Machine Learning (CE 40477)

Fall 2024

Ali Sharifi-Zarchi

CE Department
Sharif University of Technology

October 6, 2024

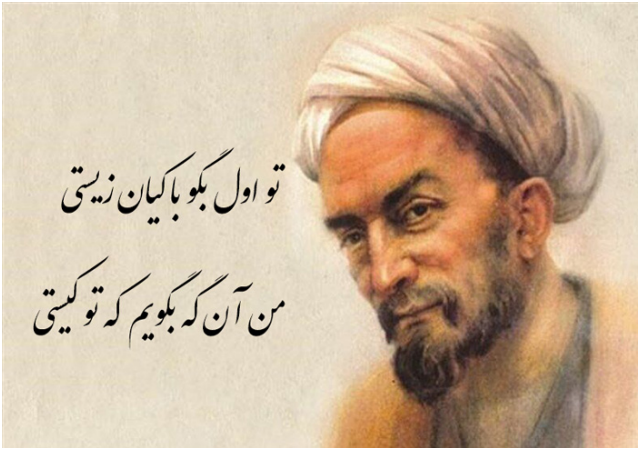
Parametric vs. non-parametric methods

- Parametric methods need to find parameters from data and then use the inferred parameters to decide on new data points
  - Learning: finding parameters from data
  - e.g., Linear regression, Logistic regression
- Non-parametric methods
  - Training examples are explicitly used
  - Training phase is not required
  - e.g., k-Nearest neighbors (kNN)
- Both supervised and unsupervised learning can be categorized into parametric and non-parametric methods

Non-parametric learners

- **Memory-based** or **Instance-based** learners
  - lazy learning: (almost) all the work at the test time
- Generic description:
  - Memorize training $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$
  - Given test x predict: $\hat{y} = f(x; x^{(1)}, y^{(1)}, \ldots, x^{(n)}, y^{(n)})$
- f is typically expressed in terms of the similarity of the test samples x to the training samples $x^{(1)}, \ldots, x^{(n)}$
- kNN is an instance-based learner

kNN concept

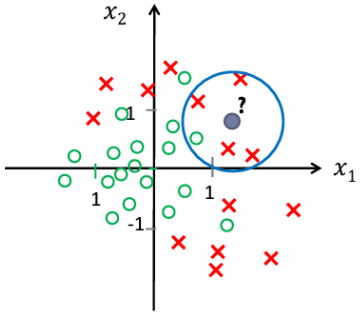


- First, tell me who you have lived with, Then, I will tell you who you are.

Figure adapted from https://setare.com

kNN

- K-NN classifier: $k \geq 1$ nearest neighbors
  - Label for x predicted by majority voting among its $k-NN$
- $k = 5$, $x = [x_1, x_2]$



kNN classifier

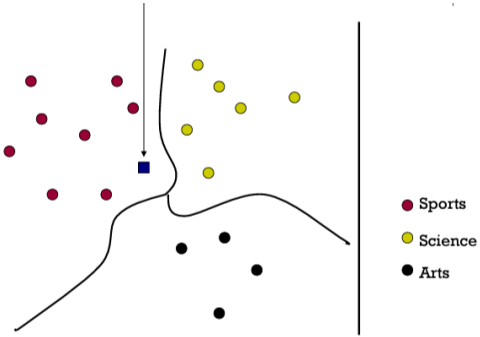
- Given
  - Training data $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\}$ are simply stored.
- To classify x:
  - Find k nearest training samples to x
  - Out of these k samples, identify the number of samples k_j belonging to class C_j ($j = 1, \ldots, C$).
  - Assign x to the class C_{j^*} where $j^* = \underset{j=1,\ldots,c}{\arg\max} \; k_j$
- It can be considered as a **discriminative** method.

kNN classifier cont.

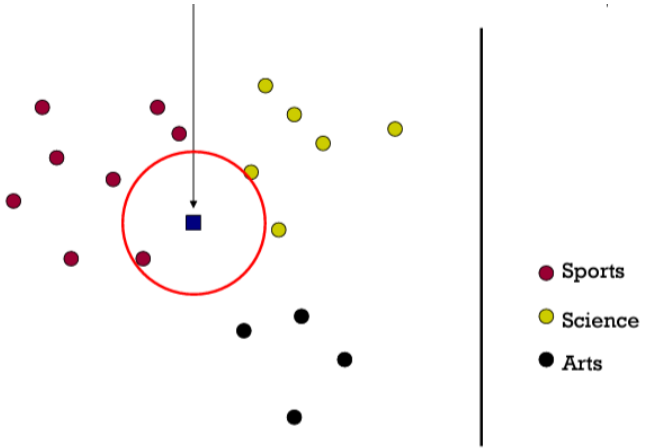
- With kNN we can obtain non-linear decision surfaces unlike the previous methods (linear and logistic regression)
- But note that this method could be prone to outliers or noisy data especially if:
  - We have small dataset
  - Our data is low-dimensional
  - We use a small value of k (like $k = 1$ is only determined by the nearest neighbor and could be misleading in many test cases.

kNN example

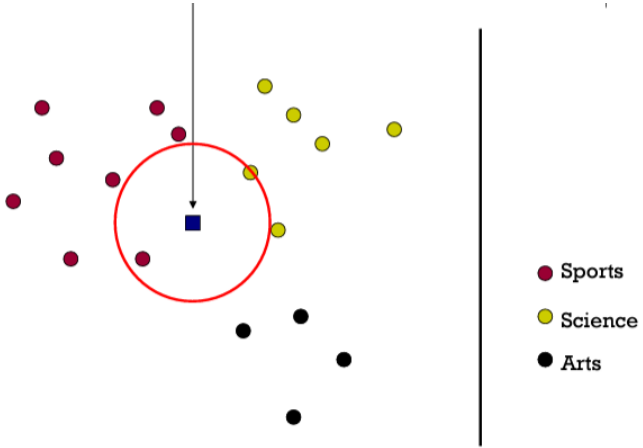
- We want to classify a new document and put it into one of three categories by studying its neighbor samples



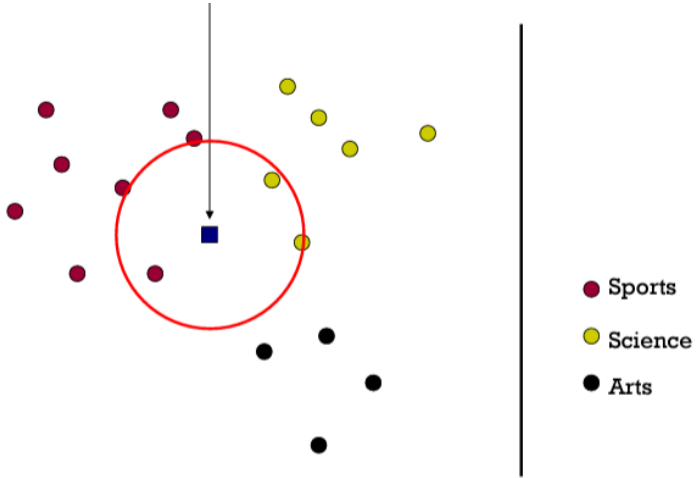
1-Nearest neighbor classifier



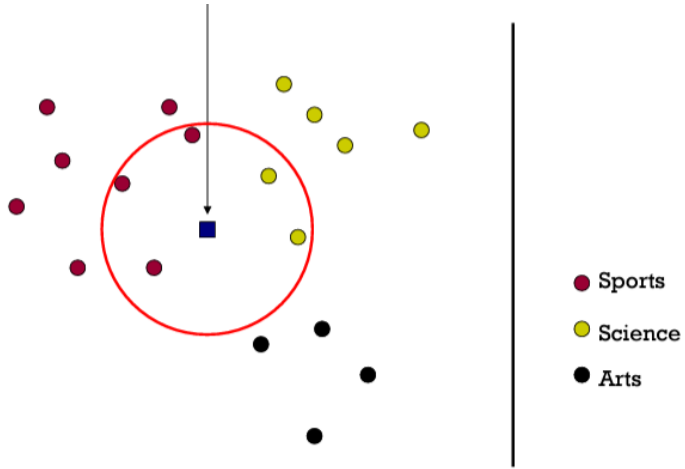
2-Nearest neighbor classifier



3-Nearest neighbor classifier

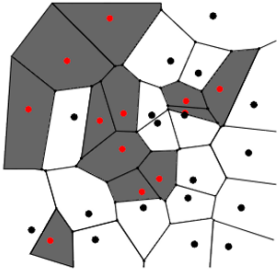


5-Nearest neighbor classifier



Voronoi tessellation

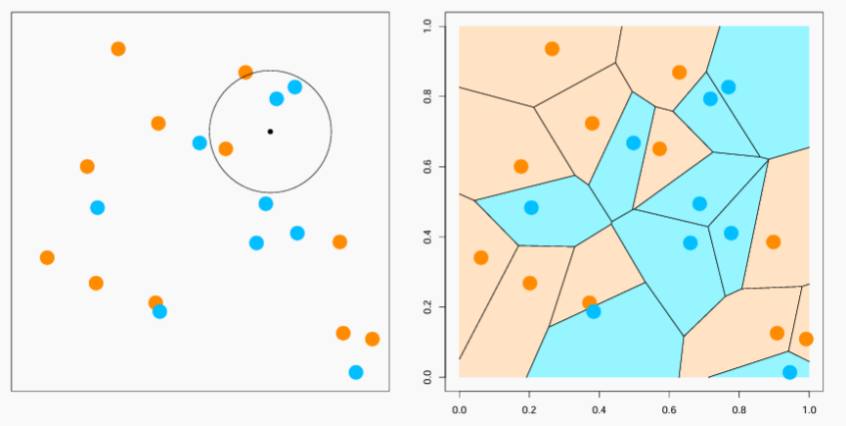
- Voronoi tessellation:
  - Each cell consists of all points closer to a given training point than to any other training points
  - All points in a cell are labeled by the category of the corresponding training point



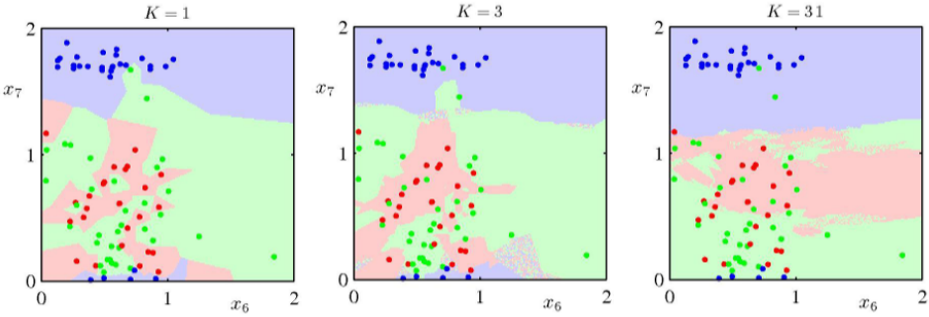
[Duda, Hurt, and Strok's Book]

Voronoi tessellation

- 1NN plot is a Voronoi tessellation



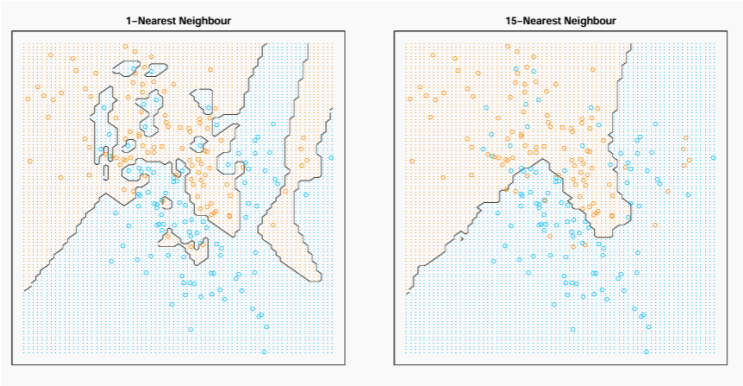
Effect of k



[Bishop]

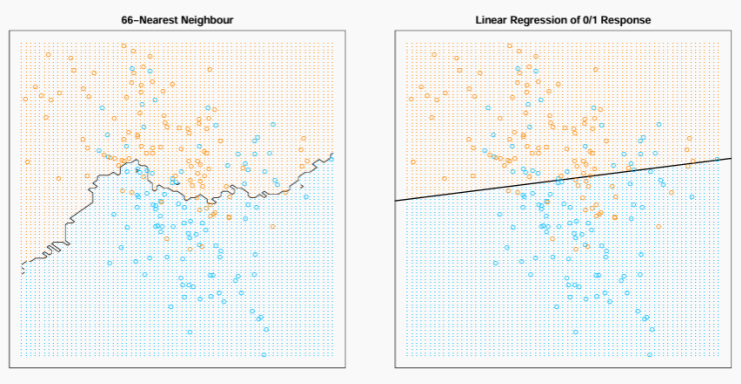
Effect of k cont.

- compare $k = 1$ with $k = 15$



Model complexity

- As we further increase k, the model tends to be less complex.
- Compare $66NN$ with a linear model that uses only 3 parameters:



Instance-based learner

- Main things to construct an instance-based learner:
  - A distance metric
  - Number of nearest neighbors of the test data that we look at
  - A weighting function (optional)
  - How to find the output based on neighbors?

Distance measures

- Euclidean distance

$$d(x, x') = \sqrt[2]{\|x - x'\|_2^2} = \sqrt[2]{(x_1 - x'_1)^2 + \cdots + (x_d - x'_d)^2}$$

- Distance learning methods for this purpose
  - Weighted Euclidean distance

$$d_w(x, x') = \sqrt[2]{w_1(x_1 - x'_1)^2 + \cdots + w_d(x_d - x'_d)^2}$$

Distance measures cont.

- Minkowski distance

$$d(x, x') = (\sum_{i=1}^{n} |x_i - x'_i|^p)^{\frac{1}{p}}$$

  - for $p \geq 1$ is a distance metric
  - As you can see Minkowski distance with $p = 2$ is the same as Euclidean distance
  - Minkowski distance is the same as L^p norm of $(x - x')$
- Remember L^p norm from linear algebra:

$$\|x\|_p = \sqrt[p]{(|x_1|^p + \cdots + |x_n|^p)}$$

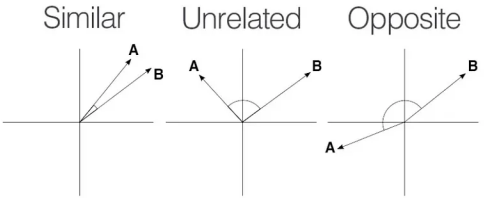
$$\text{Some famous } L^p \text{ norms} \begin{cases} \|x\|_1 & = \sum_{i=1}^{n} |x_i| \\ \|x\|_2 & = \sqrt{x_1^2 + \cdots + x_n^2} \\ \|x\|_\infty & = \max\{|x_1|, |x_2|, \ldots, |x_n|\} \end{cases}$$

Distance measures cont.

- Cosine distance (angle)

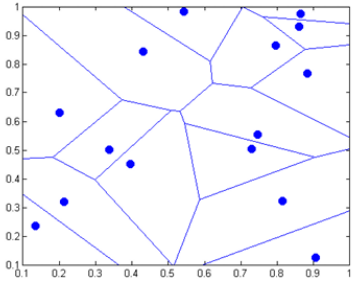
$$d(x, x') = 1 - \text{cosine similarity}(x, x')$$

$$\text{Where,} \quad \text{cosine similarity}(x, x') = \frac{x.x'}{\|x\|_2^2 \|x'\|_2^2} = \frac{\sum_{i=1}^{d} x_i x'_i}{\sqrt{\sum_{i=1}^{d} x_i^2} \sqrt{\sum_{i=1}^{d} x_i'^2}}$$

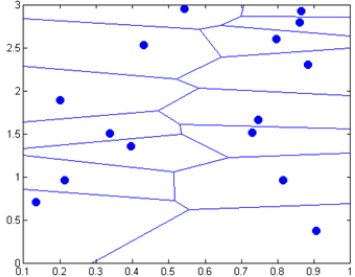


Example of angle difference for cosine similarity

Effect of distance measure



$$d(\boldsymbol{x}, \boldsymbol{x}') = \sqrt{(x_1 - x_1')^2 + (x_2 - x_2')^2}$$



$$d(\boldsymbol{x}, \boldsymbol{x}') = \sqrt{(x_1 - x_1')^2 + 3(x_2 - x_2')^2}$$

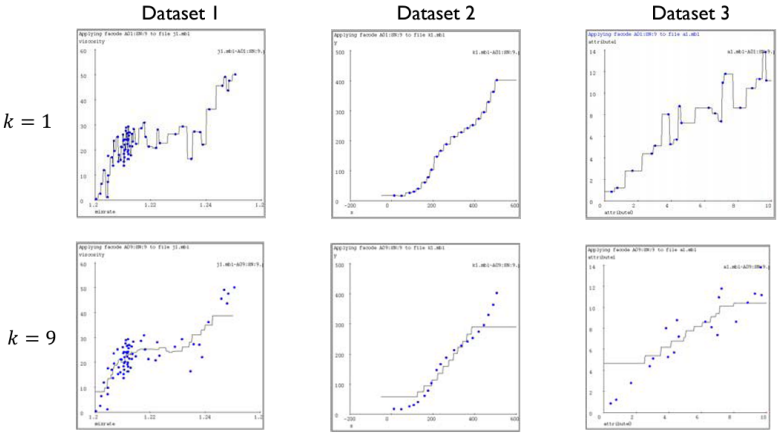
kNN regression

- Let $x'^{(1)}, \ldots, x'^{(k)}$ be the k nearest neighbors of x and $y'^{(1)}, \ldots, y'^{(k)}$ be their labels.

$$\hat{y} = \frac{1}{k} \sum_{j=1}^{k} y'^{(j)}$$

- Some problems of kNN regression for fitting functions:
  - Discontinuities in the estimated function
  - 1NN: noise-fitting problem
  - kNN ($k > 1$) smoothes away noise, but there could be other issues (e.g, flats the ends)

kNN regression cont.



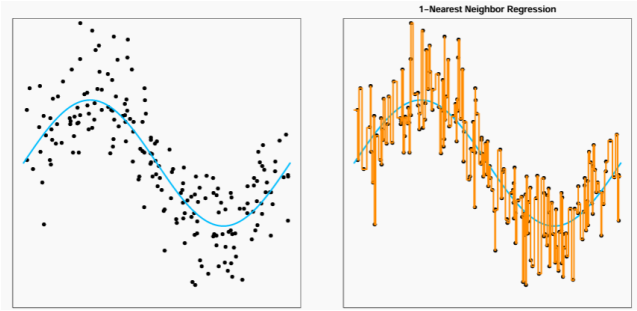
[Figs. have been adopted from Andrew Moore's tutorial on "Instance-based learning"]

kNN regression: example

- Suppose we have a dataset with only 1 feature from uniform $[0, 2\pi]$. The true model is:

$$Y = 2sin(X) + \epsilon$$

- Where ϵ is the standard normal error.
- First we simulate 200 observations, and see the model for $k = 1$

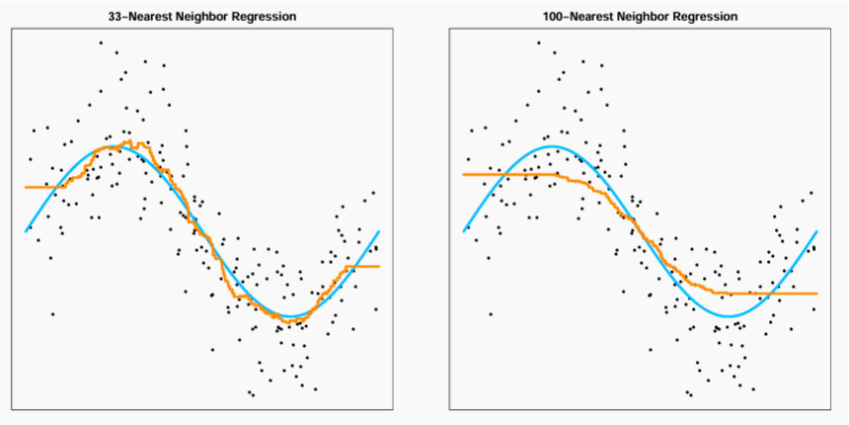


kNN regression: example

- Now for $k = 2$ and $k = 10$

kNN regression: example

- As you can see the model becomes smoother as k increases. However, this eventually deviates from the truth if k is too large



Accuracy in classification problems

- **Accuracy** is one of the simplest and most commonly used performance metrics.
- It is defined as the ratio of correctly predicted instances to the total instances:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Samples}}$$

- However, accuracy alone can be misleading, especially with imbalanced datasets.

Example: cancer detection problem

- Imagine a dataset with 1000 patients:
  - Only 10 have cancer (**positive class**).
  - 990 do not have cancer (**negative class**).
- A classifier predicts that no one has cancer (predicts all as negative).
- What will be the accuracy of this model?

Example: cancer detection problem cont

Look at this table for our model which predict negative all the time:

	Predicted Negative	**Predicted Positive**
Actual Negative	990 (TN)	0 (FP)
Actual Positive	10 (FN)	0 (TP)

$$\text{Accuracy} = \frac{990 + 0}{1000} = 99\%$$

High accuracy, but the model fails to detect any actual cases of cancer!

Why accuracy can be misleading

- In highly imbalanced datasets (e.g., cancer detection), the **minority class** (positive cases) is often underrepresented.
- A model that always predicts the majority class can still have high accuracy, but poor real-world performance.
- In the cancer detection example, 99% accuracy sounds good, but the model doesn't detect any actual cancer cases.
- We need better metrics to evaluate model performance.

Performance metrics

- **Scenario:**
  - An alarm system can either ring or not ring when a thief is present.
  - Let's define the outcomes:
    - **True Positive (TP)**: Alarm rings (correctly) when a thief is present.
    - **True Negative (TN)**: Alarm does not ring (correctly) when no thief is present.
    - **False Positive (FP)**: Alarm rings (incorrectly) when no thief is present (a false alarm).
    - **False Negative (FN)**: Alarm does not ring (incorrectly) when a thief is present (a missed alarm).

	Thief Present	**No Thief Present**
Alarm Rings	TP	FP
Alarm Does Not Ring	FN	TN

Performance metrics cont.

- **Metrics:**
  - **Sensitivity (Recall)**:

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

Indicates the ability of the alarm system to correctly identify a thief. It is the proportion of actual positives (thief present) that are correctly identified.

  - **Specificity**:

$$\text{Specificity} = \frac{TN}{TN + FP}$$

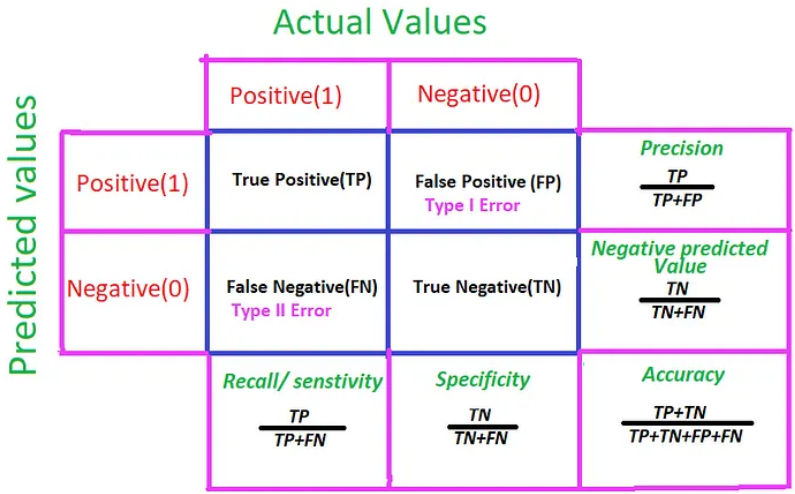
Measures the ability of the alarm system to correctly identify when no thief is present. It is the proportion of actual negatives that are correctly identified.

  - **Precision**:

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

Indicates the accuracy of the alarm when it rings. It is the proportion of times the alarm rang and a thief was indeed present out of all the times the alarm was activated.

Performance metrics cont.



A combined measure: F

- Combined measure: **F measure**
  - allows us to trade off precision and recall
  - weighted harmonic mean of P and R

$$F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

You can see: $\beta^2 = \frac{1-\alpha}{\alpha}$

A combined measure: F cont.

- People usually use balanced F ($\beta = 1$ or $\alpha = \frac{1}{2}$)

$$F = F_{\beta=1}$$

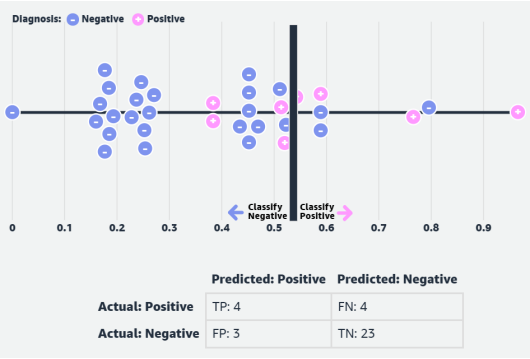
$$F = \frac{2PR}{P+R}$$

- Harmonic mean of P and R:

$$\frac{1}{F} = \frac{1}{2}(\frac{1}{P} + \frac{1}{R})$$

Precision/recall/F1

- This website could give you a perfect intuition about precision recall trade-off



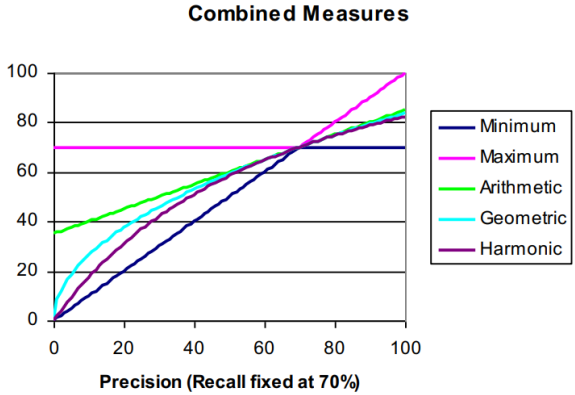
- Link: https://mlu-explain.github.io/precision-recall/

Why harmonic mean?

- Why don't we use a different mean of P and R as a measure?
  - e.g., the arithmetic mean
- The simple (arithmetic) mean is 50% for "return true for every thing", which is too high.
- Desideratum: Punch really bad performance either on precision or recall
  - Taking the minimum achieves this.
  - F (harmonic mean) is a kind of **smooth minimum**.

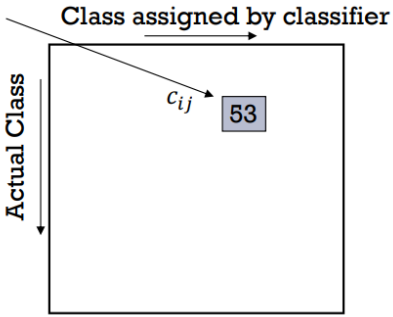
F1 and other averages

- Harmonic mean is a conservative average. We can view the harmonic mean as a kind of soft minimum



Confusion matrix

- This (i, j) entry means 53 of the samples actually in class i were put in class j by the classifier:



- In a perfect classification, only the diagonal has non-zero entries

Per class evaluation measures

- Recall: Fraction of the samples in class i classified correctly:

$$\frac{c_{ii}}{\sum_j c_{ij}}$$

- Precision: Fraction of the samples assigned class i that are actually about class i:

$$\frac{c_{ii}}{\sum_j c_{ji}}$$

- Accuracy: Fraction of the samples classified correctly:

$$\frac{\sum_i c_{ii}}{\sum_j \sum_i c_{ij}}$$

Averaging: macro vs. micro

- We now have an evaluation measure (F1) for one class.
- But we also want a single number that shows aggregate performance over all classes

Micro- vs. Macro-Averaging

- If we have more than one class, how do we combine multiple performance measures into one quantity?
- Macroaveraging: Compute performance for each class, then average
  - Compute F1 for each of the C classes
  - Average these C numbers
- Microaveraging: Collect decisions for all classes, aggregate them and then compute measure.
  - Compute TP, FP, FN for each of the C classes.
  - Sum these C numbers(e.g, all TP to get aggregate TP)
  - Compute F1 for aggregate TP, FP, FN

Micro- vs. Macro-Averaging: example

Class 1

	Truth: yes	Truth: no
Classifier: yes	10	10
Classifier: no	10	970

Class 2

	Truth: yes	Truth: no
Classifier: yes	90	10
Classifier: no	10	890

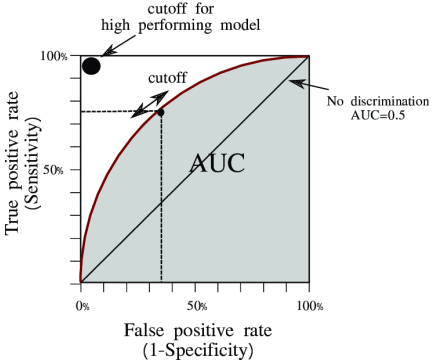
Micro Ave. Table

	Truth: yes	Truth: no
Classifier: yes	100	20
Classifier: no	20	1860

- Macroaveraged precision: $(0.5 + 0.9)/2 = 0.7$
- Microaveraged precision: $100/120 = 0.83$
- Microaveraged score is dominated by score on common classes

AUC-ROC

- Area Under the Receiver Operating Characteristic Curve
  - ROC (Receiver Operating Characteristic) is a graphical representation of the performance of a binary classification model.
  - It plots the true positive rate (TPR) against the false positive rate (FPR) at different classification thresholds.



AUC-ROC cont.

- A high AUC score indicates that the model has good discrimination ability, i.e., it can effectively differentiate between positive and negative instances at different classification thresholds.
- Conversely, a lower AUC-ROC score suggests that the model struggles to differentiate between the two classes.
- AUC ranges from 0 to 1, with 0.5 indicating random guessing and 1 indicating a perfect classifier.

Model Selection via Cross Validation

- **Cross-Validation**
  - **Purpose**: Technique for evaluating how well a model generalizes to unseen data.
  - **How It Works**: Split data into k folds; train on $k-1$ folds and validate on the remaining fold.
  - **Repeat Process**: Repeat k times, rotating the test fold each time. Average of all scores is the final score of the model.
  - Cross-validation reduces overfitting and provides a more reliable estimation of model performance.

K-Fold Cross Validation

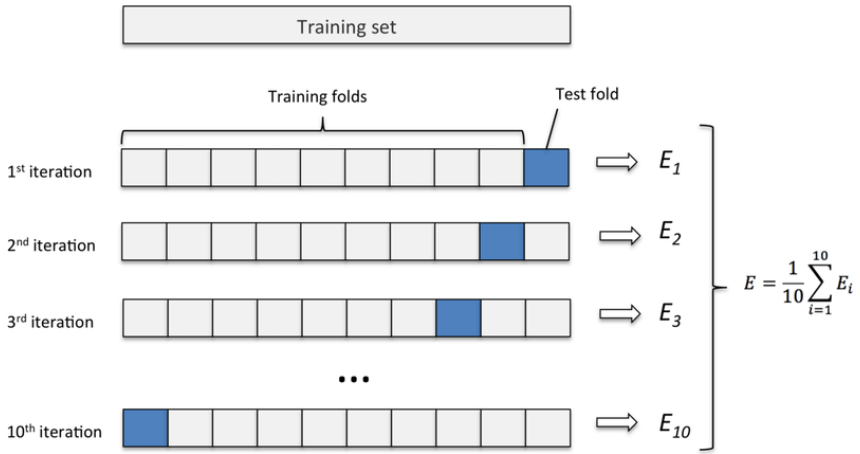
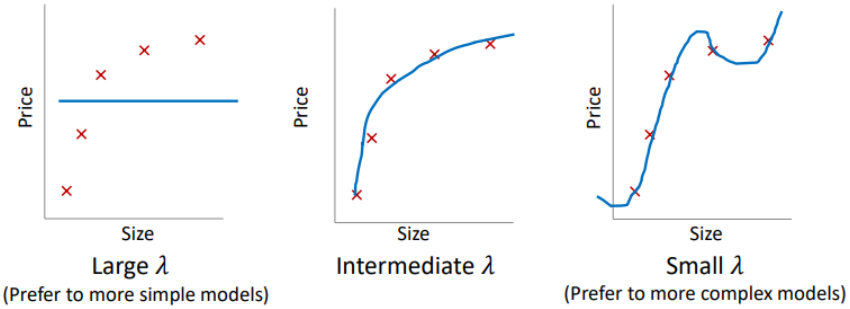


Figure adapted from Introduction to Support Vector Machines and Kernel Methods, J.M. Ashfaque.

Leave-One-Out Cross-Validation (LOOCV)

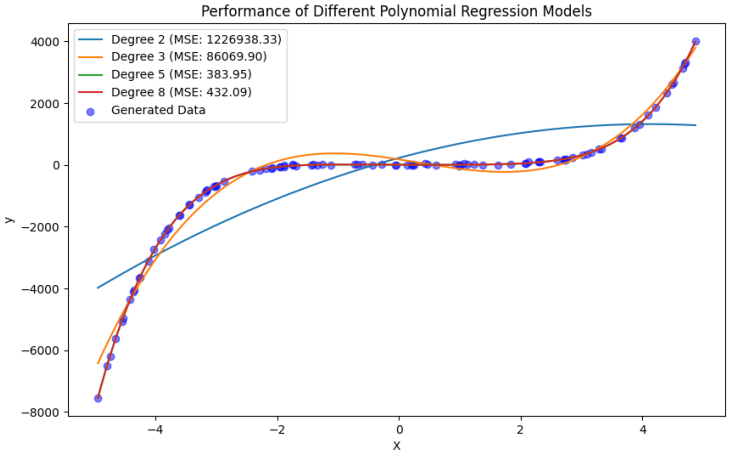
- **Leave-One-Out Cross-Validation (LOOCV)**
  - **How It Works**: Uses a single data point as the validation set ($k = 1$) and the rest as the training set. Repeat for all data points.
  - **Properties:**
    - **No Data Wastage**: Every data point is used for both training and validation.
    - **High Variance, Low Bias**.
    - **Computationally Expensive**: Requires training the model N times for N data points, making it slow for large datasets.
    - **Best for small datasets**.

Cross-Validation for Choosing Regularization Term



Figures adapted from slides of F.Salehi, Machine Learning course, Sharif University of Technology.

Cross-Validation for Choosing Model Complexity



Contributions

- **These slides are authored by:**
  - Danial Gharib
  - Mahan Bayhaghi

[1] C. M., *Pattern Recognition and Machine Learning*.
Information Science and Statistics, New York, NY: Springer, 1 ed., Aug. 2006.

[2] M. Soleymani Baghshah, "Machine learning." Lecture slides.

[3] M. Soleymani Baghshah, "Modern information retrieval." Lecture slides.

[4] T. Mitchell, *Machine Learning*.
McGraw-Hill series in computer science, New York, NY: McGraw-Hill Professional, Mar. 1997.

[5] R. Zhu, "Stat 542: Statistical learning - k-nearest neighbor and the bias-variance trade-off." Lecture notes.

[6] E. Xing, "Theory of classification and nonparametric classifier." Lecture notes.