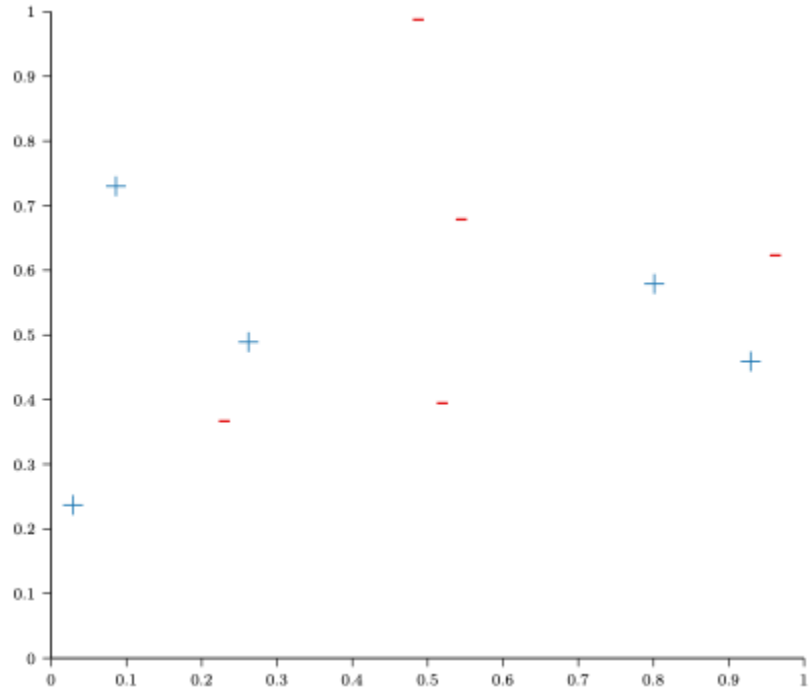
def h(x'):

Let $x^{(i)}$ = the point in $\mathcal{D}$ that is nearest to x'

return $y^{(i)}$

Credit to CMU 10601

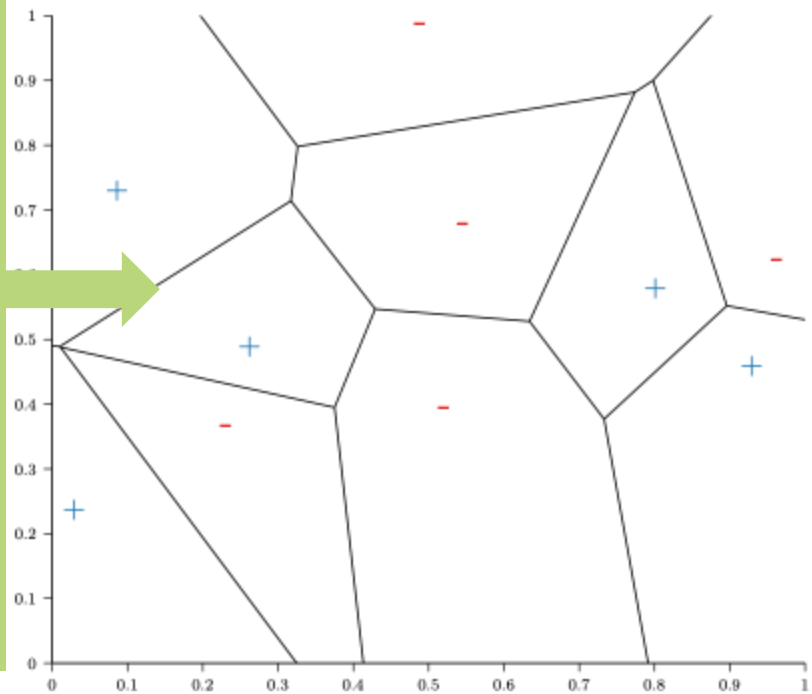
Nearest Neighbor Algorithm



Credit to CMU 10601

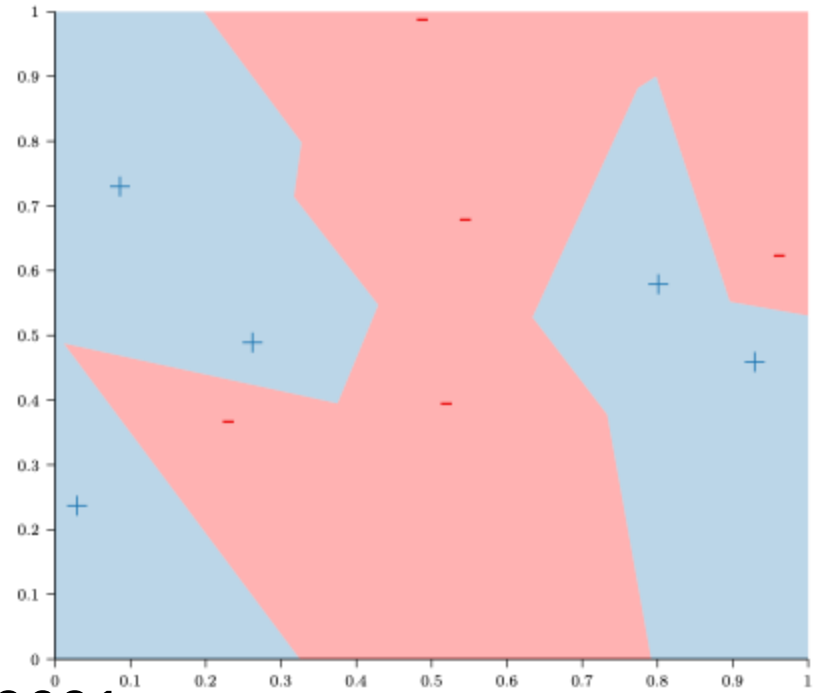
Nearest Neighbor Algorithm

- This is a **Voronoi diagram**
- Each **cell** contain one of our training examples
- **All points within a cell** are **closer** to that training example, than to any other training example
- **Points on the Voronoi line segments** are **equidistant** to one or more training examples



Credit to CMU 10601

Nearest Neighbor Decision Boundary



Credit to CMU 10601

Nearest Neighbor Algorithm

- Requires no training!
- Always has zero training error!
  - ***A data point is always its own nearest neighbor***

5
7

Credit to CMU 10601

k Nearest Neighbor Algorithm

def set_hyperparameters(k, d):
    Store k
    Store d(·, ·)

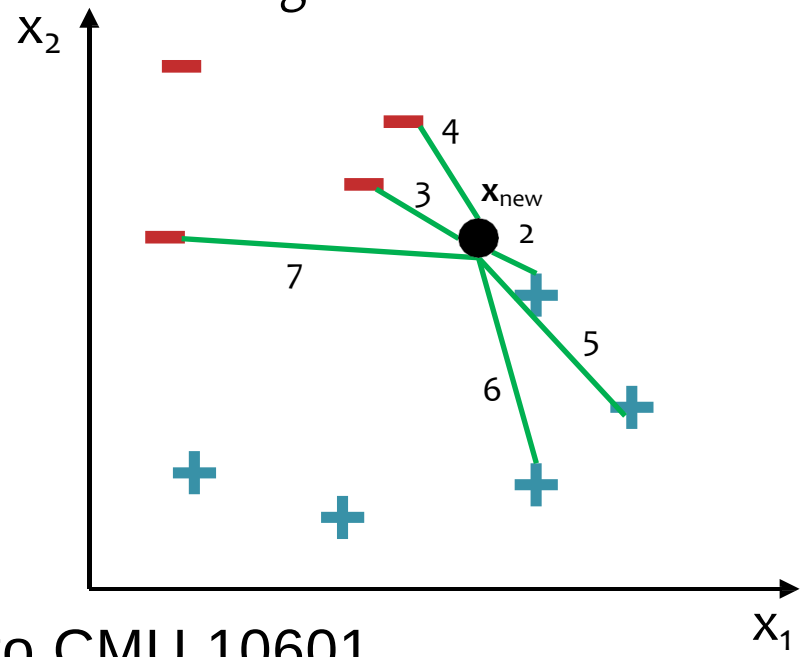
def train($\mathcal{D}$):
    Store $\mathcal{D}$

def h(x'):
    Let S = the set of k points in $\mathcal{D}$ nearest to x'
        according to distance function
        d(**u**, **v**)
    Let v = majority_vote(S)
    return v

Credit to CMU 10601

k Nearest Neighbor Algorithm

Suppose we have the training dataset below.



How should we label the new point?

It depends on k:

if k=2, $h(\mathbf{x}_{new}) = +1$

if k=3, $h(\mathbf{x}_{new}) = -1$

if k=5, $h(\mathbf{x}_{new}) = +1$

Credit to CMU 10601

k Nearest Neighbor Algorithm

Distance Functions:

- KNN requires a **distance function**

$$d : \mathbb{R}^M \times \mathbb{R}^M \rightarrow \mathbb{R}$$

- The most common choice is **Euclidean distance**

$$d(\boldsymbol{u}, \boldsymbol{v}) = \sqrt{\sum_{m=1}^{M} (u_m - v_m)^2}$$

- But there are other choices (e.g. **Manhattan distance**)

$$d(\boldsymbol{u}, \boldsymbol{v}) = \sum_{m=1}^{M} |u_m - v_m|$$

Credit to CMU 10601

How Does k Change the Decision?

- What is the effect of k?
- Will a large k make the decision boundary coarse or smooth?
- How will that affect the overfitting and underfitting?

kNN on Fisher Iris Data

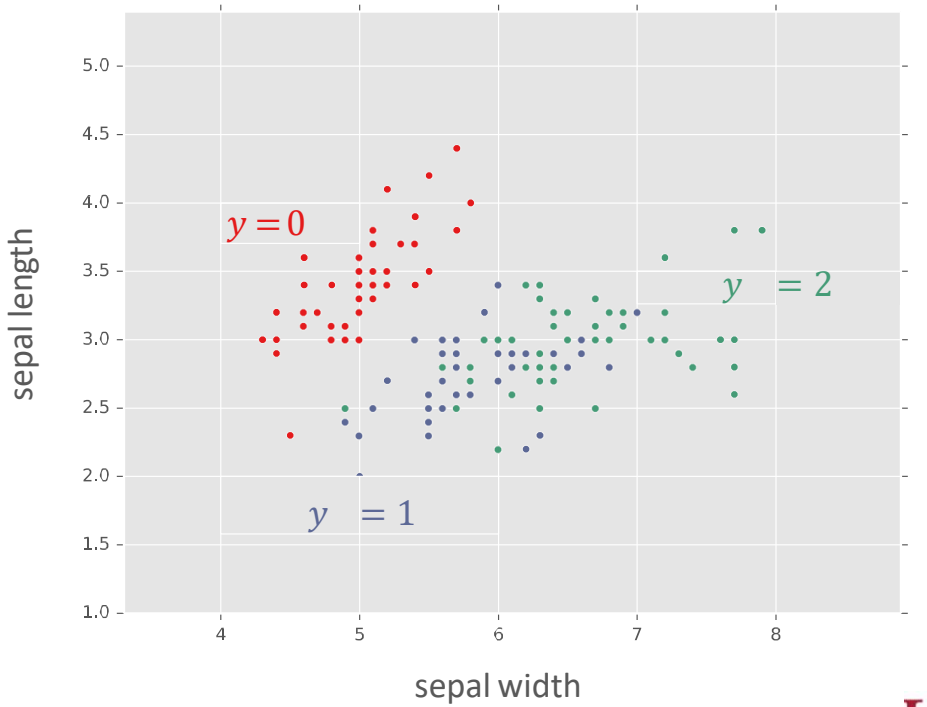


Figure courtesy of Matt Gormley

k=1

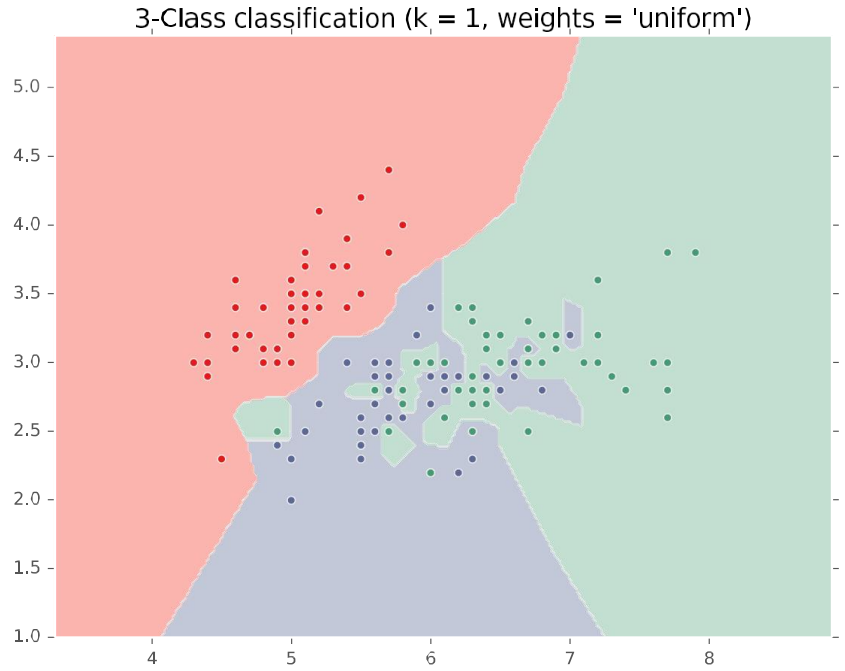


Figure courtesy of Matt Gormley

k=2

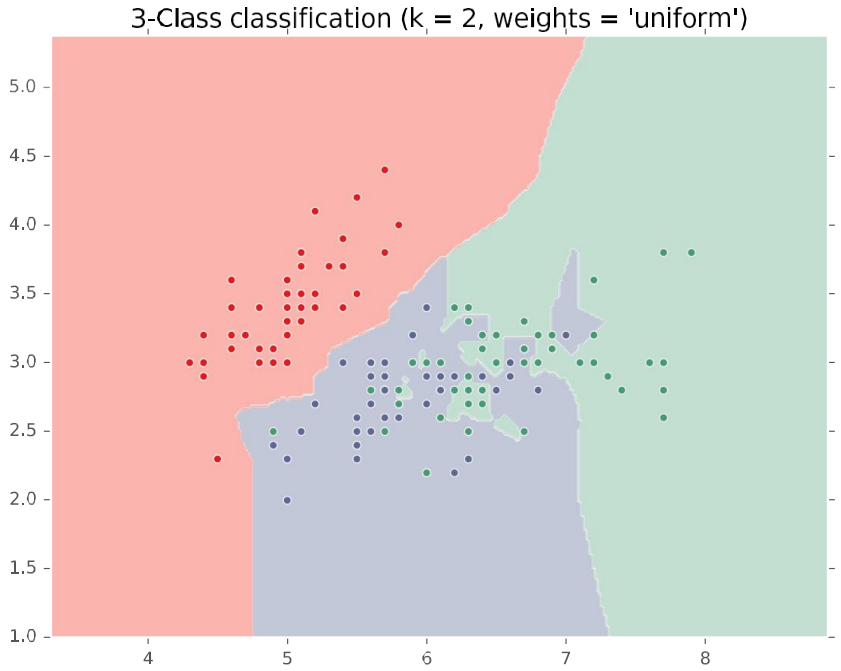


Figure courtesy of Matt Gormley

k=3

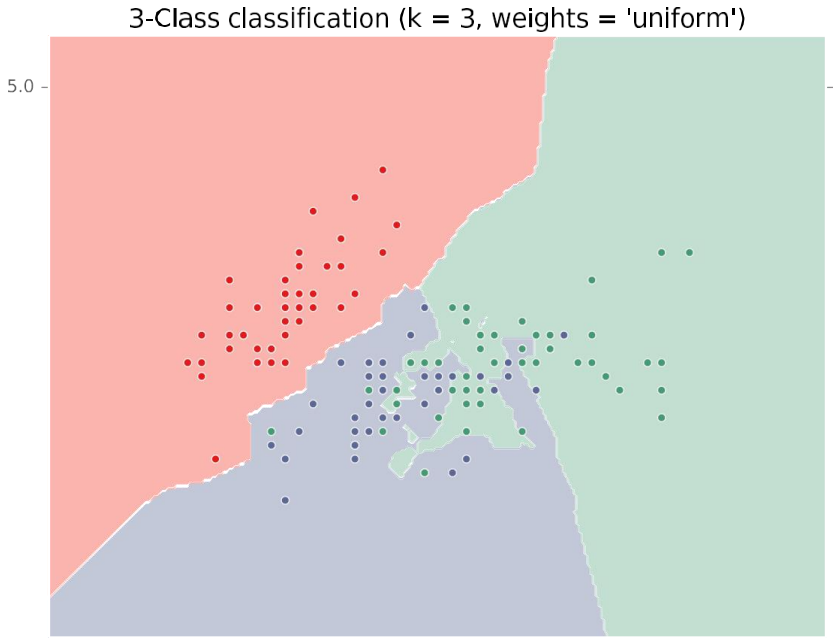


Figure courtesy of Matt Gormley

k=5

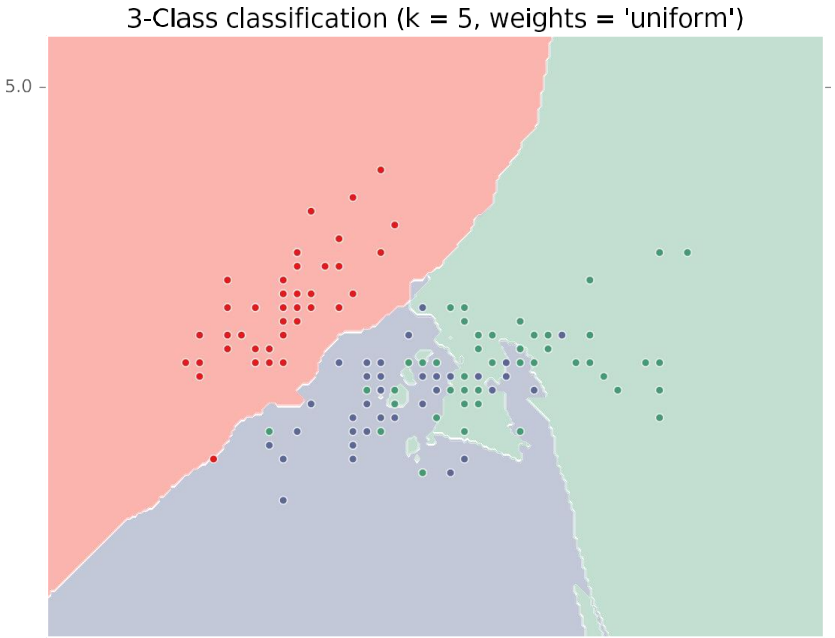


Figure courtesy of Matt Gormley

k=10

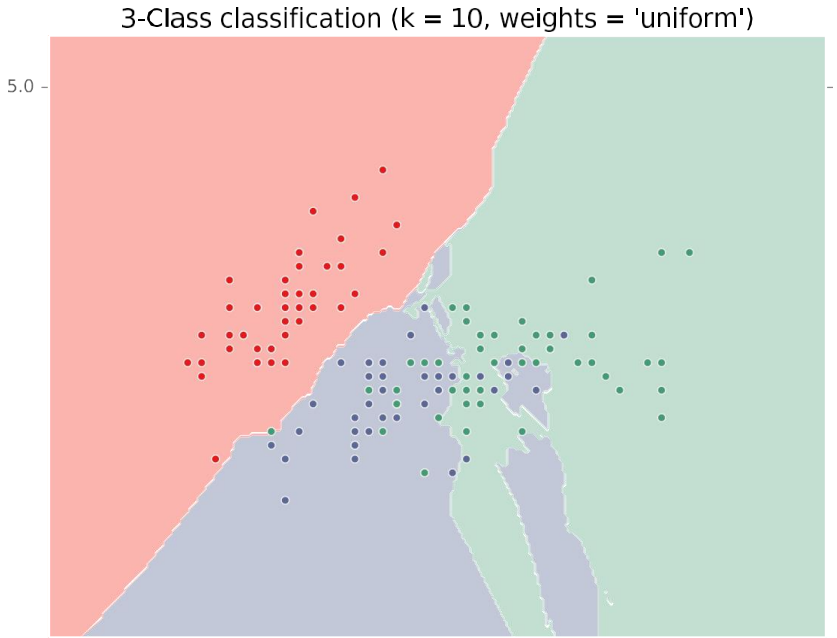


Figure courtesy of Matt Gormley

k=20

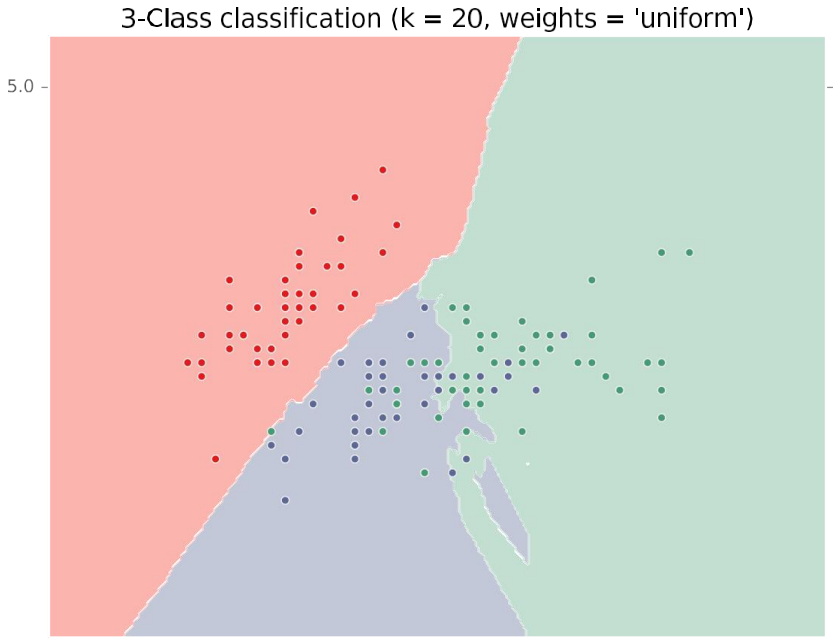


Figure courtesy of Matt Gormley

k=30

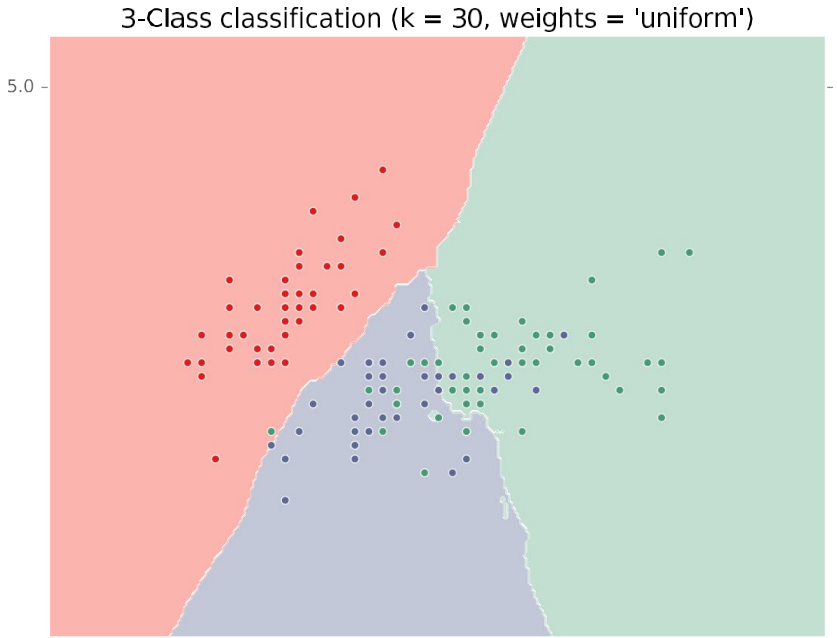
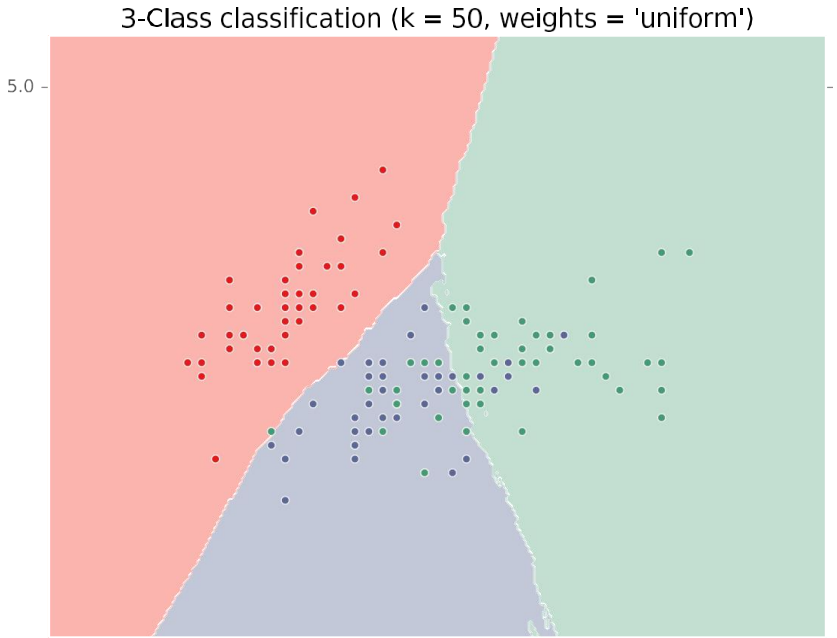
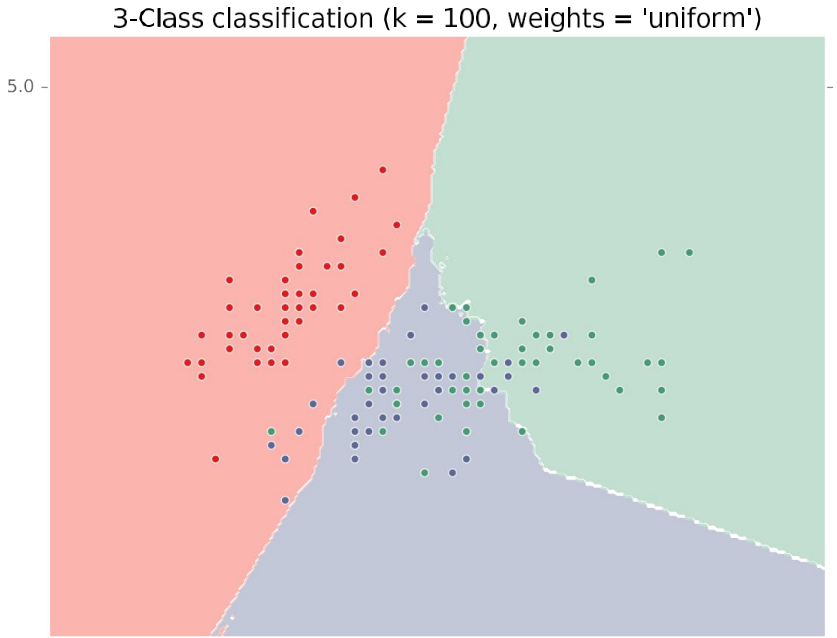


Figure courtesy of Matt Gormley

k=50



k=100



k=120

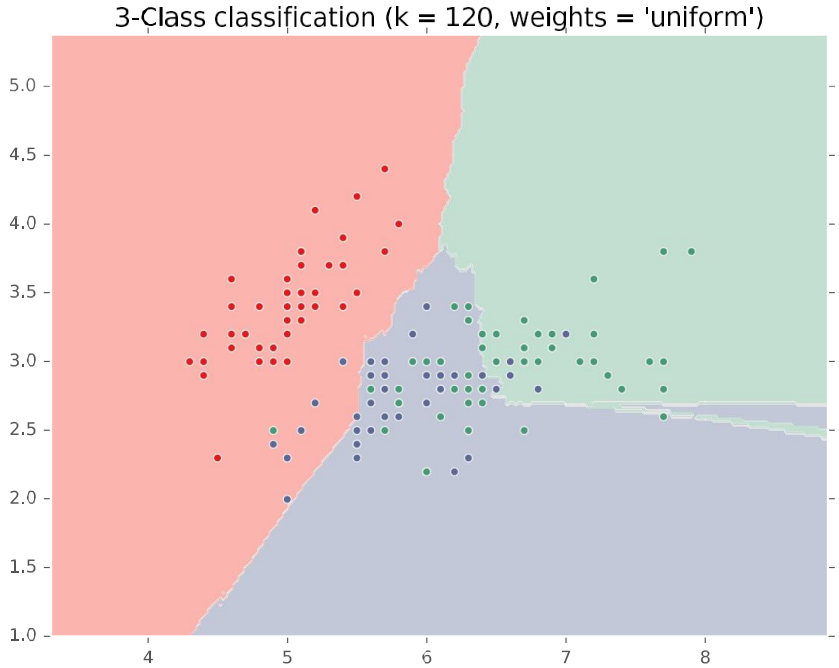


Figure courtesy of Matt Gormley

k=150

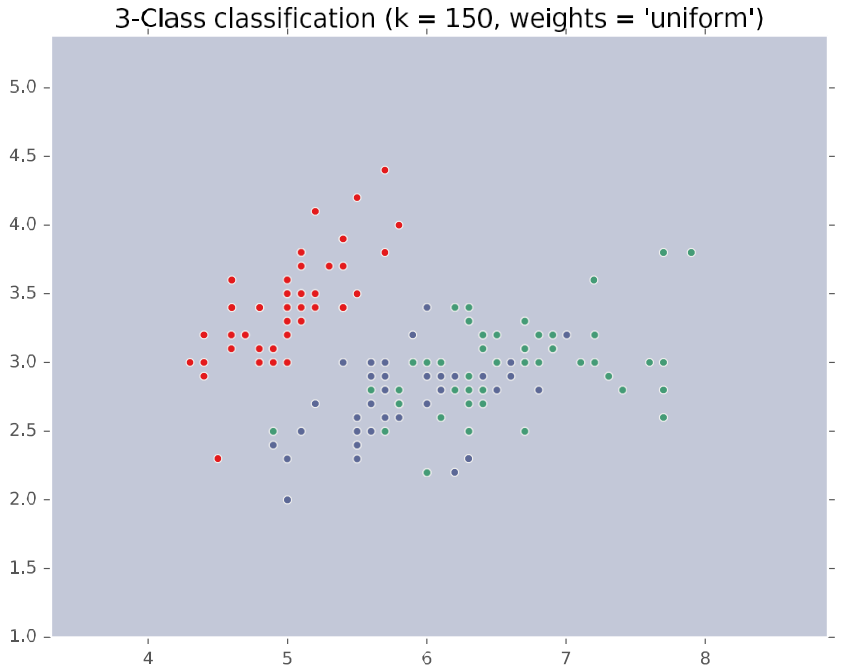
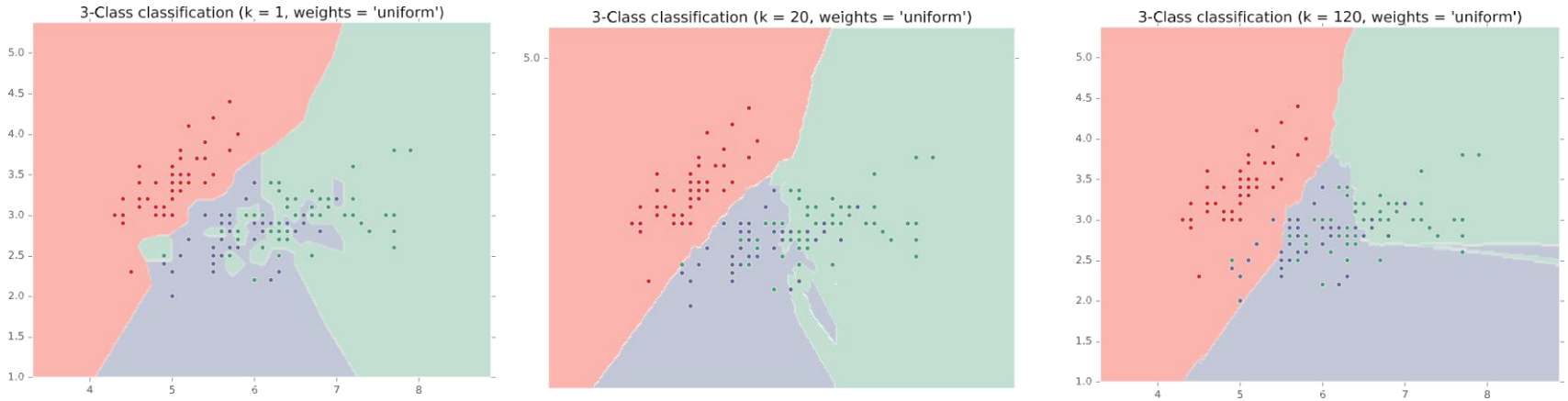


Figure courtesy of Matt Gormley

k=1, 20, 120



Credit to CMU 10601

Figure courtesy of Matt Gormley

Deciding Hyperparameter

What is the best value of **k** to use?

What is the best **distance** to use?

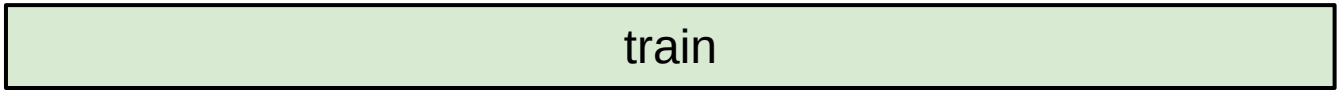
These are **hyperparameters**: choices about the algorithms themselves.

Very problem/dataset-dependent.

Must try them all out and see what works best.

Deciding Hyperparameter

Idea #1: Choose hyperparameters that work best on the **training data**

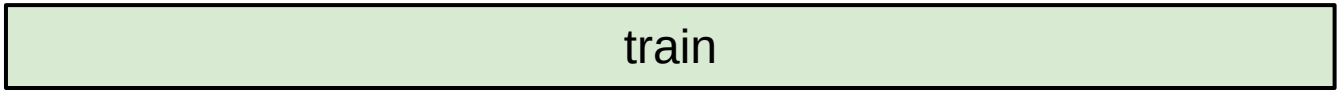


Credit to Stanford CS 229

Deciding Hyperparameter

Idea #1: Choose hyperparameters that work best on the **training data**

BAD: K = 1 always works perfectly on training data

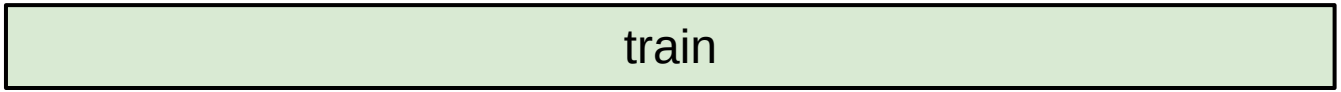


Credit to Stanford CS 229

Deciding Hyperparameter

Idea #1: Choose hyperparameters that work best on the **training data**

BAD: K = 1 always works perfectly on training data



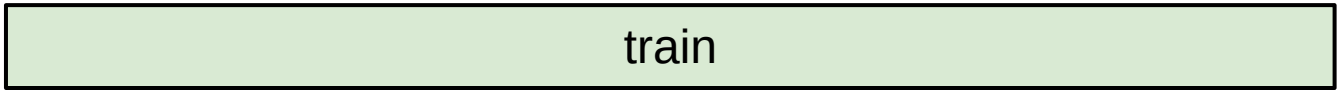
Idea #2: choose hyperparameters that work best on **test** data

Credit to Stanford CS 229

Deciding Hyperparameter

Idea #1: Choose hyperparameters that work best on the **training data**

BAD: K = 1 always works perfectly on training data



Idea #2: choose hyperparameters that work best on **test** data

BAD: No idea how algorithm will perform on new data

Never do this!

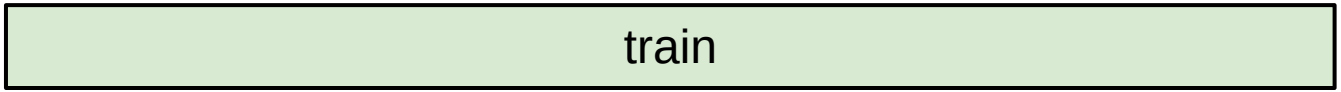
Credit to Stanford CS 229

Deciding Hyperparameter

Credit to Stanford CS 229

Idea #1: Choose hyperparameters that work best on the **training data**

BAD: K = 1 always works perfectly on training data



Idea #2: choose hyperparameters that work best on **test** data

BAD: No idea how algorithm will perform on new data

Idea #3: Split data into **train**, **val**; choose hyperparameters on val and evaluate on test

Better!

Deciding Hyperparameter

train

Idea #4: **Cross-Validation**: Split data into **folds**,
try each fold as validation and average the results

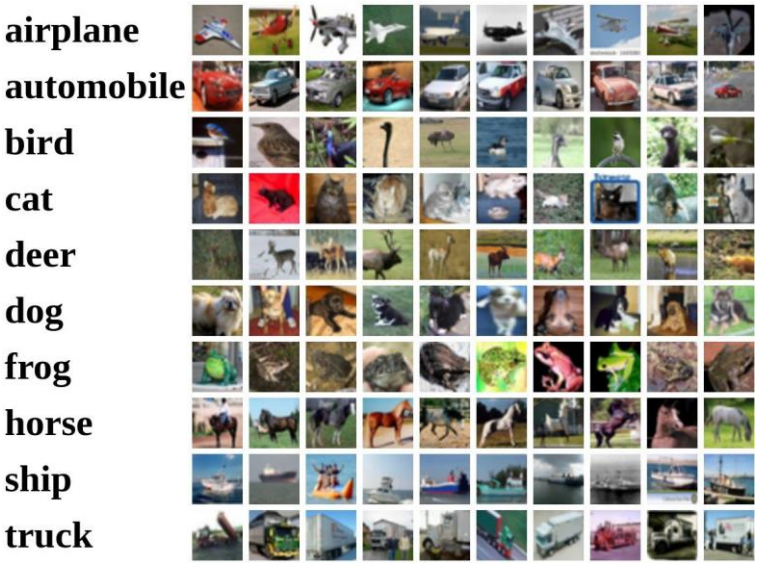
fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test

Useful for small datasets, but not used too frequently in deep learning

Credit to Stanford CS 229

Example Dataset: **CIFAR10**

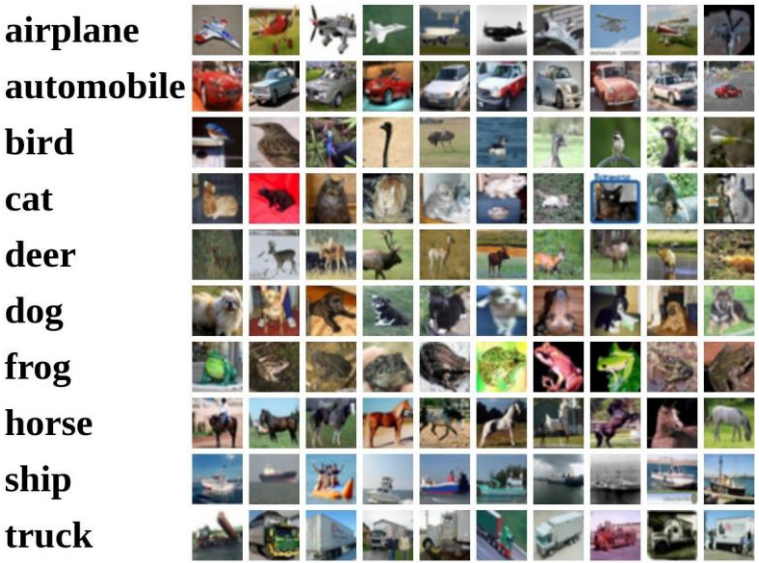
10 classes
50,000 training images
10,000 testing images



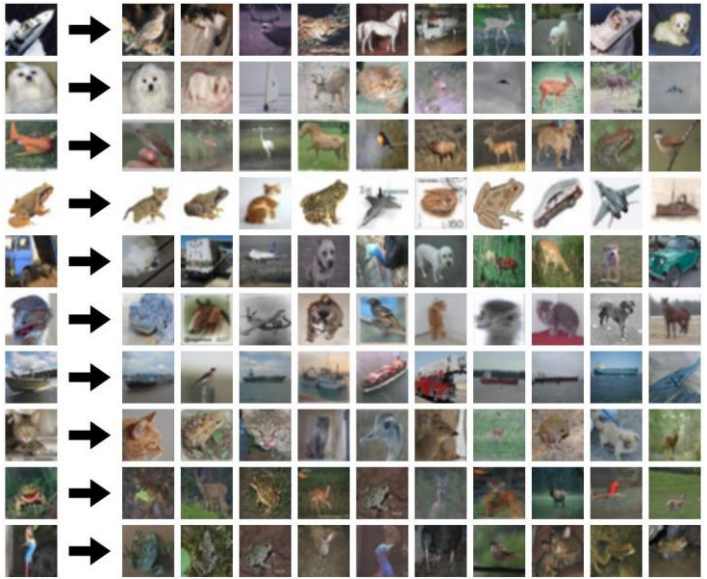
Alex Krizhevsky, "Learning Multiple Layers of Features from Tiny Images", Technical Report, 2009.

Example Dataset: **CIFAR10**

10 classes
50,000 training images
10,000 testing images

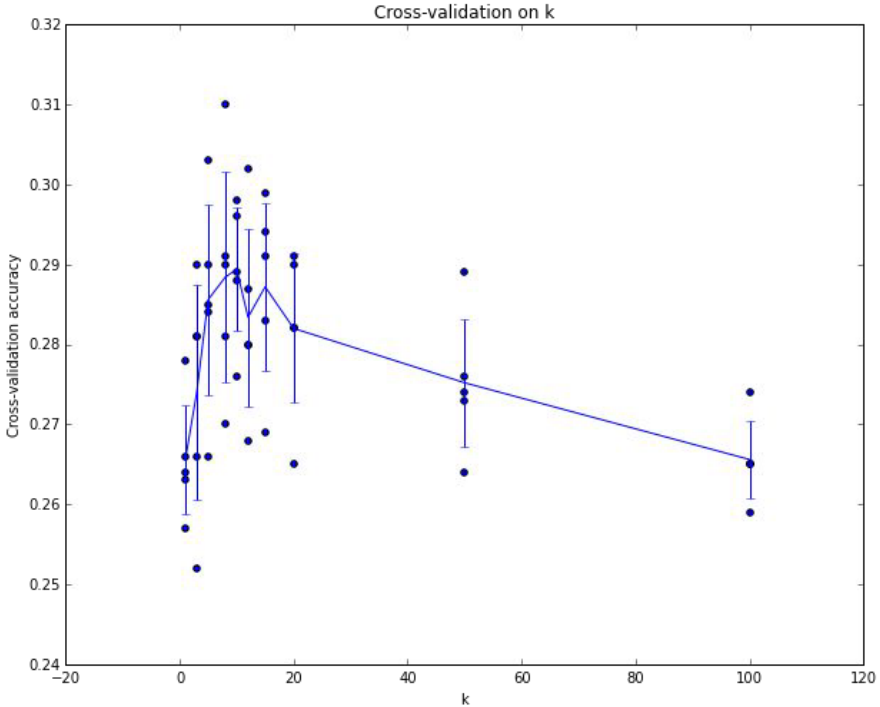


Test images and nearest neighbors



Alex Krizhevsky, "Learning Multiple Layers of Features from Tiny Images", Technical Report, 2009.

Deciding Hyperparameter



Example of
5-fold cross-validation
for the value of **k.**

Each point: single outcome.

The line goes through the mean, bars indicated standard deviation

(Seems that k ~= 7 works best for this data)

Credit to Stanford CS 229

How Does k Change the Decision?

- What is the effect of k? **control the model capacity/complexity**
- Will a large k make the decision boundary coarse or smooth? **Smoother**
- How will that affect the overfitting and underfitting? **Small k leads to overfitting and large k leads to underfitting**

How do you decide k properly? Also, how to decide these (hyper)parameters of ML models?

- one idea is to monitor the error rate/accuracy

- more systematic approaches are deferred to the later lecture: automated ML, **cross-validation**, Bayesian optimization, etc.,

- $\sqrt{N}$ (the number of total samples)

kNN Summary

- Pros:
  - Intuitive / explainable
  - No training/retraining
  - Provably near-optimal in terms of true error rate
- Cons:
  - Computationally expensive
    - Always needs to store all data: $O(ND)$
    - Finding the k closest points in D dimensions: $O(ND + N\log(k))$
      - Can be sped up through clever use of data structures (trades off training and test costs)
      - Can be approximated using stochastic methods
  - Affected by feature scale Intuitive/explainable

Important kNN Interview Questions

Question: What are the advantages and disadvantages of KNN?

Important kNN Interview Questions

Question: What are the advantages and disadvantages of KNN?

Answer: Advantages of KNN include its **simplicity**, effectiveness, and the fact that it requires **no training** phase. It's versatile, being used for both classification and regression.

However, KNN has several disadvantages like being **computationally expensive**, particularly with large datasets, as it requires storing all the training data. It's also **sensitive to the scale of the data and irrelevant features**, which can significantly degrade its performance.

Important kNN Interview Questions

Question: How does the choice of distance metric affect the performance of KNN?

Important kNN Interview Questions

Question: How does the choice of distance metric affect the performance of KNN?

Answer: The choice of distance metric in KNN can significantly affect its performance. Common distance metrics include Euclidean, Manhattan, and Minkowski distances. Euclidean distance works well in many cases, especially when the features are in the same units. Manhattan distance is more suitable for high-dimensional data. The choice of distance metric should match the data's structure and the problem's nature.

Treat it as a hyperparameter as K.

Important kNN Interview Questions

Question: Can KNN be used for regression? How?

Important kNN Interview Questions

Question: Can KNN be used for regression? How?

Answer: Yes, KNN can be adapted for regression. In KNN regression, instead of voting for a class, the algorithm calculates the **average (or sometimes the median) of the values of its k nearest neighbors**. This average is then used as the predicted value for the input instance.

Classical ML Algorithms

Clustering

What is Clustering?

Discover groups (*clusters*) of related articles

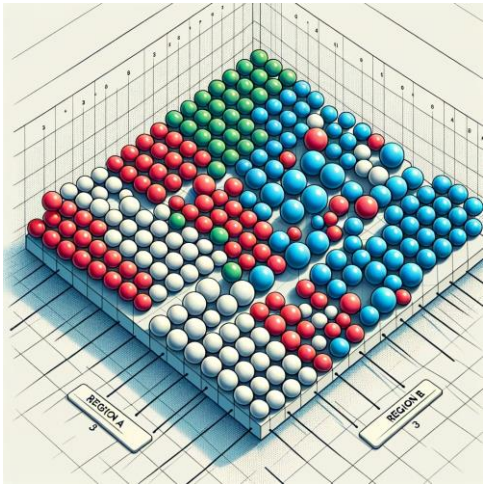
SPORTS

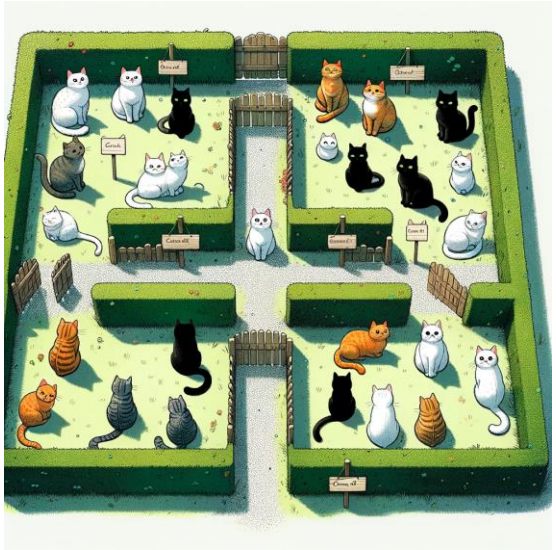
WORLD NEWS

Credit to Stanford CS 229

What is Clustering?

Discover groups (*clusters*) of related items

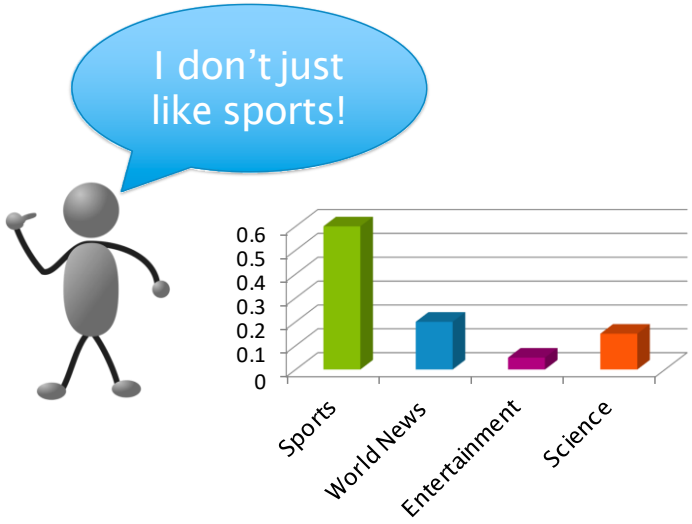




Credit to Stanford CS 229

Clustering

Why might clustering be useful?

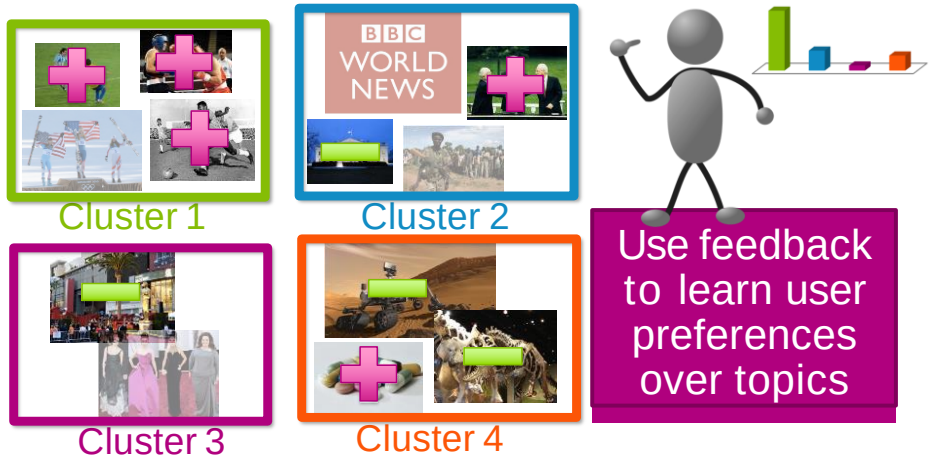


Credit to Stanford CS 229

Clustering

Learn user preferences

Set of clustered documents read by user



Credit to Stanford CS 229

Clustering

What if some of the labels are known?

Training set of labeled docs

Clustering: An unsupervised learning task (X only no Y)

Credit to Stanford CS 229

Clustering

Clustering

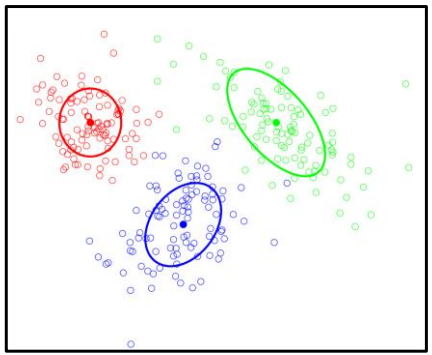
No labels provided

…uncover cluster structure from input alone

Input: docs as vectors x_i

Output: cluster labels z_i

An unsupervised learning task



Credit to Stanford CS 229

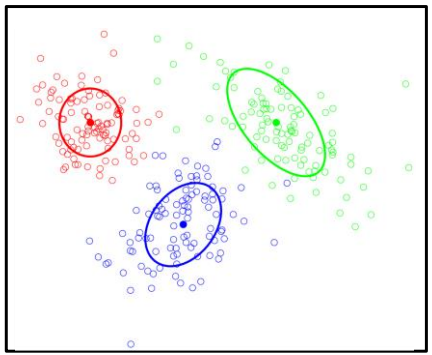
Clustering

What defines a cluster?

Cluster defined by center & shape/spread

Assign observation x_i (doc) to cluster k (topic label) if

- Score under cluster k is higher than under others
- For simplicity, often define score as distance to cluster center (ignoring shape)



Credit to Stanford CS 229

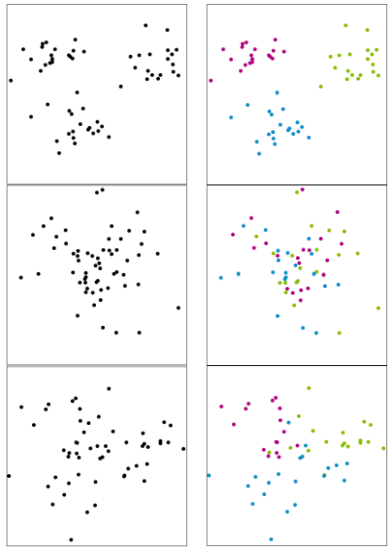
Clustering

Hope for unsupervised learning

Easy

Impossible

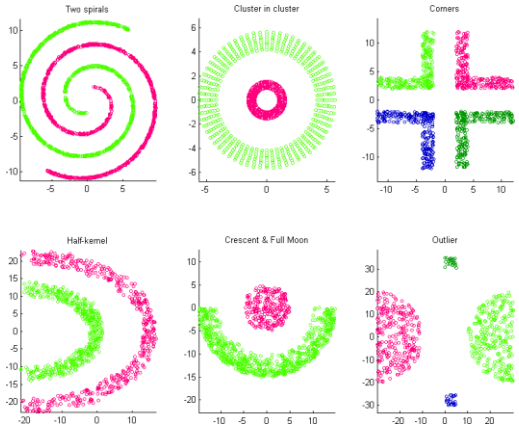
In between



Credit to Stanford CS 229

Clustering

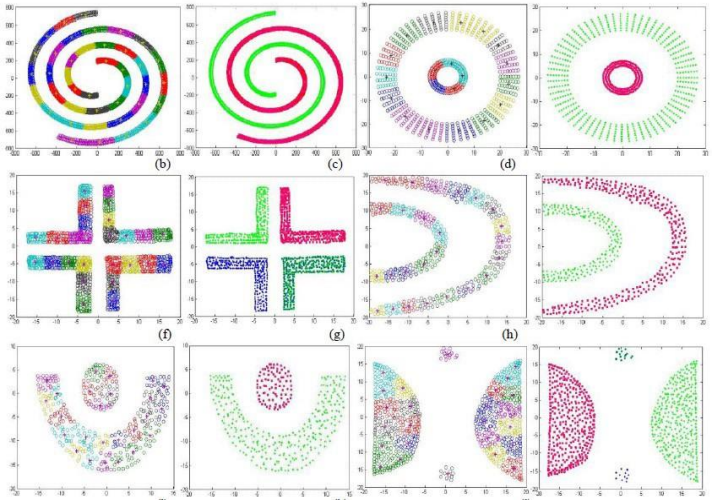
Other (challenging!) clusters to discover…



Credit to Stanford CS 229

Clustering

Other (challenging!) clusters to discover…



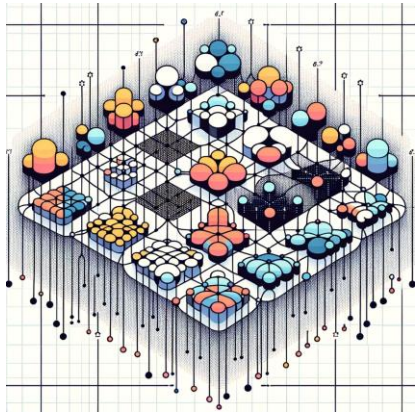
Credit to Stanford CS 229

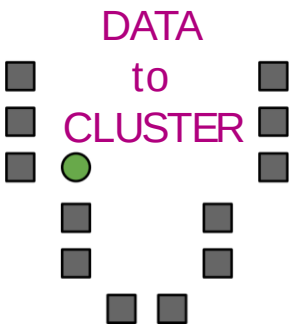
Clustering: k-means

k- means

Assume

- Score= distance to cluster center (smaller better)



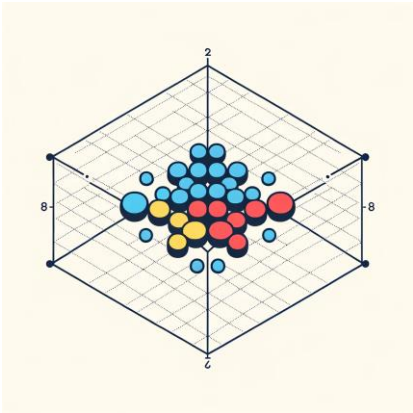


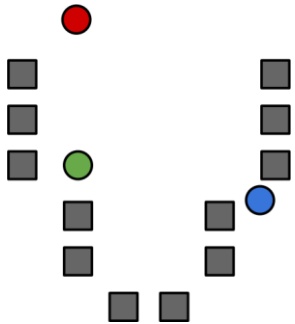
Credit to Stanford CS 229

Clustering: k-means

k-means algorithm

0. Initialize cluster centers
   $\mu_1, \mu_2, \ldots, \mu_k$





Credit to Stanford CS 229

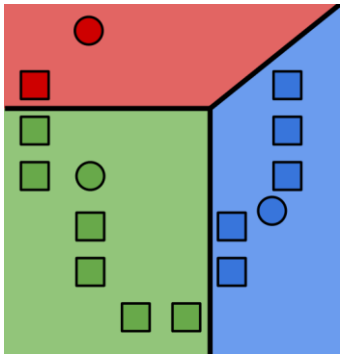
Clustering: k-means

k-means algorithm

1. Initialize cluster centers
2. Assign observations to closest cluster center

$$z_i \leftarrow \arg\min_j ||\mu_j - \mathbf{x}_i||_2^2$$

Inferred label for obs i, whereas supervised learning has given label y_i



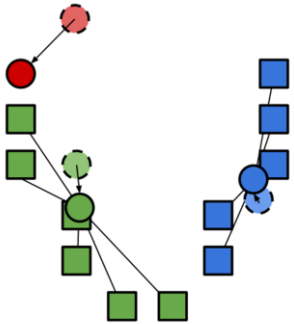
Credit to Stanford CS 229

Clustering: k-means

k-means algorithm

1. Initialize cluster centers
2. Assign observations to closest cluster center
3. Revise cluster centers as mean of assigned observations

$$\mu_j = \frac{1}{n_j} \sum_{i:z_i=j} \mathbf{x}_i$$

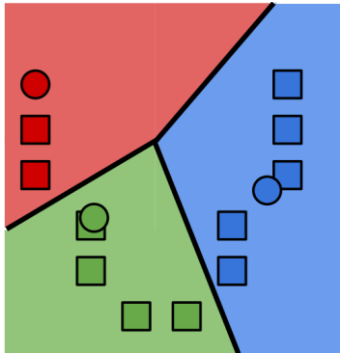


Credit to Stanford CS 229

Clustering: k-means

k-means algorithm

1. Initialize cluster centers
2. Assign observations to closest cluster center
3. Revise cluster centers as mean of assigned observations
4. Repeat 1.+2. until convergence



Credit to Stanford CS 229

Clustering: k-means

Limitations of k-means

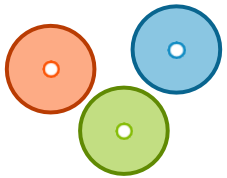
Assign observations to closest cluster center

$$z_i \leftarrow \arg\min_j \|\mu_j - \mathbf{x}_i\|_2^2$$

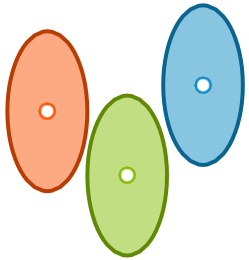
Only center matters

Can use weighted Euclidean, but requires *known* weights

Equivalent to assuming spherically symmetric clusters



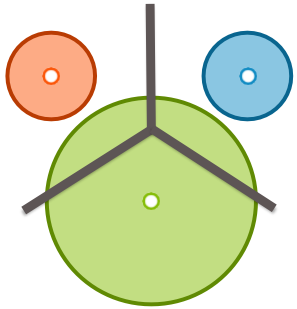
Still assumes all clusters have the same axis-aligned ellipses



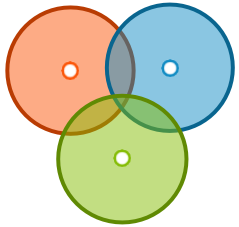
Credit to Stanford CS 229

Clustering: k-means

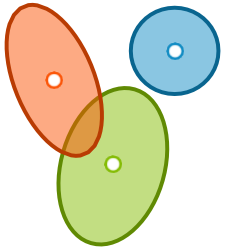
Failure modes of k-means



disparate cluster sizes



overlapping clusters



different shaped/oriented clusters

Credit to Stanford CS 229

Clustering: k-means

What you can do now…

- Describe the input (unlabeled observations) and output (labels) of a clustering algorithm
- Determine whether a task is supervised or unsupervised
- Cluster documents using k-means
- Describe potential applications of clustering

Credit to Stanford CS 229

Important Clustering Interview Questions

Question: What are the main challenges in clustering analysis?

Important Clustering Interview Questions

Question: What are the main challenges in clustering analysis?

Answer: The main challenges include:

- **Determining the Number of Clusters**: It's often not clear how many clusters should be chosen, especially in K-Means.
- **Sensitivity to Initial Conditions**: Some algorithms, like K-Means, are sensitive to the initial choice of centroids.
- **Noise and Outlier Sensitivity**: Some clustering algorithms are sensitive to noise and outliers in the data.
- **Scalability**: Large datasets pose computational challenges.

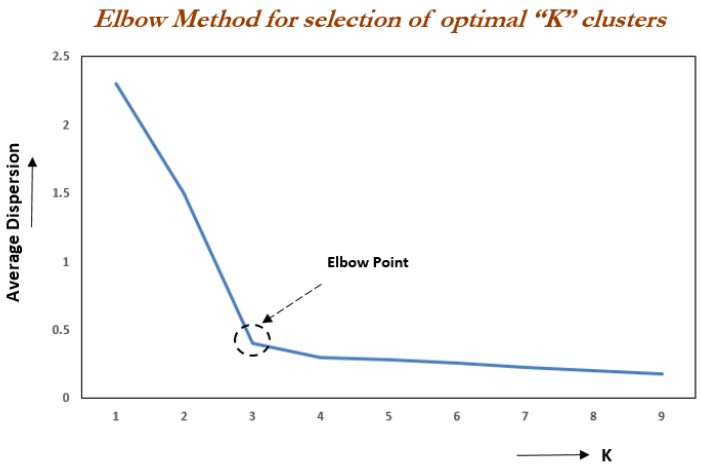
Important Clustering Interview Questions

Question: How to Decide k in kmeans?

Important Clustering Interview Questions

Question: How to Decide k in kmeans?

Answer: Elbow Method:



- Run K-Means cluste alues of 'k' (for example, from 1 to 1
- For each value of 'k' um of Square (WSS).
- Plot the curve of WS ers k.
- The "elbow" in the pl sh more slowly, is considered a good ir of clusters.

Important Clustering Interview Questions

Question: How to initialize k-means?

Important Clustering Interview Questions

Question: How to initialize k-means?

Answer:

Random Initialization

- Centroids are chosen randomly from the data points. This method is simple but can lead to poor convergence if centroids are not well chosen.

K-Means++:

- One centroid is chosen uniformly at random from the data points.
- Remaining centroids, choose data points as new centroids with probability proportional to their squared distance from the nearest existing centroid.

This tends to spread out the initial centroids, leading to better convergence.

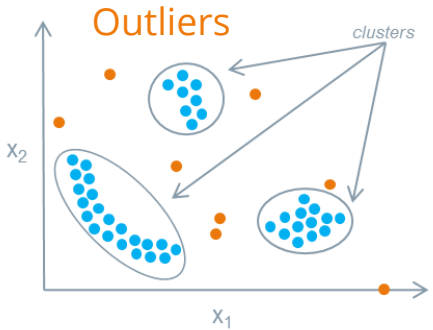
Classical ML Algorithms

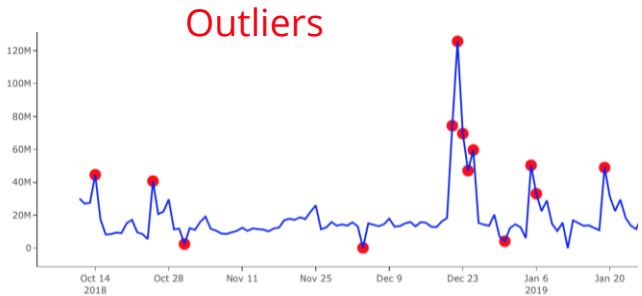
Outlier/Anomaly Detection

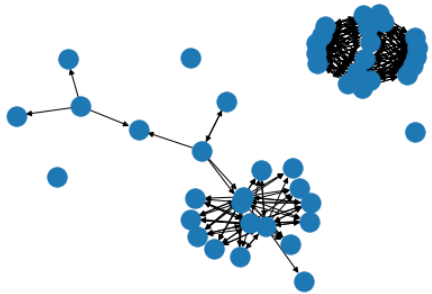
Outlier Detection

Outlier detection (**OD**), also known as anomaly detection (AD)*, aims to identify the data samples that are **deviant** from the **general distribution**.

This process is often ***unsupervised*** in practice.







Outliers in Tabular Data

Source: https://developer.mindsphere.io/apis/analytics-anomalydetection/api-anomalydetection-overview.html

Outliers in Time-series

Source: https://towardsdatascience.com/anomaly-detection-with-isolation-forest-visualization-23cd75c281e2

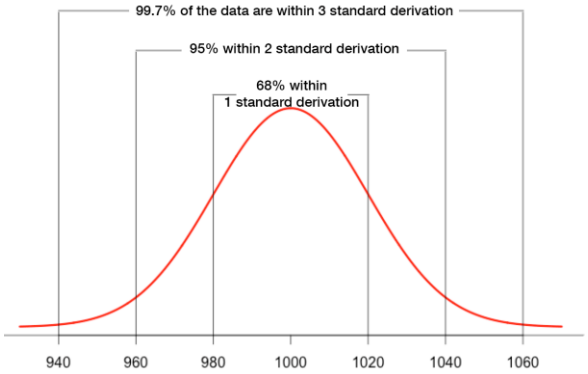
Outliers in Graph

https://blog.munhou.com/2021/03/27/Network-Anomaly-Detection-with%20MIDAS-MIDAS-R-and-MIDAS-F/

Moving From Statistical Rules to ML-based Solutions

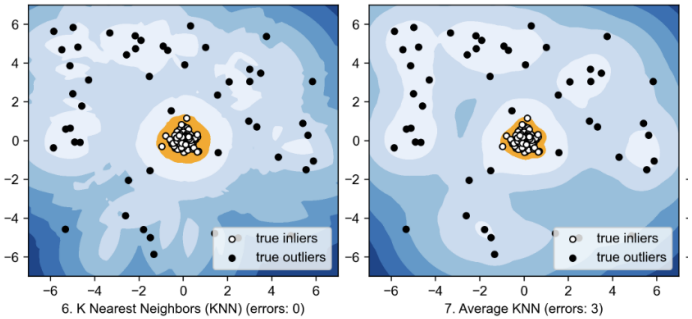
Old **statistical** approaches may try to identify the anomalies by the 3-sigma rule or some fixed rules.

- Better interpretability



The latest **ML-based approaches** are more flexible and can have complex (soft) decision boundaries:

- Work better for high-dimensional, complex datasets
- Limited interpretability
- But more pitfalls…such as noise



Outlier Detection Applications

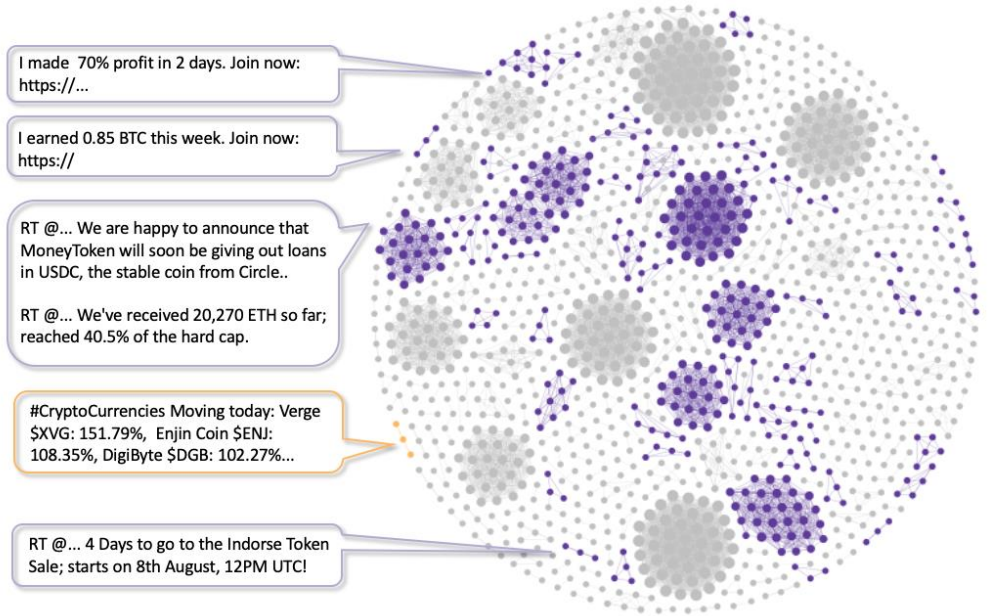
Outlier detection has many key applications with high im

- **Security & Risk Modeling:**
  - Real-time detection for autonomous driving
  - *Email privacy breach detection
- **Finance: Fraud detection** [5,6,7,8]
  - *AutoAudit: anti-money laundering
- **Healthcare and bioinformatics**:
  - *Rare disease detection
  - Identifying chronic brain infarcts on MRI
  - Detecting interesting gene expression

As long as rare items show interest to you, OD can be helpful!

Graph-based Fraud Detection

Bot Account on Twitter



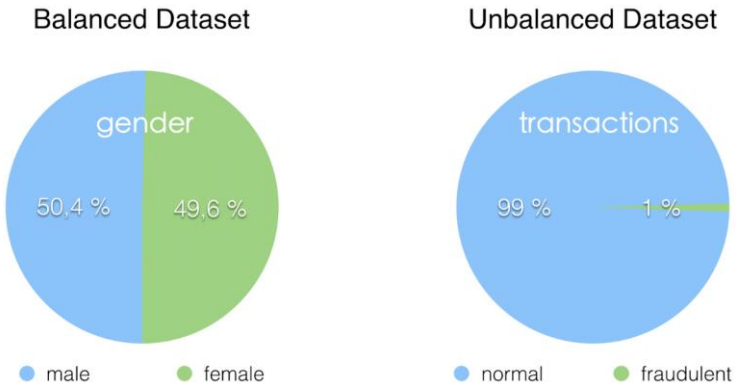
Coordinated Accounts on Social Network

[1] Pacheco, Diogo, et al. "Uncovering Coordinated Networks on Social Media: Methods and Case Studies." *ICWSM*. 2021.

Characteristics of Outlier Detection

Characteristics and challenges of outlier detection:

1. Outlier detection often faces **data imbalance**
   - The percentage of outliers << the percentage of normal samples
   - There are often more than one type of outliers in the data
2. **Data quality can often be a huge problem in OD, with noise and corruption**



Characteristics of Outlier Detection

Characteristics & challenges of outlier detection:

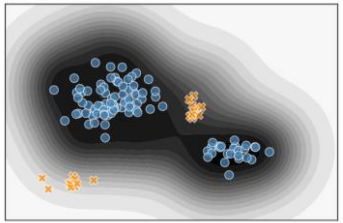
1. Outliers are conducted in an **unsupervised** fashion, no ground truth labels for evaluation
2. Outlier detection often faces **data imbalance**
3. It is often challenging to provide **interpretability** of outlier detection results
4. Outlier detection algorithms are often **costly**—scalable algorithms are needed
5. Data quality can often be a huge problem in OD, with noise and corruption

These challenges tell us that we should consider outlier detection problems by :

(1) availability of labels (2) computational cost and (3) interpretability (4) data quality

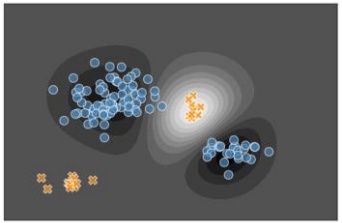
Un-, Semi-, and Fully Supervised OD Algorithms

***Unsupervised AD methods* are proposed with different assumptions of data distribution.** Its performance is limited by the correctness of the assumptions.



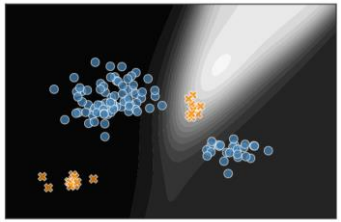
(b) Unsupervised AD (OC-SVM)

With the accessibility of full ground truth labels (which is rare), *supervised classifiers* may identify known anomalies at the risk of missing unknown anomalies. People often use XGBoost, LightGBM, and recent neural networks for it.



(c) Supervised classifier (SVM)

***Semi-supervised AD algorithms* are designed to capitalize the supervision from partial labels, while keeping the ability of detecting unseen types of anomalies**.



(d) Semi-supervised classifier

Outlier Detection Algorithms

Category of OD algorithms:

- **Linear Model**: PCA
- **Proximity-based Model**: kNN, LOF, HBOS
- **Probabilistic Model**: ECOD
- **Ensemble Model**: Isolation Forest (iForest)
- **Neural Networks**: AutoEncoder (AE) and variational outlier encoder (VAE)

Due to the data imbalance, the evaluation metric of OD cannot be accuracy:

- ROC-AUC
- Precision @ Rank k: top k predictions' precision
- Average Precision: the area under the precision-recall curve

OD Algorithm 1: k Neatest Neighbors (kNN)

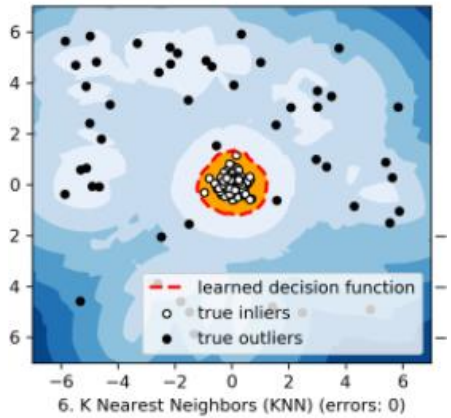
Idea: calculate the distance to the k-th nearest neighbor. The larger, the more anomalous.

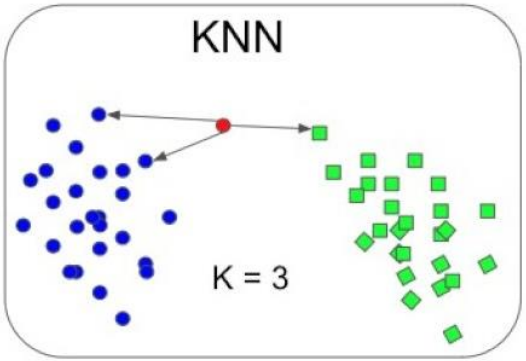
Usage: from pyod.models.knn import KNN

Advantages: easy to understand and implement

Disadvantages: high computational cost; not for high dimensional datasets

Decision boundary





OD Algorithm 2: Local Outlier Factor (LOF)

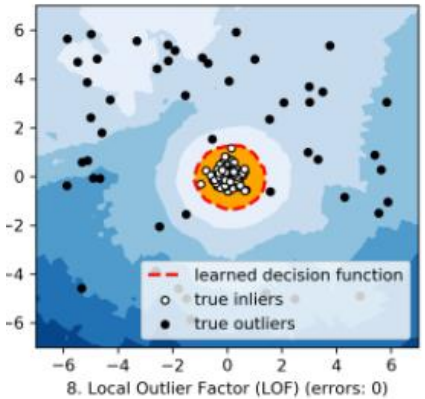
Idea: calculate sample density on the local region and the density of its neighbors

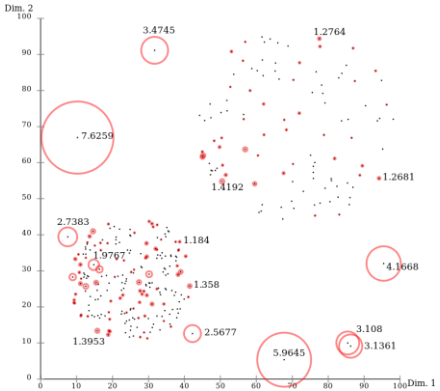
Usage: from pyod.models.lof import LOF

Advantages: easy to understand (slightly more complex than kNN)

Disadvantages: high computational cost; not for high dimensional datasets

Decision boundary





OD Algorithm 3: Histogram-based OD (HBOS)

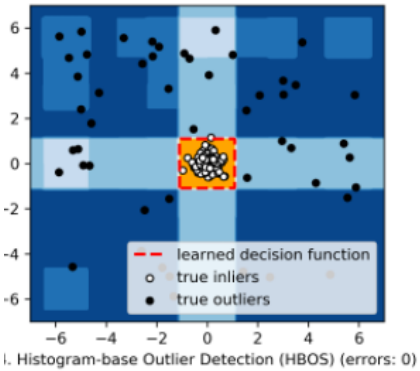
Idea: assume each feature is independent; estimate the histograms separately and combine

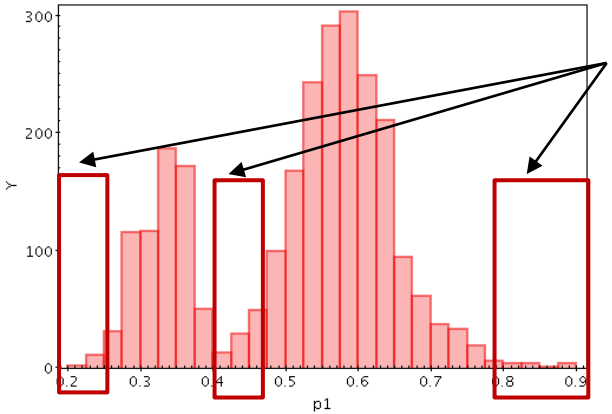
Usage: from pyod.models.hbos import HBOS

Advantages: simple to use; easy to be distributed; suited for large-scale problem

Disadvantages: cannot capture complex feature dependency, while it works well in general

Decision boundary





Low density region

OD Algorithm 4: Isolation Forests (iForest)

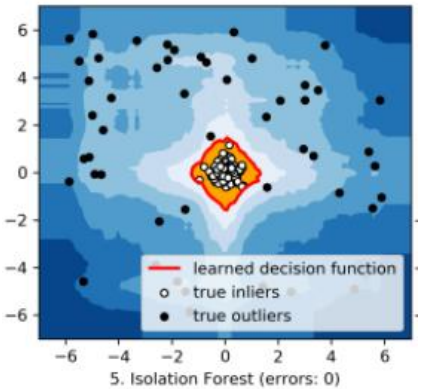
Idea: build multiple decision trees to split the feature space, and observe the difficulty of "isolating" a sample. An outlier is easier to be isolated with a small tree depth.

Usage: from pyod.models.iforest import IForest

Advantages: fast computation; easy to be distributed; and empirically robust

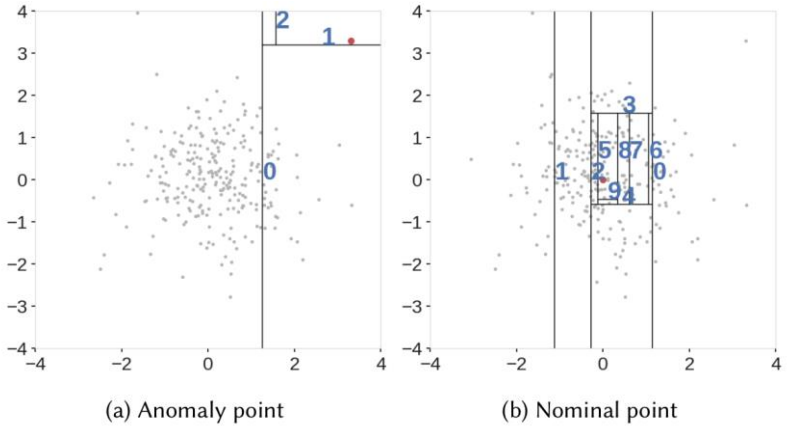
Disadvantages: many hypereparameters to tune with some built-in randomness

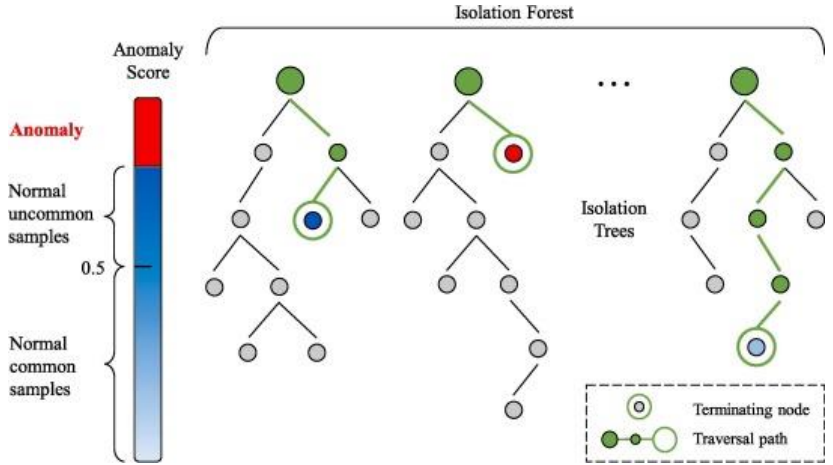
Decision boundary



OD Algorithm 4: Isolation Forests (iForest)

The decision process is to iteratively split the feature space. Outliers are closer to the root.





OD Algorithm 5: Autoencoder (AE) and Variational AE

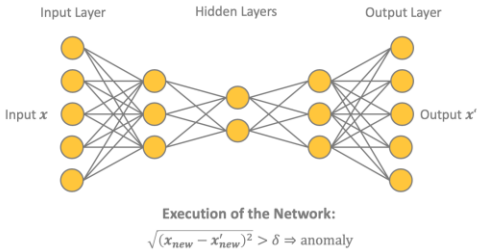
Idea: similar to PCA, AE uses non-linear modeling. The idea is to the data through the encoder and the reconstruct with the decoder), outliers have larger reconstruction error, $||\mathrm{X} - \mathrm{X}'||$)

Usage: from pyod.models.auto_encoder import AutoEncoder

Usage: from pyod.models.vae import VAE

Advantages: easy to understand; neural networks are suited for high-dimensional datasets

Disadvantages: neural networks may face the issue of convergence



OD Summary

1. Most of the algorithms are unsupervised
2. Detection depends on the data similarity – measured by distance and/or density
3. Data faces extreme imbalance – need to choose the evaluation carefully

Important Anomaly Detection Interview Questions

Question: What challenges are faced in anomaly detection?

Important Anomaly Detection Interview Questions

Question: What challenges are faced in anomaly detection?

Answer:

- **High False Positive Rate**: Distinguishing between a true anomaly and a legitimate data point that happens to be unusual.
- **Dynamic Data**: In many real-world scenarios, the data is constantly evolving, making it difficult to establish a static model of normal behavior.
- **Imbalanced Data**: Typically, in anomaly detection, the number of normal instances significantly outweighs the number of anomalies.
- **Adversarial setting**: patterns change frequently

Important Anomaly Detection Interview Questions

Question: Explain the difference between supervised and unsupervised anomaly detection.

Important Anomaly Detection Interview Questions

Question: Explain the difference between supervised and unsupervised anomaly detection.

Answer: In supervised anomaly detection, the algorithm is trained on a labeled dataset that contains both normal and anomalous samples. It's essentially a classification problem. However, in **unsupervised anomaly detection**, the algorithm is not trained on labeled data, and it tries to identify anomalies based on the inherent characteristics of the data, often by building a profile of what's normal and flagging data points that deviate significantly from this profile.

Important Anomaly Detection Interview Questions

Question: How do you evaluate the performance of an anomaly detection system?

Important Anomaly Detection Interview Questions

Question: How do you evaluate the performance of an anomaly detection system?

Answer: human verification (human-in-the-loop, sampling, etc.)

The performance of anomaly detection systems is typically evaluated using metrics such as precision, recall, F1-score, and the area under the ROC curve (AUC-ROC). Due to the imbalanced nature of anomaly detection problems (where anomalies are rare), precision-recall curves are often more informative than ROC curves.

The Next Few Lectures

***TAs or myself will cover some basic Python/PyTorch and DL libraries**

We will move to the neural networks. But these basics are important:

- You know the linear models as the components of DL
- You know how to frame an ML problem
- You know loss and optimization – and know overfitting and underfitting
- You know multiple classical ML topics – they can come in handy
- You know why non-linear models are important

Let us move to the neural networks – non-linear, powerful ML models