MTH 221

Fundamentals of Machine Learning

Batuhan Bardak

Lecture 2: Basic concepts of ML & ML by example (k-NN)

Date: 10.10.2023

Plan for today

- Recap on ML Basics
- KNN
- Curse of Dimensionality
- Hands on examples with Python (Demo of KNN & Curse of Dimensionality)
- Feature Scaling & Evaluation Metrics

Grading

- Midterm: %20 (closed book, no notes, no cheat sheet)
- Final: %30 (closed book, no notes, no cheat sheet)
- Homeworks: %20 (with Python)
- Course Project: %30 (done in groups of 2)

Note: If you submit your project final reports to the IEEE conference, 20 points will be added to your project grade.

Note2: Extra %5 points for active participation in the class.

$\vec{x}_1 = [x_1, x_2 \dots x_d]$

Recap on ML Basics

instance

labels

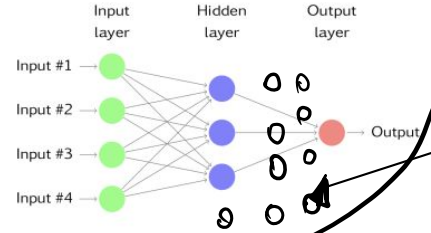
Dataset:

$\mathcal{D} = \{(x_1, y_1), \dots, (x_n, y_n)\}, x_i \in \mathbb{R}^d, y_i \in \mathcal{C}$ (e.g., $\mathcal{C} = \{-1, 1\}$), $(x_i, y_i) \sim \mathcal{P}$

$\{-1, 1\}$ or $\{0, 1\}$ → Binary classification.

$\{0, 1, 2, 3, 4\}$ → multi-class classification.

235K$ → Regression problem.



Hypothesis:

$h : \mathbb{R}^d \mapsto \mathcal{C}$

i.e., a neural network-based classifier that maps image to label of cat or dog

Hypothesis class

$\mathcal{H} = \{h\}$

i.e., a large family of NNs with different parameters

$f(x) = y$

best h inside of the H.

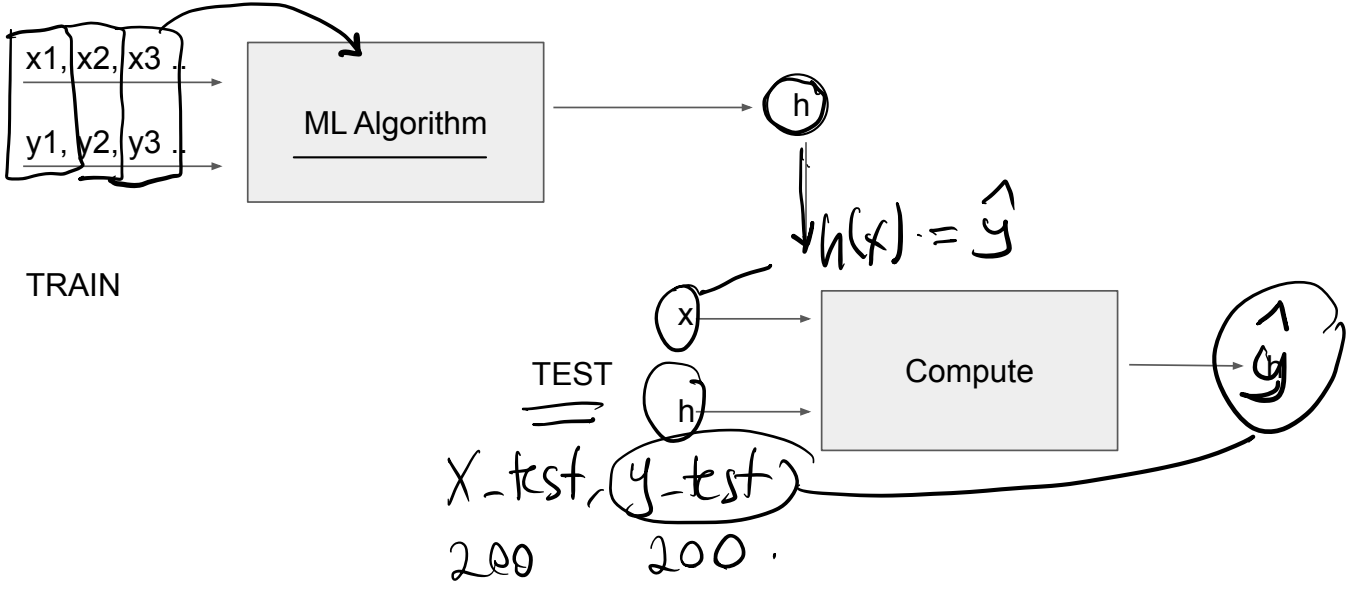
min → loss function.

800 800

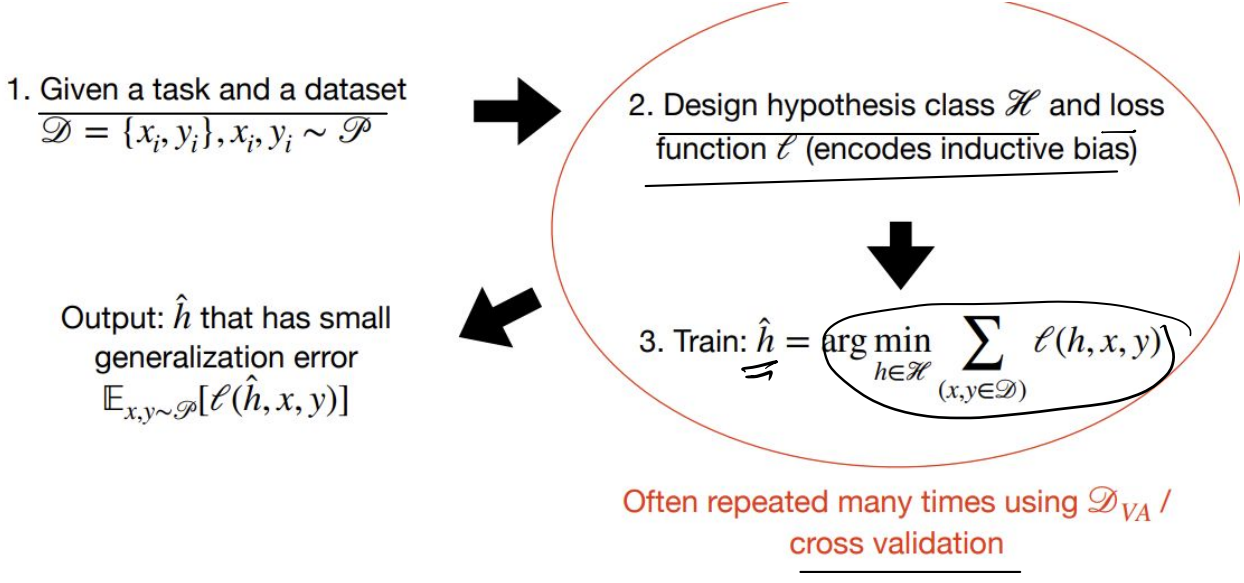
X_train, y_train

Training

X, Y = 1,000.



Summary

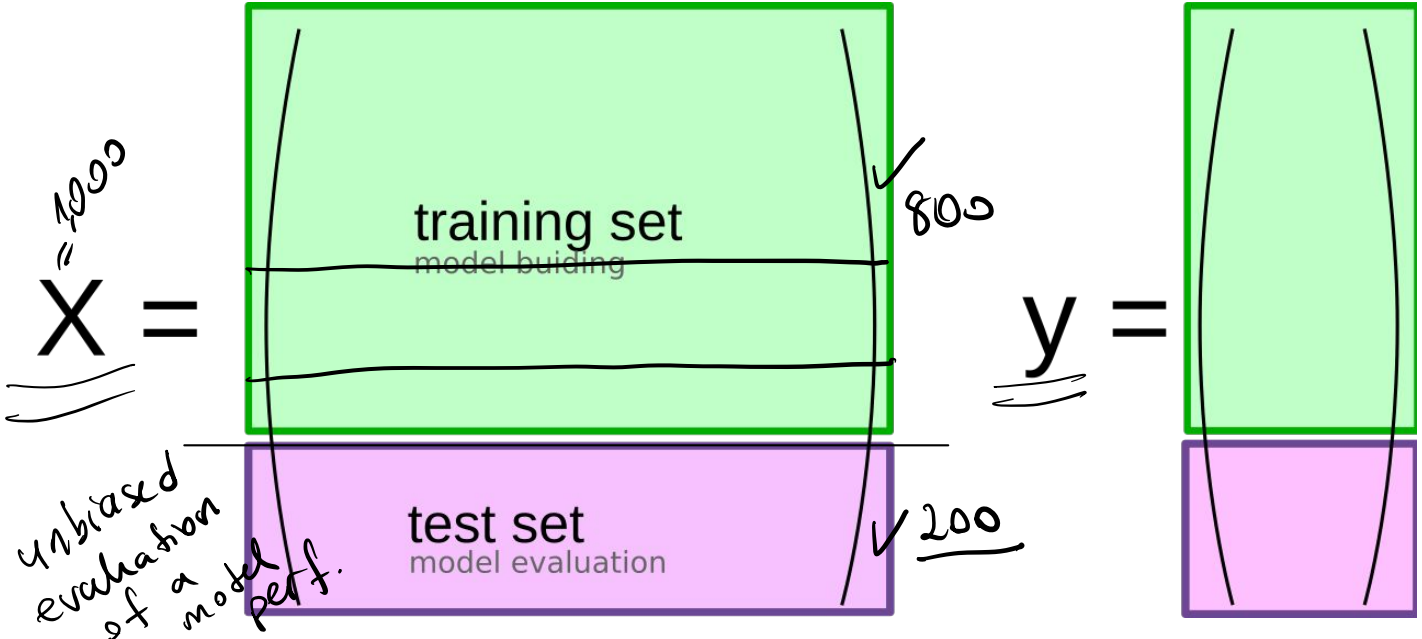


Slide Credit: Cornell CS4780

limited data

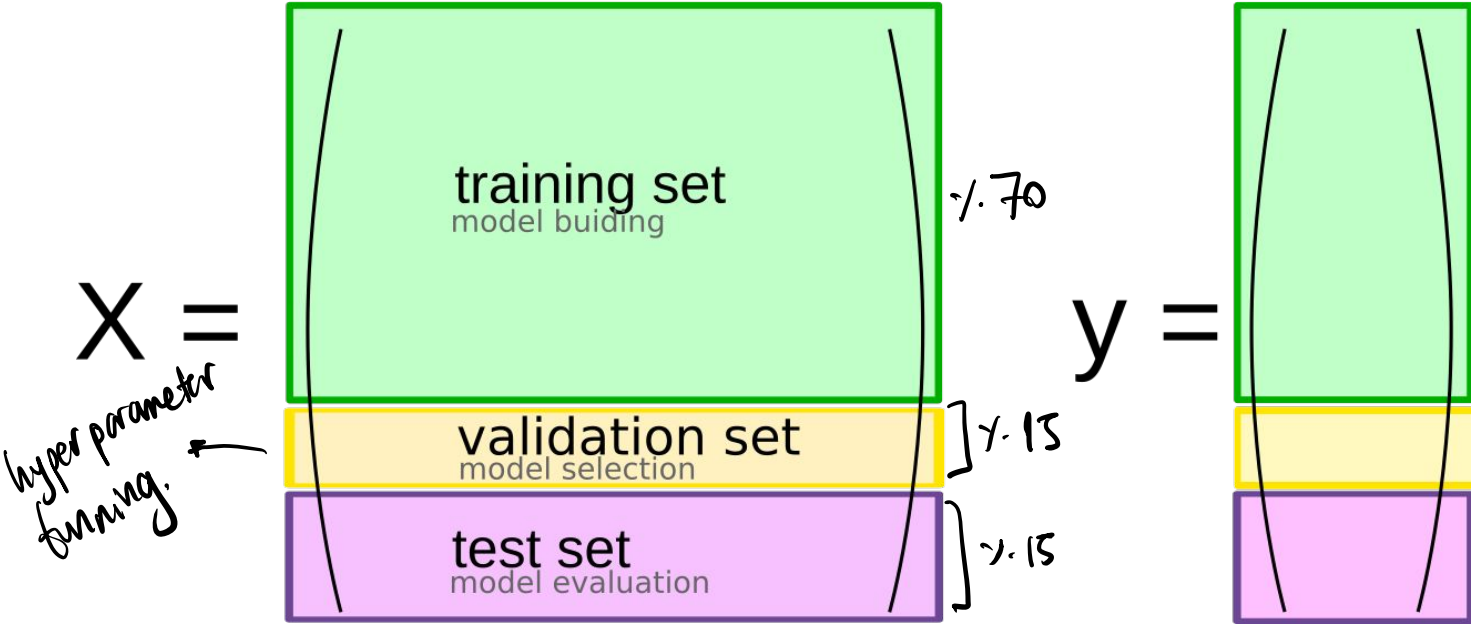
1,000

Train / Test Split



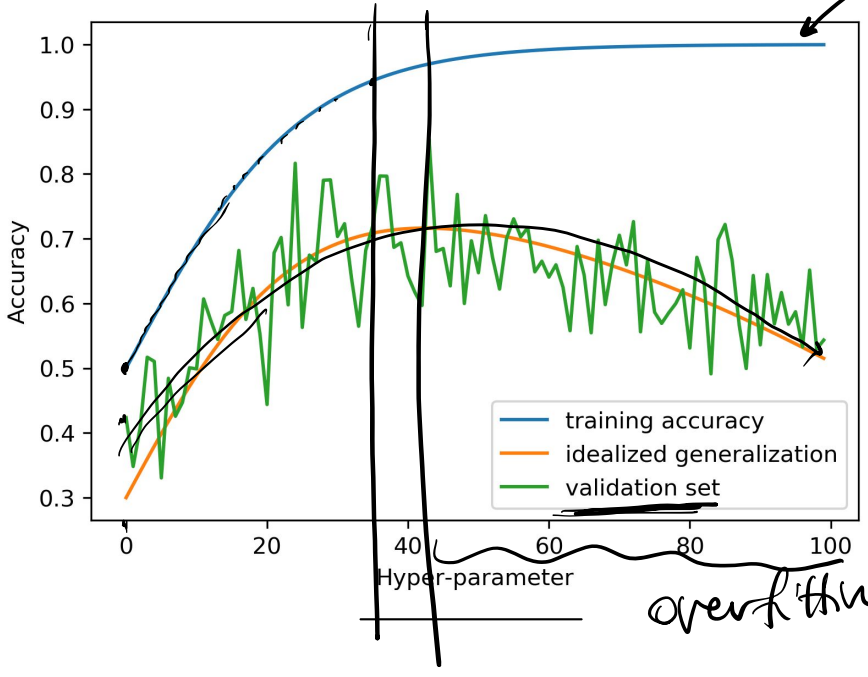
Slide Credit: Amuller, COMS4995

Train / Test Split

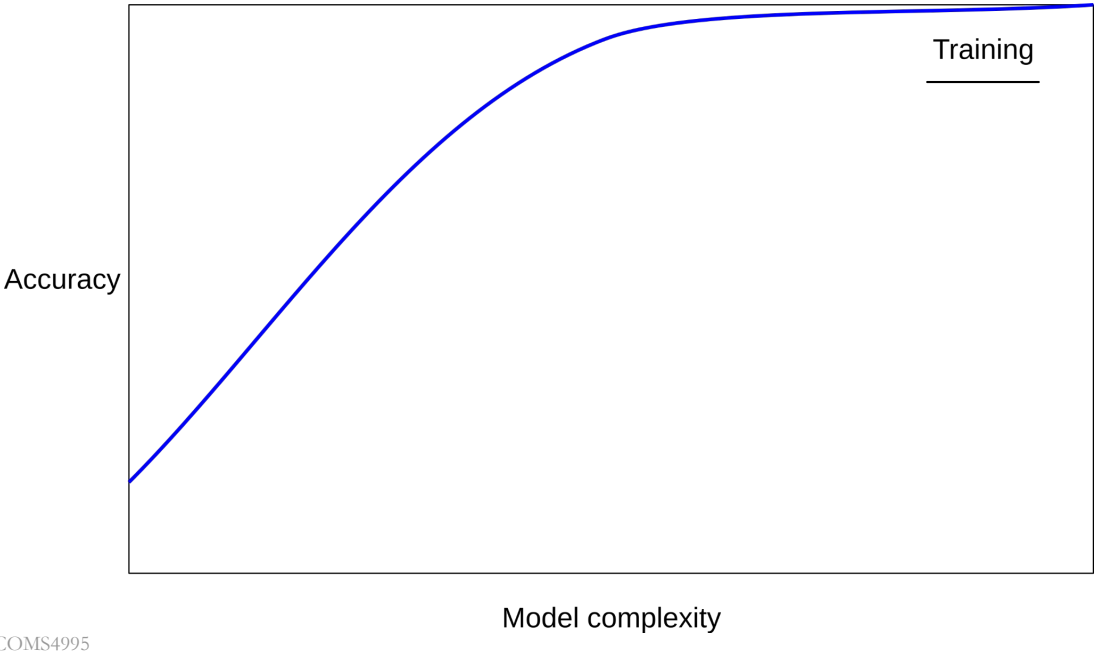


Slide Credit: Amuller, COMS4995

Train / Test Split

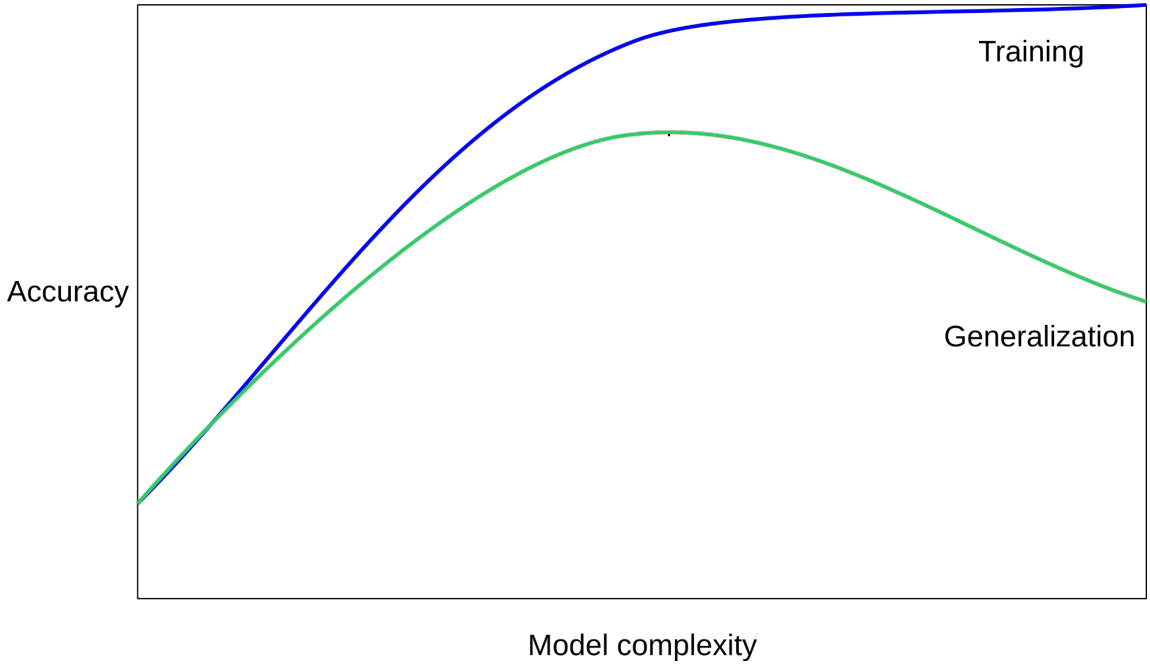


Overfitting / Underfitting



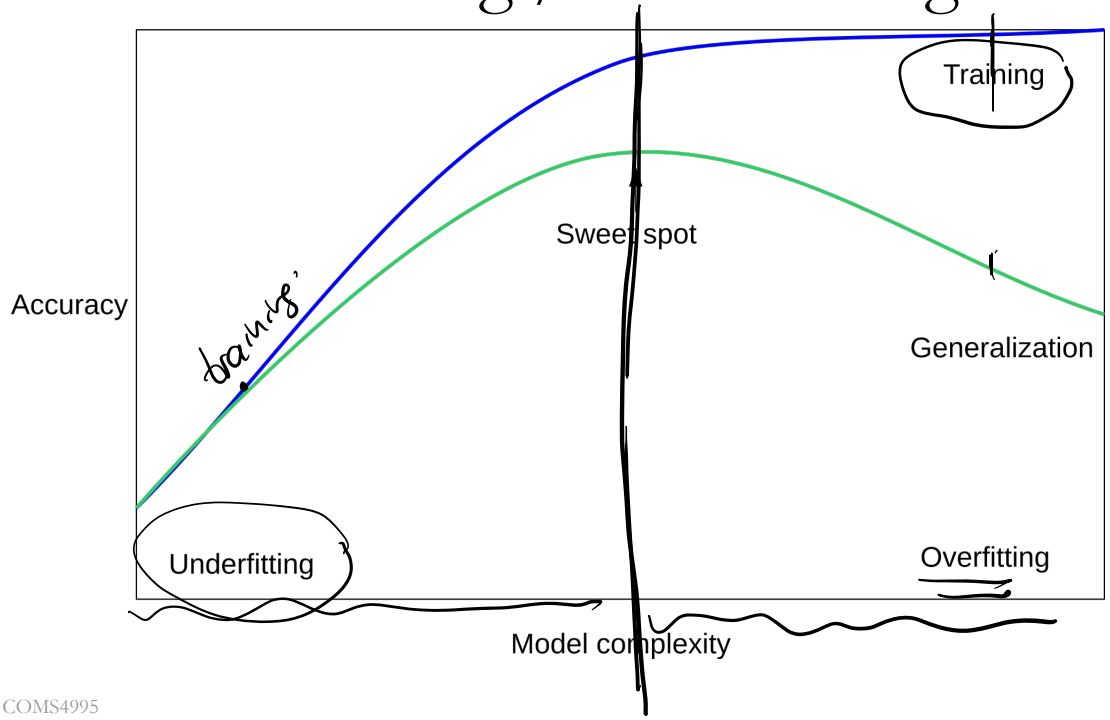
Slide Credit: Amuller, COMS4995

Overfitting / Underfitting

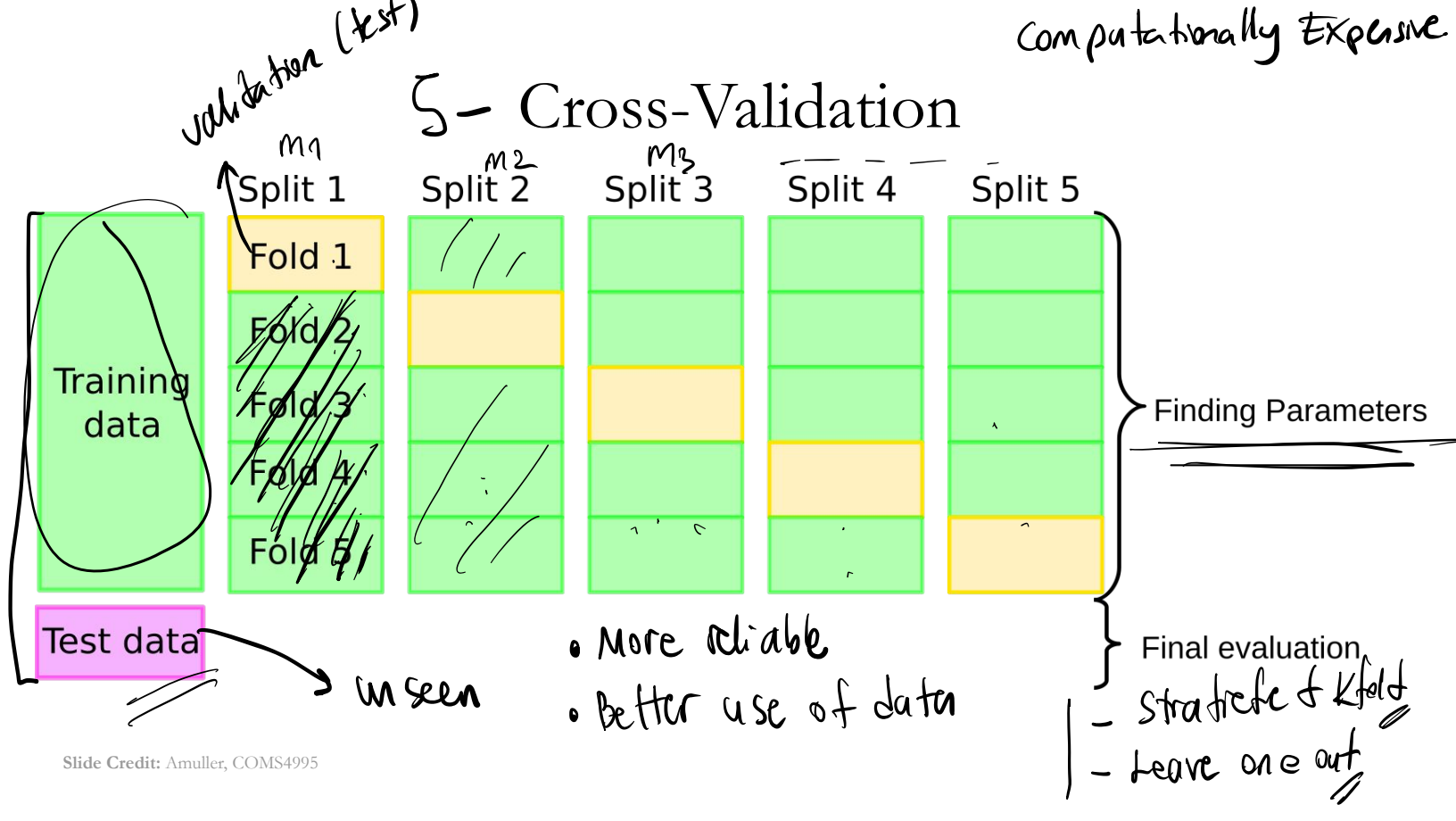


Slide Credit: Amuller, COMS4995

Overfitting / Underfitting

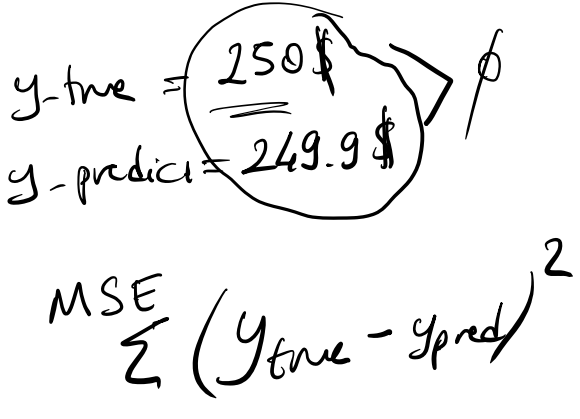


Slide Credit: Amuller, COMS4995


validation (test)
Computationally Expensive
5- Cross-Validation
m1
m2
m3
Split 1
Split 2
Split 3
Split 4
Split 5
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Training data
Finding Parameters
Test data
unseen
More reliable
Better use of data
Final evaluation
- Stratified & Kfold
- Leave one out
Slide Credit: Amuller, COMS4995

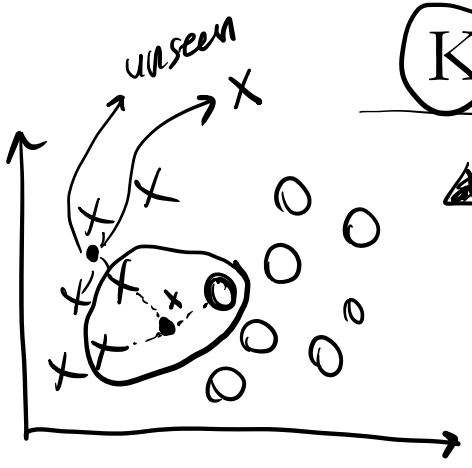
Recap on ML Basics

- T / F: A hypothesis that achieves zero training error is always good
- T / F: zero-one loss is a good loss function for regression
- T / F: We can use validation dataset to check if our model overfits



K Nearest Neighbors

K = 3



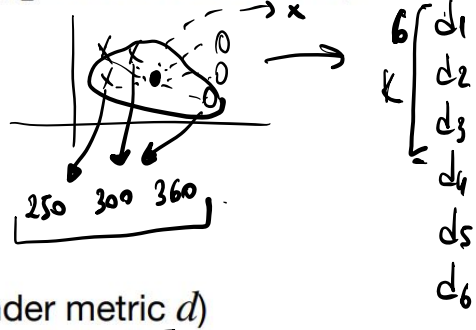
- No Free Lunch
- No one-size-fits-all algorithm.

- KNN assumption: the data points that are similar, have similar labels.

2 x
1 . } → x

The K-NN Algorithms

Input: classification training dataset $\{x_i, y_i\}_{i=1}^n$, and parameter $K \in \mathbb{N}^+$, and a distance metric $d(x, x')$ (e.g., $\|x - x'\|_2$ euclidean distance)



K-NN Algorithm:

Store all training data

For any test point x :

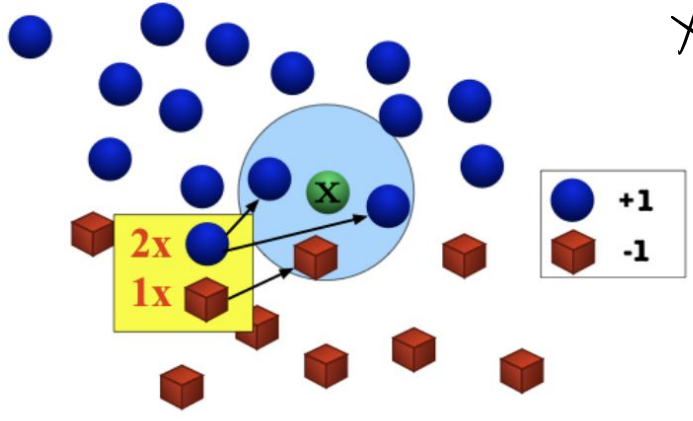
> Find its top K nearest neighbors (under metric d)
>
> Return the most common label among these K neighbors
>
> (If for regression, return the average value of the K neighbors)

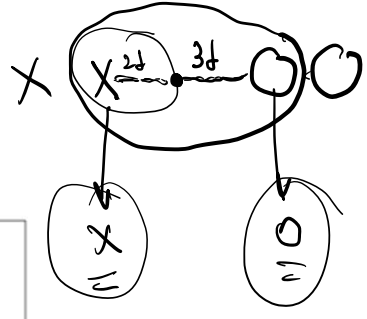
Slide Credit: Cornell CS4780

The K-NN Algorithms

K=2

Example: 3-NN for binary classification using Euclidean distance





Slide Credit: Cornell CS4780

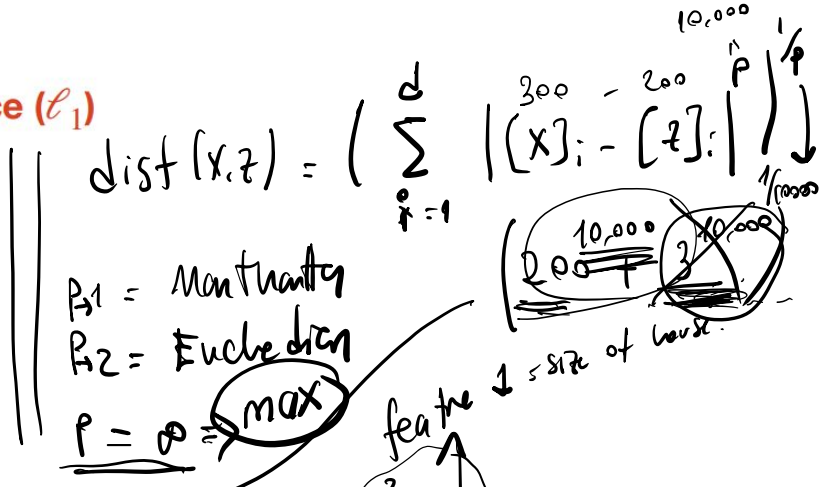
The K-NN Algorithms

1. We believe our metric d captures similarities between examples:

Examples that are close to each other share similar labels

Another example: Manhattan distance (ℓ_1)

$$d(x, x') = \sum_{j=1}^{d} |x[j] - x'[j]|$$



Slide Credit: Cornell CS4780

Distance Metrics

- Euclidean Distance

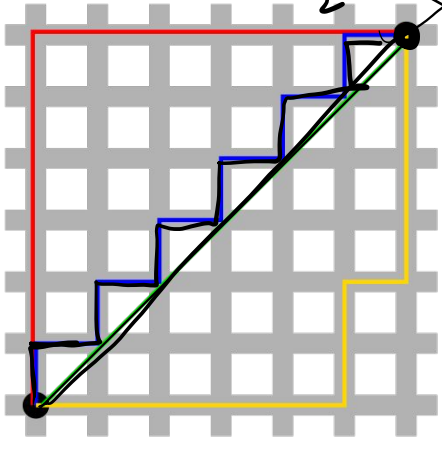
$$d(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}.$$

- Manhattan Distance

$$d_1(\mathbf{p},\mathbf{q}) = \|\mathbf{p}-\mathbf{q}\|_1 = \sum_{i=1}^{n} |p_i - q_i|,$$

- Minkowski Distance

$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}}.$$

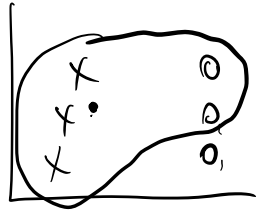


scikit-learn

p : *int, default=2*

Power parameter for the Minkowski metric. When p = 1, this is equivalent to using manhattan_distance (l1), and euclidean_distance (l2) for p = 2. For arbitrary p, minkowski_distance (l_p) is used.

The choice of K



1. What if we set K very large?

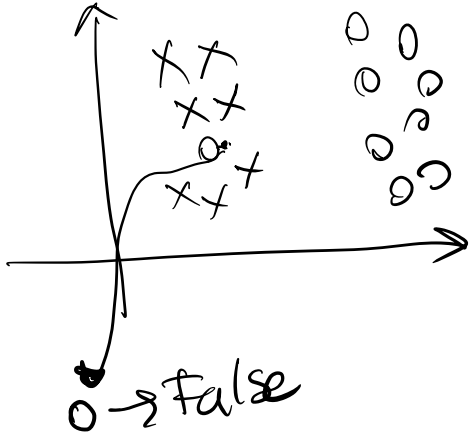
Top K-neighbors will include examples that are very far away…

K = 5

2. What if we set K very small (K=1)?

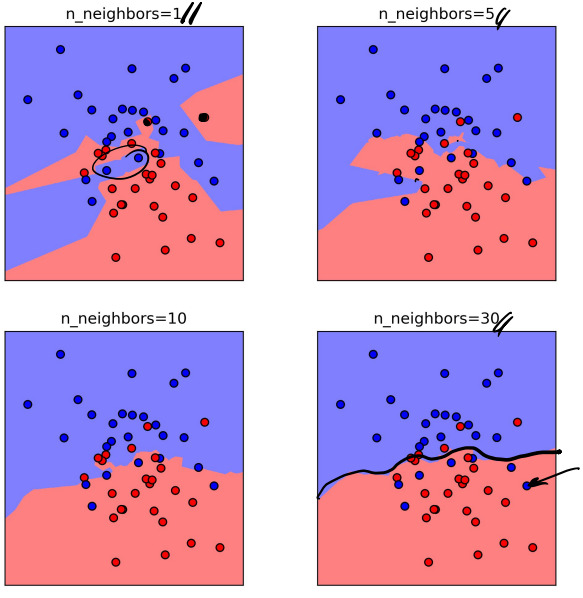
label has noise (easily overfit to the noise)

(What about the training error when K = 1?)



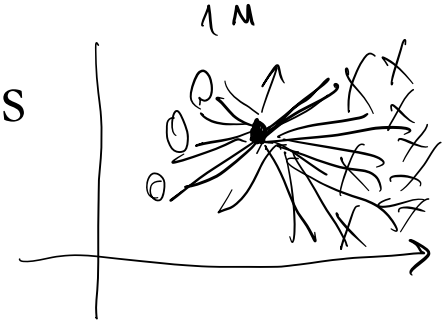
Slide Credit: Cornell CS4780

k-Nearest Neighbors



Slide Credit: Amuller, COMS4995

k-Nearest Neighbors

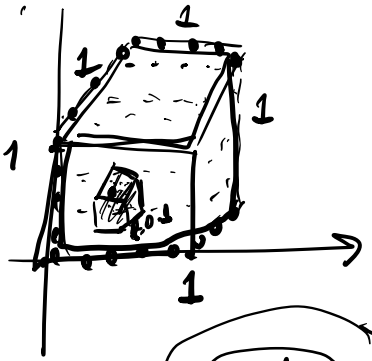


- Main advantage
  - Simple to understand and implement
  - Immediately adapts as we collect new training data
- Downside
  - The computational complexity for classifying new examples grows linearly with the number of examples in the training dataset in the worst-case scenario
  - We can't discard training examples since no training step is involved. Thus, storage space can become a challenge if we working with large datasets.
  - The algorithm's performance might degrade with high-dimensionality data (aka **Curse of Dimensionality**!)
  - It's sensitive to irrelevant or redundant features since all features contribute equally to the distance calculation (you may use weight to solve this problem!)

Slide Credit: Sebastian Raschka, Python Machine Learning

Code Example

- 1-Introduction-to-Python.ipynb
- 2-KNN-Implementation.ipynb
- 3-Find Best K Value for KNN.ipynb
- Check this link for hyperparameters of kNN.



$d = 1$

$l^d \approx \frac{k}{N} \rightarrow l \approx \left(\frac{k}{N}\right)^{1/d} \rightarrow \left(\frac{10}{1000}\right)^{1/2}$

$\rightarrow (0.01)^{1/2} \Rightarrow l \approx 0.1$

$l^d \approx \frac{k}{N}$

$k = 10$

$N = 1000$

$d = 10 \rightarrow l = 0.63$

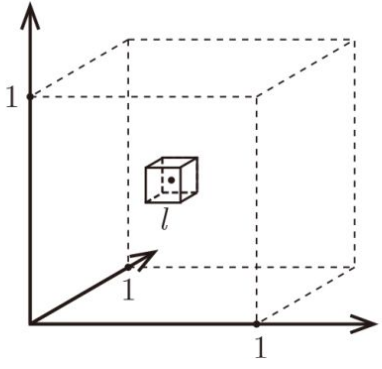
$d = 100 \rightarrow l = 0.95$

$d = 1000 \rightarrow l = 0.99$

Curse of Dimensionality

Key problem: in high dimensional space, points that are draw from a distribution tends to be far away from each other!

Example: let us consider uniform distribution over a cube $[0,1]^d$

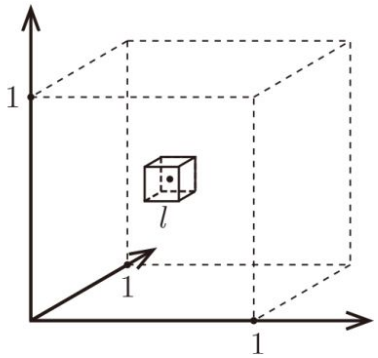


Q: sample x uniformly, what is the probability that x is inside the small cube?

A: Volume(small cube)/volume($[0,1]^d$) $= l^d$

Curse of Dimensionality

Example: let us consider uniform distribution over a cube $[0,1]^d$



Now assume we sampled n points uniform randomly, and we observed K points fall inside the small cube

So empirically, the probability of sampling a point inside the small cube is roughly K/n

Thus, we have $l^d \approx \frac{K}{n}$

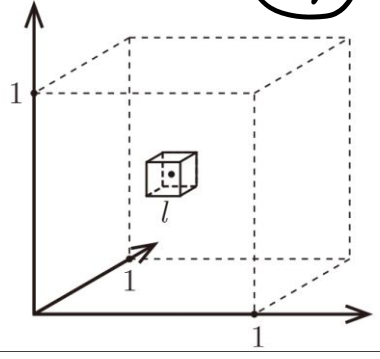
Slide Credit: Cornell CS4780

$d \rightarrow d > 100$

Curse of Dimensionality

Example: let us consider uniform distribution over a cube $[0,1]^d$

We have $l^d \approx \frac{K}{n}$



Q: how large we should set l, s.t., we will have K examples (out of n) fall inside the small cube?

$$l \approx (K/n)^{1/d} \rightarrow 1, \text{ as } d \rightarrow \infty$$

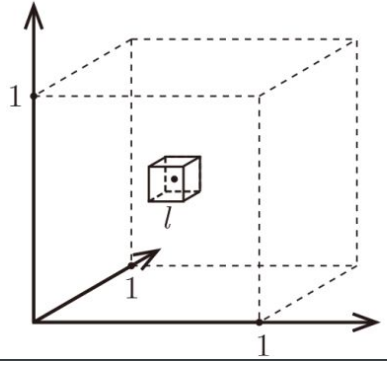
Bad news: when $d \rightarrow \infty$, the K nearest neighbors will be all over the place!
(Cannot trust them, as they are not nearby points anymore!)

Slide Credit: Cornell CS4780

Can we just increase "n" to avoid this?

Example: let us consider uniform distribution over a cube $[0,1]^d$

We have $l^d \approx \frac{K}{n}$



Q: to make sure that we have one sample inside a small cube, how large n needs to be?

Set $\ell = 0.1$, $K = 1$, then $n = 1/(0.1)^d = 10^d$

Bad news: when $d \geq 100$, # of samples needs to be larger than total # of atoms in the universe!

Slide Credit: Cornell CS4780

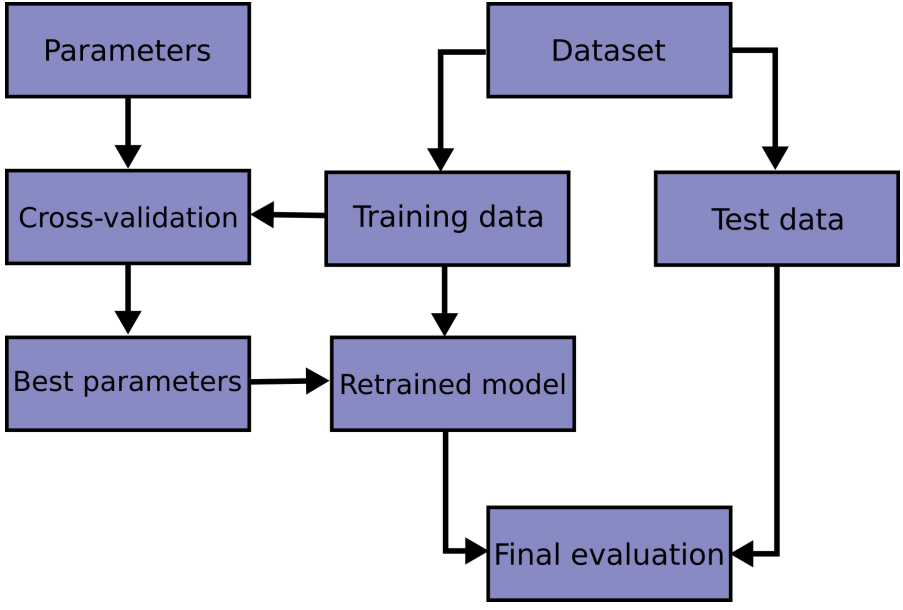
The curse of dimensionality

- As dimensions increase ..
  - Euclidean distances become less meaningful
  - Uniform distributions become exponentially harder to sample
  - Data becomes more difficult to visualize

Code Example

- 4-KNN & Curse_of_Dimensionality.ipynb

Pipeline for hyper parameter tuning

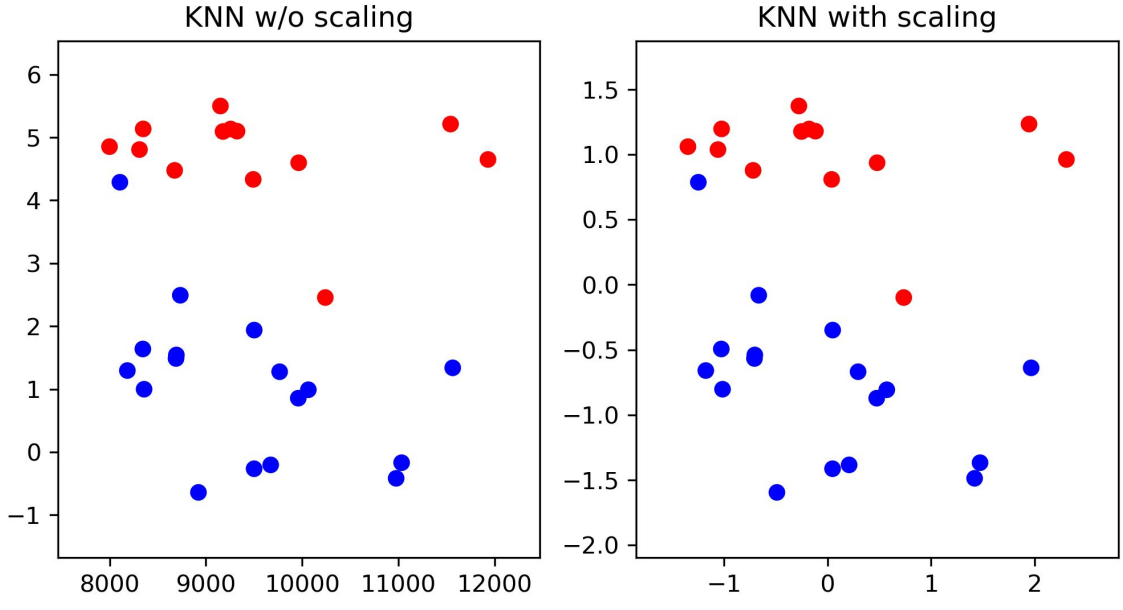


Slide Credit: Amuller, COMS4995

Further Reading on model evaluation & selection

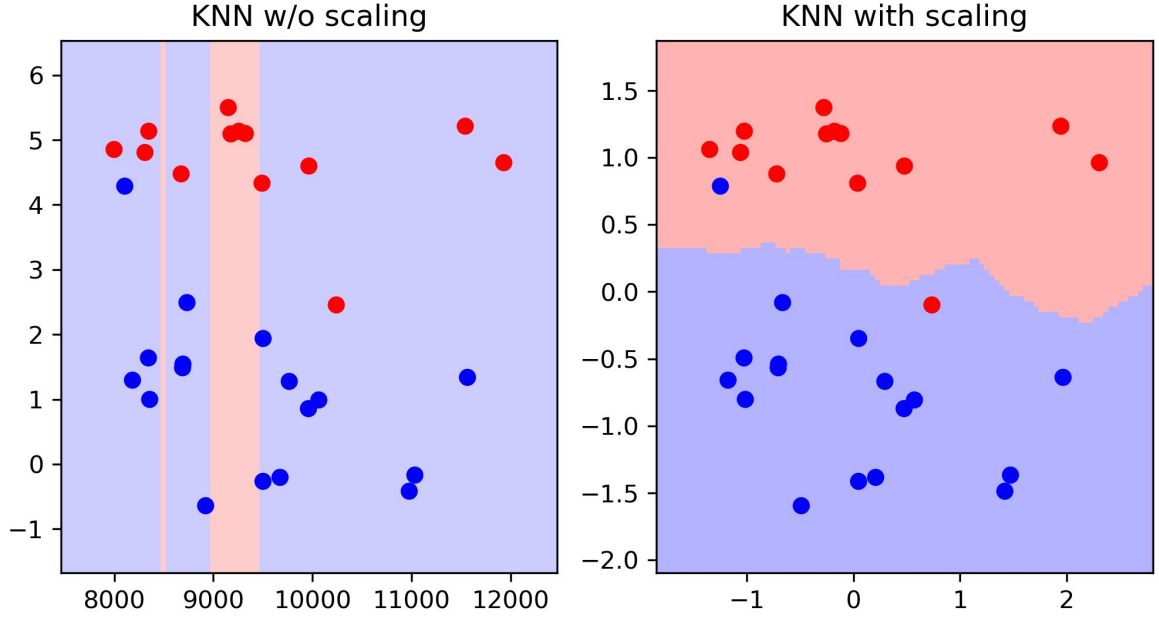
- Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning
  - https://arxiv.org/pdf/1811.12808.pdf
- Cross-validation failure: small sample sizes lead to large error bars
  - https://hal.inria.fr/hal-01545002/file/paper.pdf

Feature Scaling

- 

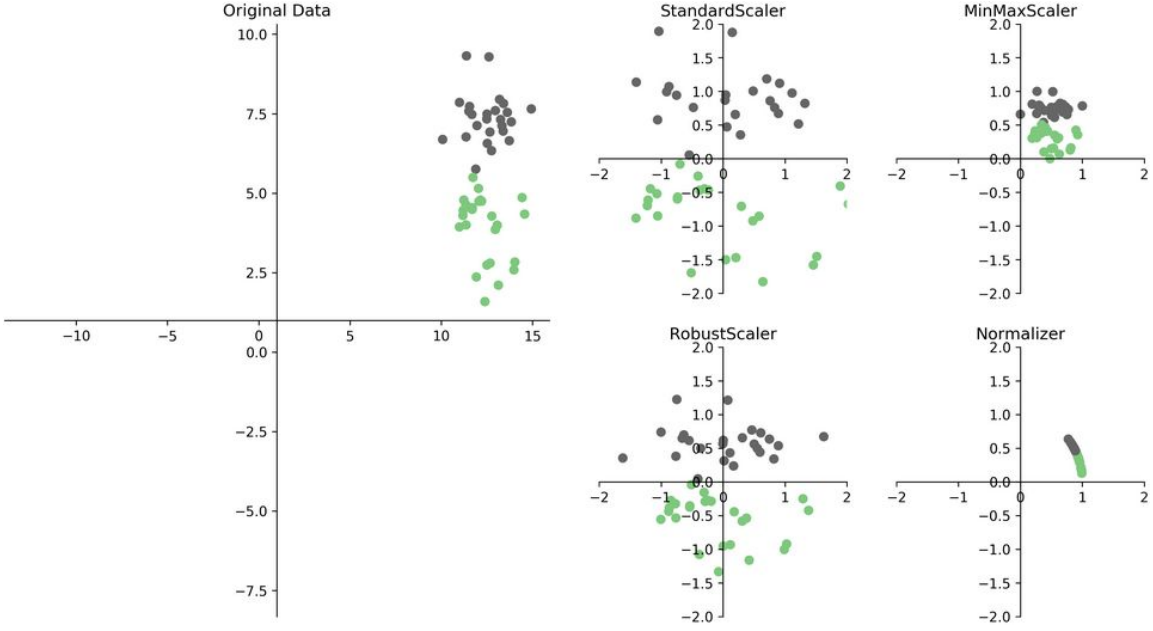
Slide Credit: Amuller, COMS4995

Feature Scaling

- 

Slide Credit: Amuller, COMS4995

Different Methods for Scaling Data



Slide Credit: Amuller, COMS4995

Feature Scaling (Normalization)

- MinMax Scaler
  - Transformers features between 0 and 1.

$$\frac{x_i - \min(x)}{\max(x) - \min(x)}$$

- Standard Scaler
  - Assumes a normal distribution for data within each feature.
  - Scaling makes the distribution centered around 0, with standard deviation of 1 and the mean removed.

$$\frac{x_i - \text{mean}(x)}{sd(x)}$$

- Robust Scaler
  - Scale features that are robust to outliers. Similar to MinMax Scaler but it uses the interquartile range (rather than the min-max). The median and scales of the data are removed by this scaling algorithm according to the quantile range.

$$\frac{x_i - Q_1(x)}{Q_3(x) - Q_1(x)}$$

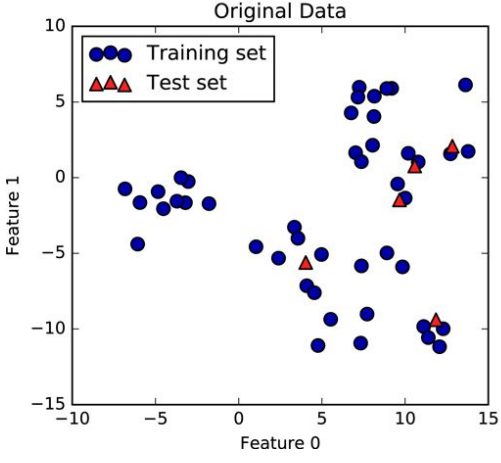
- For comparing methods
  - https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html

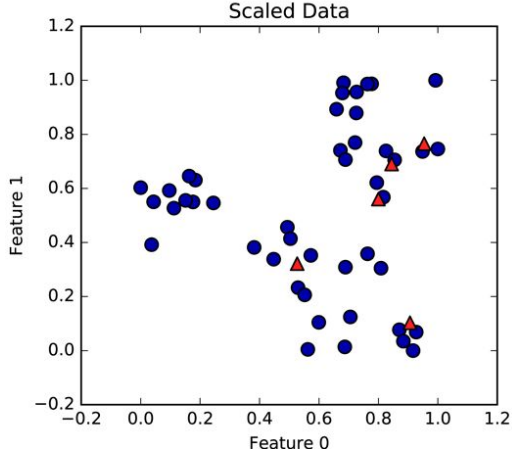
Slide Credit: machinelearninggeek.com/feature-scaling-minmax-standard-and-robust-scaler

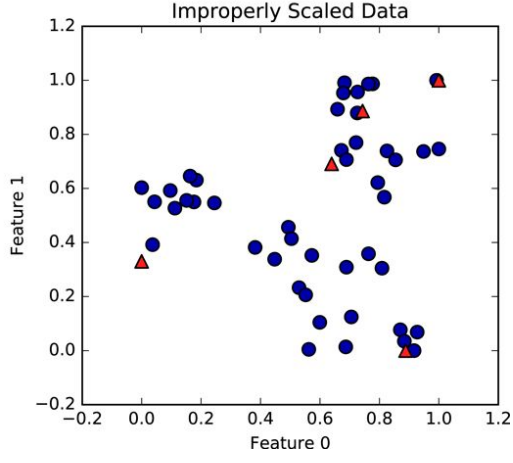
Improperly Scaled Data

X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.fit_transform(X_test)





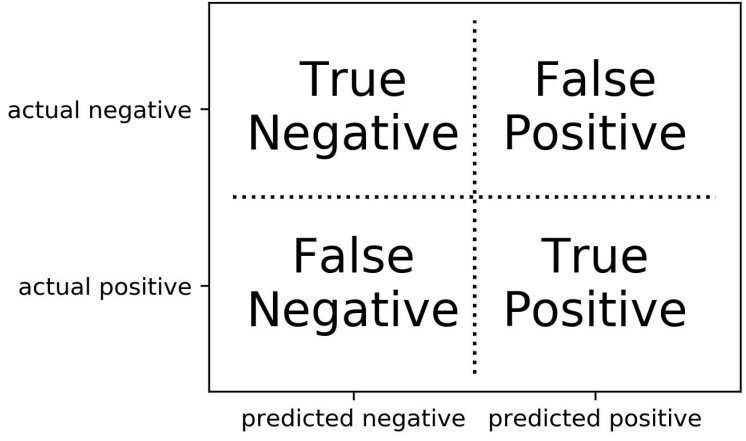


Slide Credit: Amuller, COMS4995

Code Example

- 5-Feature Scaling.ipynb

Evaluation Metrics for Binary Classification



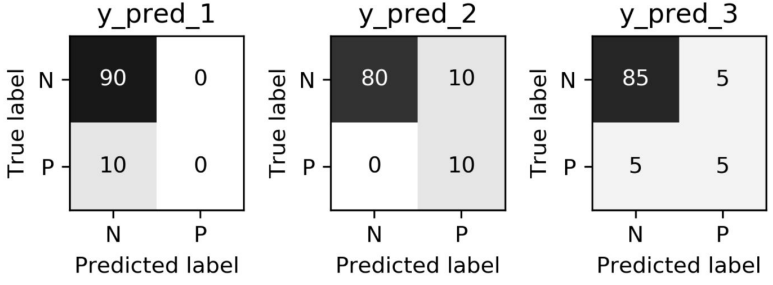
$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Problems with Accuracy

Data with 90% negatives:

```
from sklearn.metrics import accuracy_score
for y_pred in [y_pred_1, y_pred_2, y_pred_3]:
    print(accuracy_score(y_true, y_pred))
```

```
0.9
0.9
0.9
```



Slide Credit: Amuller, COMS4995

Evaluation Metrics for Classification

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Positive Predicted Value (PPV)

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Sensitivity, coverage, true positive rate.

$$\text{F} = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Harmonic mean of precision and recall

Slide Credit: Amuller, COMS4995

Code Example

- 6-Evaluation Metrics.ipynb

Administrative

- You can do the projects in teams of 2 people.
- Please email the team member(s) till 13.10.2023 and send your project proposal till 20.10.2023.
  - To: cinarb@metu.edu.tr
  - Cc: bbatuhan@metu.edu.tr
- The project proposal should be max one page (min. 300 words) .Your project proposal should describe:
  - What is the problem that you will be investigating? Why is it interesting?
  - What reading will you examine to provide context and background?
  - What data will you use? If you are collecting new data, how will you do it?
  - What method or algorithm are you proposing? If there are existing implementations, will you use them and how? How do you plan to improve or modify such implementations? You don't have to have an exact answer at this point, but you should have a general sense of how you will approach the problem you are working on.
  - How will you evaluate your results? Qualitatively, what kind of results do you expect (e.g. plots or figures)? Quantitatively, what kind of analysis will you use to evaluate and/or compare your results?
- Kaggle (active competition) | Algorithmic Trading | Other

Please submit your proposal as a PDF. **Only one person on your team should** submit. Please have this person add the rest of your team as collaborators as a "Group Submission".

Slide Credit: Stanford CS231n, project proposal , http://cs231n.stanford.edu/project.html

Administrative

- Expect some questions from the paper below in the midterm exam. (Reading assignment)
  - A Few Useful Things to Know about Machine Learning, *P. Domingos*
    - *https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf*

Next Class:

Perceptron