ARISTOTLE UNIVERSITY OF THESSALONIKI

FACULTY OF ENGINEERING

Pattern Recognition & Machine Learning

Our first model

Panagiotis C. Petrantonakis

Assistant Professor

Dept. of Electrical and Computer Engineering

ppetrant@ece.auth.gr

Fall Semester

Objectives of this lecture

- Build our first model
- Highlight the difference between generalization and simple memorization
- Strict formalization of the learning problem
- Sources of errors
- How to tune a learning model
- How to evaluate a learning model

What is learning?

- What about learning in real life?
- At the end of the course, you will be expected to have “learned” all about PR&ML topic.
- How are you going to evaluate learning?

What is learning?

- What about learning in real life?
- At the end of the course, you will be expected to have “learned” all about PR&ML topic.
- How are you going to evaluate learning?
  - Exam!
- What is a good exam?

What is learning?

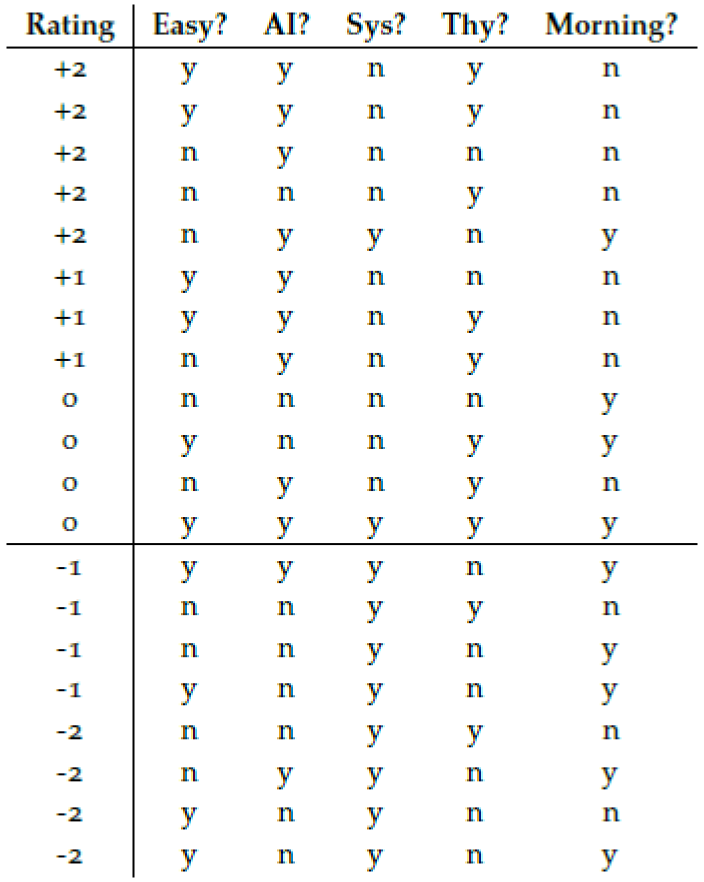
- What about learning in real life?
- At the end of the course, you will be expected to have “learned” all about PR&ML topic.
- How are you going to evaluate learning?
  - Exam!
- What is a good exam?
  - observe specific examples from the course, and then answer new, but related questions on the exam (generalize)

What is learning?

- What about learning in real life?
- At the end of the course, you will be expected to have “learned” all about PR&ML topic.
- How are you going to evaluate learning?
  - Exam!
- What is a good exam?
  - observe specific examples from the course, and then answer new, but related questions on the exam (generalize)
- ***Generalization*** is perhaps the most central concept in (machine) learning.

A toy dataset

- Think of a recommendation system for undergraduate courses
- The evaluation is a score from -2 to 2
- A recommender system could **predict** how much a student would like a particular course.
  - At a basic level, machine learning is about predicting the future based on the past
- What would be a bad “exam” for this system?



Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2	y	y	n	y	n
+2	y	y	n	y	n
+2	n	y	n	n	n
+2	n	n	n	y	n
+2	n	y	y	n	y
+1	y	y	n	n	n
+1	y	y	n	y	n
+1	n	y	n	y	n
0	n	n	n	n	y
0	y	n	n	y	y
0	n	y	n	y	n
0	y	y	y	y	y
-1	y	y	y	n	y
-1	n	n	y	y	n
-1	n	n	y	n	y
-1	y	n	y	n	y
-2	n	n	y	y	n
-2	n	y	y	n	y
-2	y	n	y	n	n
-2	y	n	y	n	y

Hal Daume III

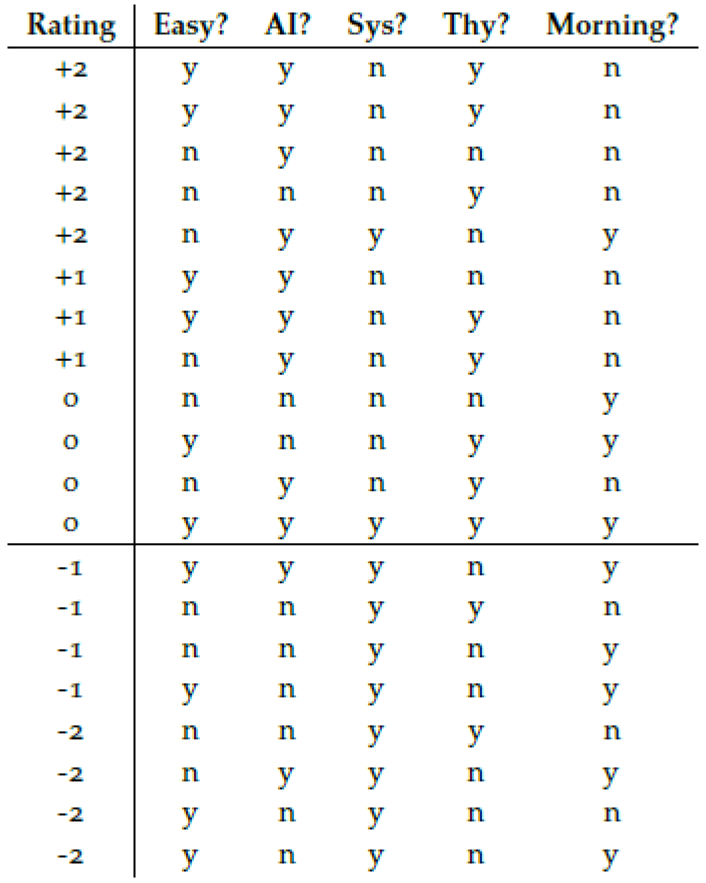
A toy dataset

- Think of a recommendation system for undergraduate courses

- The evaluation is a score from -2 to 2

- A recommender system could **predict** how much a student would like a particular course.
  - At a basic level, machine learning is about predicting the future based on the past

- What would be a bad “exam” for this system?
  - Ask it for course that is not related o Computer Science or
  - Ask it for a recommendation about a course that is already registered in the dataset for a particular student

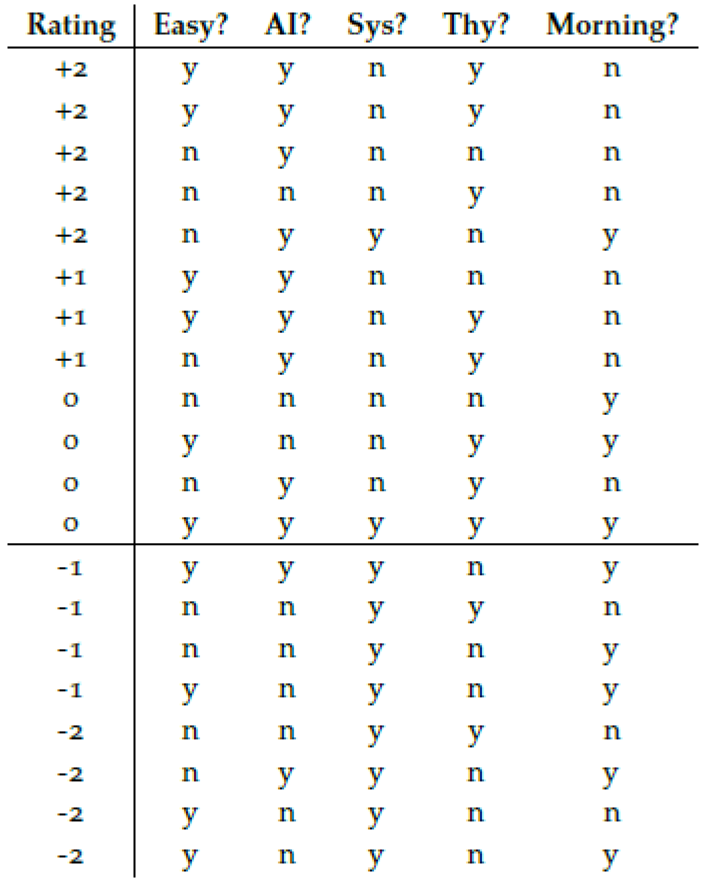


Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2	y	y	n	y	n
+2	y	y	n	y	n
+2	n	y	n	n	n
+2	n	n	n	y	n
+2	n	y	y	n	y
+1	y	y	n	n	n
+1	y	y	n	y	n
+1	n	y	n	y	n
0	n	n	n	n	y
0	y	n	n	y	y
0	n	y	n	y	n
0	y	y	y	y	y
-1	y	y	y	n	y
-1	n	n	y	y	n
-1	n	n	y	n	y
-1	y	n	y	n	y
-2	n	n	y	y	n
-2	n	y	y	n	y
-2	y	n	y	n	n
-2	y	n	y	n	y

Hal Daume III

A toy dataset

- The objects that our algorithm will make predictions about are examples. (they are actually samples of the dataset; *feature vector* and *label*)
- In this dataset the example would be any set of values for a particular course (*feature values*) and the corresponding answer (*label*).
- The desired prediction would be the rating that a (*particular*) student would give to Algorithms. (in this example we will only work on the subject dependent prediction)
- Thus we first train our predictor and then test it



Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2	y	y	n	y	n
+2	y	y	n	y	n
+2	n	y	n	n	n
+2	n	n	n	y	n
+2	n	y	y	n	y
+1	y	y	n	n	n
+1	y	y	n	y	n
+1	n	y	n	y	n
0	n	n	n	n	y
0	y	n	n	y	y
0	n	y	n	y	n
0	y	y	y	y	y
-1	y	y	y	n	y
-1	n	n	y	y	n
-1	n	n	y	n	y
-1	y	n	y	n	y
-2	n	n	y	y	n
-2	n	y	y	n	y
-2	y	n	y	n	n
-2	y	n	y	n	y

Hal Daume III

Learning problems

- The goal of ML is to use training data and induct a function f. This function takes as input the feature vector and outputs a value. This value is:
    - a real value for regression problems
    - 0 or 1 for binary classification
    - a number of classes > 2 for multiclass classification
    - order of relevance in ranking problems (e.g., PageRank by Google)
- Why is it important to make this break down?

Learning problems

- The goal of ML is to use training data and induct a function f. This function takes as input the feature vector and outputs a value. This value is:
  - a real value for regression problems
  - 0 or 1 for binary classification
  - a number of classes > 2 for multiclass classification
  - order of relevance in ranking problems (e.g., PageRank by Google)
- Why is it important to make this break down?
  - measuring error!

The Decision Tree Model (DT)

- We will now examine the training process of a DT model under the simplest classification case: binary classification.

- We will develop a DT predictor that will answer whether some unknown user will enjoy some unknown course.

- We are allowed to ask binary questions about the course under consideration

- The main goals of the training process of a DT model:
  - what questions to ask,
  - in what order to ask them
  - what answer to predict once you have asked enough questions.

The Decision Tree Model - Training

- It is computationally infeasible to consider all of the possible DT

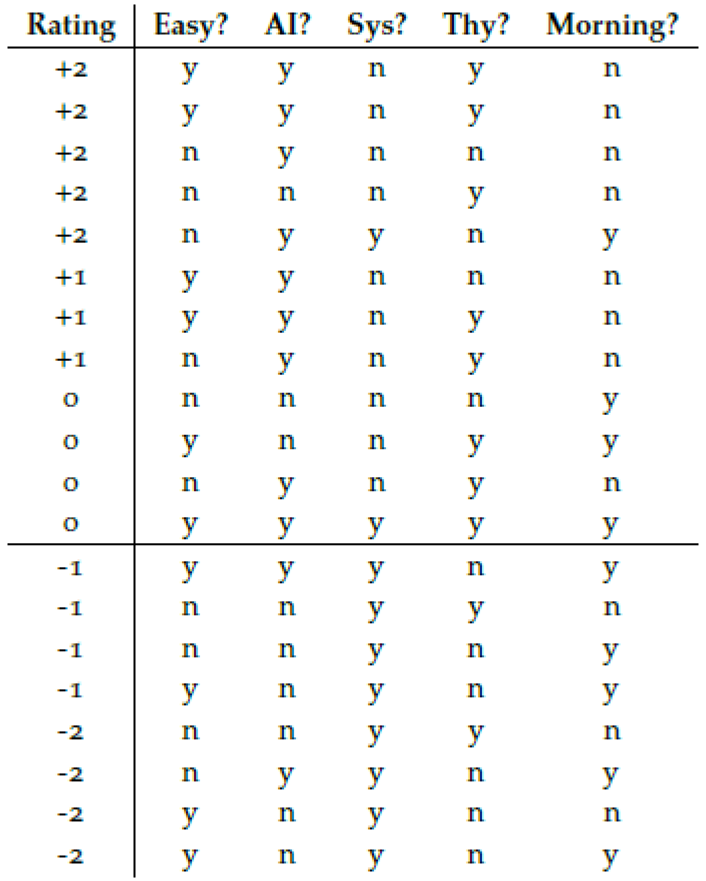
- We will try to train our DT model greedily. We will begin by asking:

  If I could only ask one question (feature), what question would I ask?

  this actually means:

  what is my best feature?

- The histogram of labels for each feature would help



Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2	y	y	n	y	n
+2	y	y	n	y	n
+2	n	y	n	n	n
+2	n	n	n	y	n
+2	n	y	y	n	y
+1	y	y	n	n	n
+1	y	y	n	y	n
+1	n	y	n	y	n
0	n	n	n	n	y
0	y	n	n	y	y
0	n	y	n	y	n
0	y	y	y	y	y
-1	y	y	y	n	y
-1	n	n	y	y	n
-1	n	n	y	n	y
-1	y	n	y	n	y
-2	n	n	y	y	n
-2	n	y	y	n	y
-2	y	n	y	n	n
-2	y	n	y	n	y

Hal Daume III

The Decision Tree Model - Training

- It is computationally infeasible to consider all of the possible DT

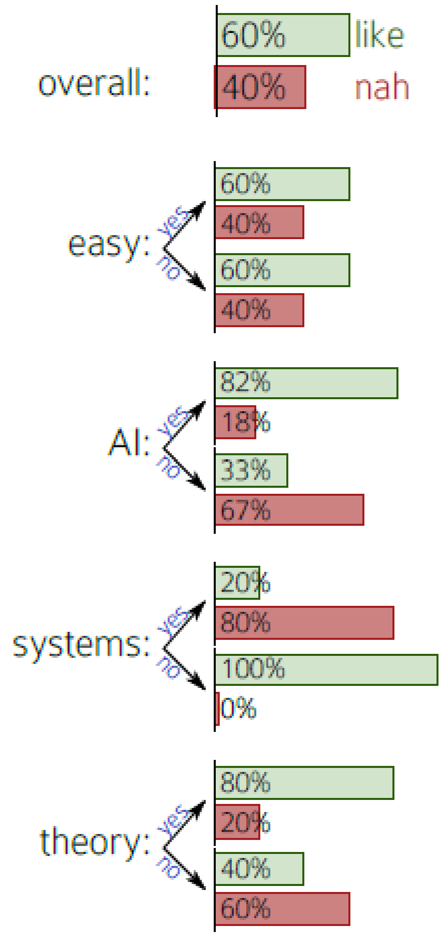
- We will try to train our DT model greedily. We will begin by asking:

  If I could only ask one question (feature), what question would I ask?

  this actually means:

  what is my best feature?

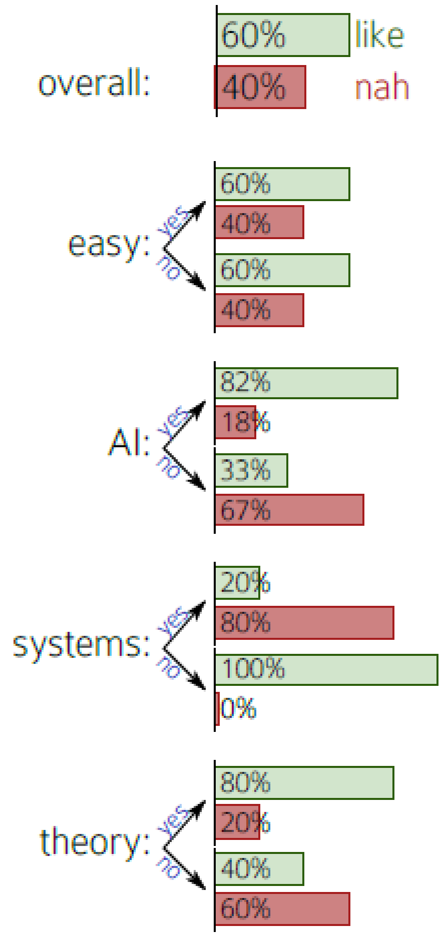
- The histogram of labels for each feature would help



Hal Daume III

The Decision Tree Model - Training

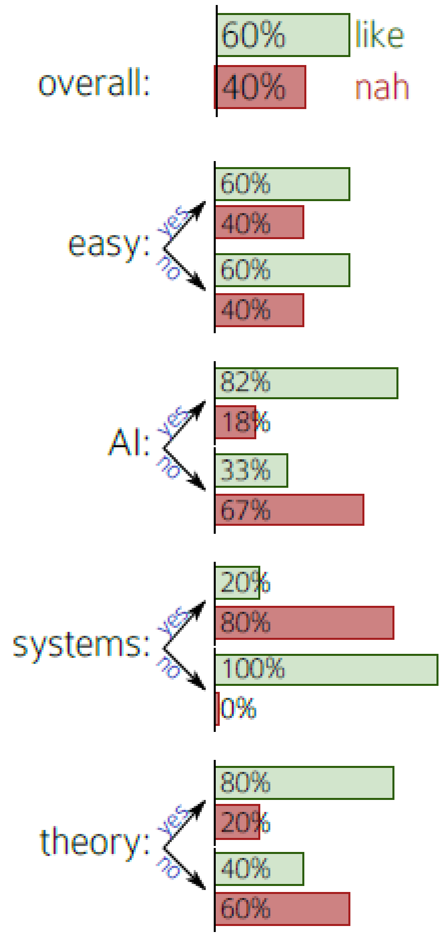
- We have to consider each feature in turn.
- Is the first feature useful?
- What is the best feature?



Hal Daume III

The Decision Tree Model - Training

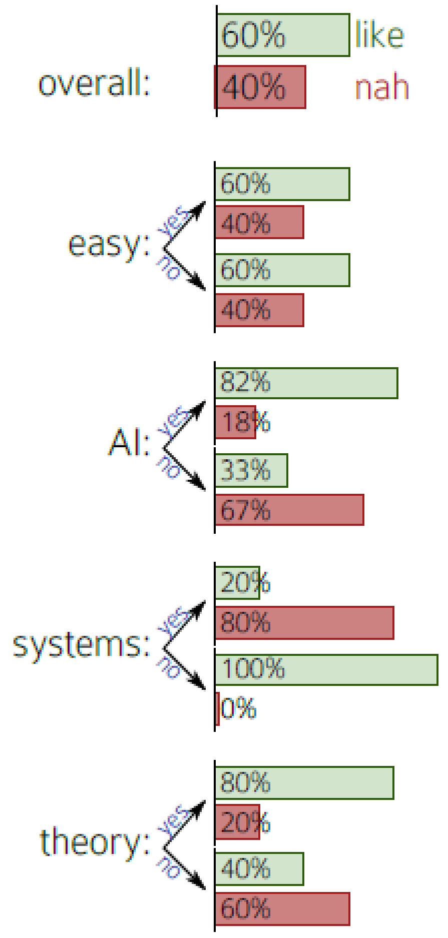
- We have to consider each feature in turn.
- Is the first feature useful?
- What is the best feature?
- **Systems**
  - two possible values: yes or no
- suppose you were to ask this question on a random example and observe a value of “no”, what label you would guess?



Hal Daume III

The Decision Tree Model - Training

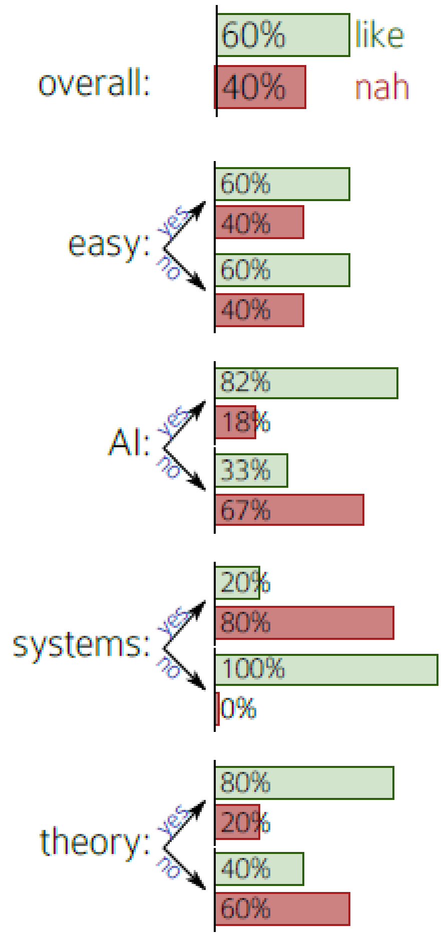
- We have to consider each feature in turn.
- Is the first feature useful?
- What is the best feature?
- **Systems**
  - two possible values: yes or no
- suppose you were to ask this question on a random example and observe a value of “no”, what label you would guess?
  - **“like”**



Hal Daume III

The Decision Tree Model - Training

- Thus for this single feature you know what you would guess if you had to.
- How many examples would I classify correctly in the training set?



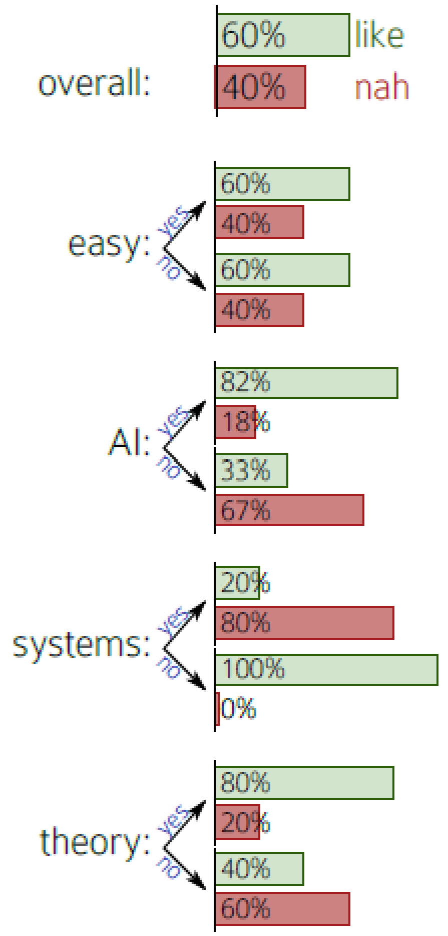
Hal Daume III

The Decision Tree Model - Training

- Thus for this single feature you know what you would guess if you had to.

- How many examples would I classify correctly in the training set?

- The answer is: $\frac{18}{20} \sim 90\%$ (the highest score among the features!)



Hal Daume III

The Decision Tree Model - Training

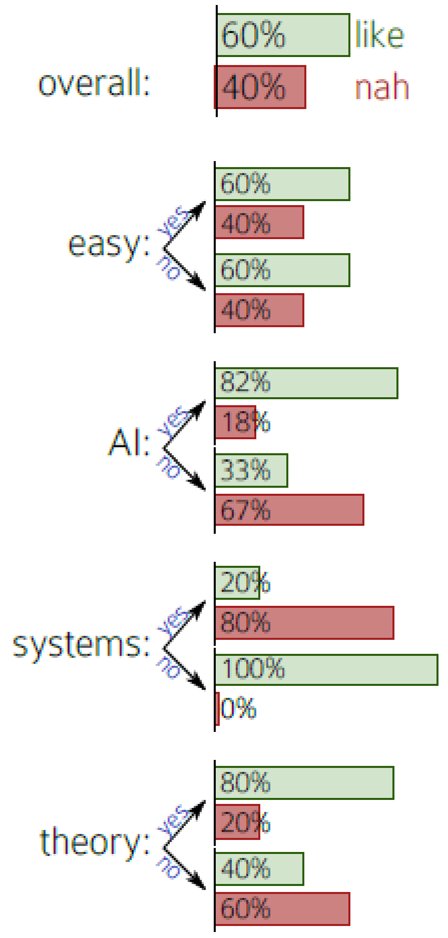
- Thus for this single feature you know what you would guess if you had to.

- How many examples would I classify correctly in the training set?

- The answer is: $\frac{18}{20} \sim 90\%$ (the highest score among the features!)

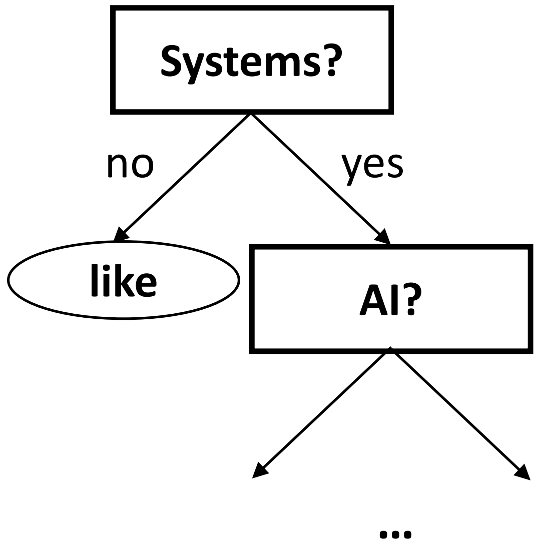
- We put this feature at the root of the DT and continue with the rest of the features by asking again only for the samples that are marked with yes or no:

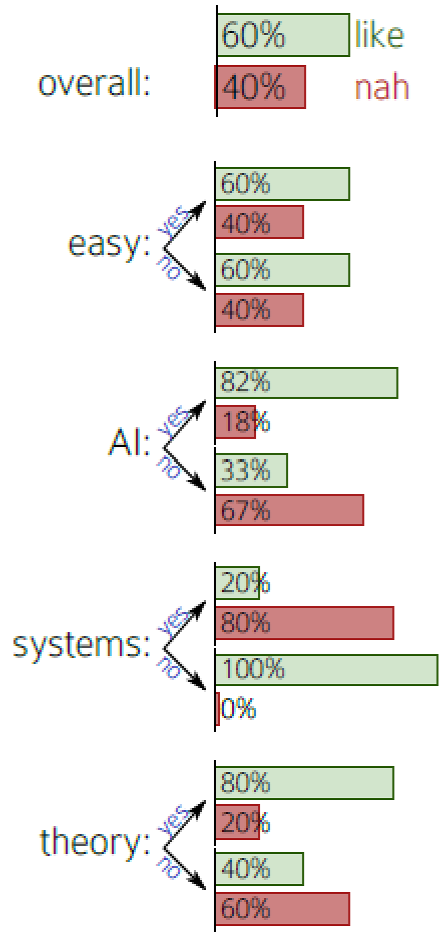
  What is the best feature? (highest score)



Hal Daume III

The Decision Tree Model - Training





Hal Daume III

- At some point it will become useless to query on additional features and then we return the decision (labels):
  - unambiguous data
  - no remaining features
- The prediction algorithm just recurses down the DT and return a guess associated with a particular leaf.

The loss function

- Until now we have seen that:
  - The performance measure depends on the problem we are trying to solve (regression, binary, multiclass, ranking)
  - The performance should be measured on **unseen** data (test set).
  - There should be strong relation between the training and test data.

- To construct a performance measure we define a *loss function.*

- *Different learning problems require different loss functions*

  - *Regression:* $\ell(y, \hat{y}) = (y - \hat{y})^2$ *or* $\ell(y, \hat{y}) = |y - \hat{y}|$
  - *Binary classification:* $\ell(y, \hat{y}) = \begin{cases} 0, & y = \hat{y} \\ 1, & otherwise \end{cases}$
  - Multiclass: same as binary classification (zero/one loss)

The data generation distribution

- Previously:
  - The performance should be measured on **unseen** data (test set).
  - There should be strong relation between the training and test data.

- Where the training and test data are coming from?

The data generation distribution

- Previously:
  - The performance should be measured on **unseen** data (test set).
  - There should be strong relation between the training and test data.

- Where the training and test data are coming from?
  - Probability distribution $p_{\mathcal{D}}$ that generates the input output pairs.

The data generation distribution

- Previously:
  - The performance should be measured on **unseen** data (test set).
  - There should be strong relation between the training and test data.

- Where the training and test data are coming from?
  - Probability distribution $p_{\mathcal{D}}$ that generates the input output pairs.

- Thus, $p_{\mathcal{D}}(\boldsymbol{x}, y)$ is a probability distribution over $(\boldsymbol{x}, y)$ pairs where x is the feature vector values and y the corresponding label.

- We say that $p_{\mathcal{D}}(\boldsymbol{x}, y)$ generates our data. In essence, it gives high probability to "reasonable" pairs and low probability to "unreasonable" ones.

- The hardest thing about ML is that we don't know what $p_{\mathcal{D}}(\boldsymbol{x}, y)$ is: all we get is a random sample from it. This random sample is our data.

Formalizing the learning problem

- **Recap**:
  - We get access to training data (random sample drawn from $p_{\mathcal{D}}(\boldsymbol{x}, y)$) and through a training process we seek to get a function f that maps inputs $\boldsymbol{x}$ to a corresponding prediction $\hat{y} = f(\boldsymbol{x})$.
  - f should perform well, in terms of the loss function $\ell(y, \hat{y})$ in future samples that are also drawn from $p_{\mathcal{D}}(\boldsymbol{x}, y)$

- Expected loss:

$$\epsilon = \mathbb{E}_{\boldsymbol{x},y\sim p_{\mathcal{D}}(\boldsymbol{x},y)}[\ell(y, f(\boldsymbol{x}))] = \sum_{(\boldsymbol{x},y)} p_{\mathcal{D}}(\boldsymbol{x}, y)\, \ell(y, f(\boldsymbol{x}))$$

- What we ultimately want is to minimize ϵ. The problem is that we do not know $p_{\mathcal{D}}(\boldsymbol{x}, y)$!!

Formalizing the learning problem

- Expected loss:

$$\epsilon = \mathbb{E}_{\boldsymbol{x},y\sim p_{\mathcal{D}}(\boldsymbol{x},y)}[\ell(y, f(\boldsymbol{x}))] = \sum_{(\boldsymbol{x},y)} p_{\mathcal{D}}(\boldsymbol{x}, y)\, \ell(y, f(\boldsymbol{x}))$$

- Instead we can minimize training error.
- Suppose we have training set D that consists of N samples, i.e., $(\boldsymbol{x}, y)$ pairs. Then the training error is:

$$\hat{\epsilon} = \frac{1}{N}\sum_{n=1}^{N} \ell(y_n, f(\boldsymbol{x}_n))$$

- We can drive $\hat{\epsilon}$ to zero by simply…

Formalizing the learning problem

- Expected loss:

$$\epsilon = \mathbb{E}_{\boldsymbol{x},y\sim p_{\mathcal{D}}(\boldsymbol{x},y)}[\ell(y, f(\boldsymbol{x}))] = \sum_{(\boldsymbol{x},y)} p_{\mathcal{D}}(\boldsymbol{x}, y)\, \ell(y, f(\boldsymbol{x}))$$

- Instead we can minimize training error.
- Suppose we have training set D that consists of N samples, i.e., $(\boldsymbol{x}, y)$ pairs. Then the training error is:

$$\hat{\epsilon} = \frac{1}{N}\sum_{n=1}^{N} \ell(y_n, f(\boldsymbol{x}_n))$$

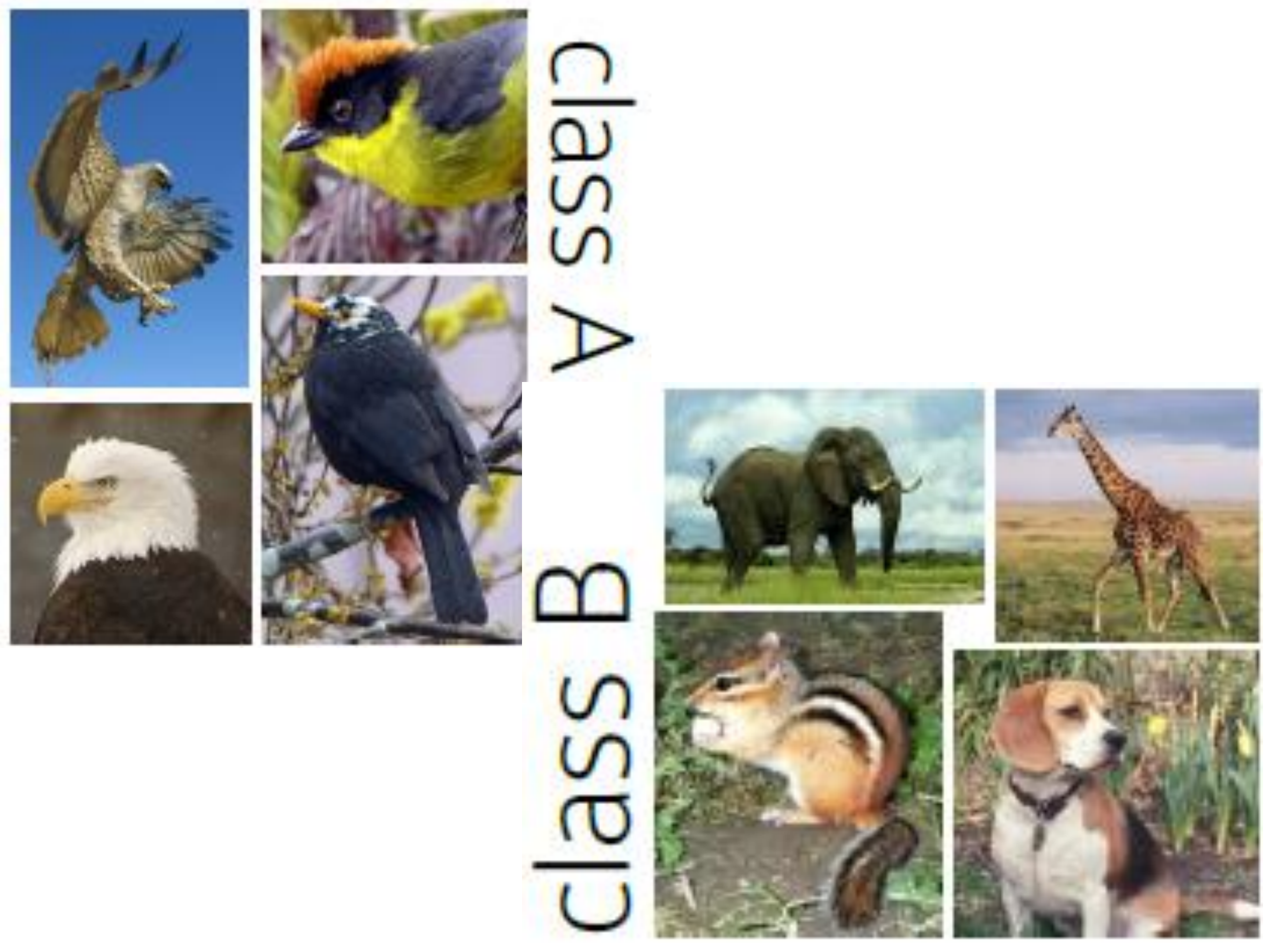
- We can drive $\hat{\epsilon}$ to zero by simply… memorizing our training data! But this would lead to poor generalization!

Formalizing the learning problem

- Altogether:
  - given a loss function $\ell(y, f(\boldsymbol{x}))$
  - a sample D drawn from $p_{\mathcal{D}}(\boldsymbol{x}, y)$
  - we have to compute a function $f(\boldsymbol{x}) = \hat{y}$ with
  - low expected error over $p_{\mathcal{D}}(\boldsymbol{x}, y)$ w.r.t. $\ell(y, f(\boldsymbol{x}))$
- Do not forget that:
  - distribution D for training data must match the distribution D for the test data
  - ML algorithms cannot generalize beyond the data distribution it has seen during training time!

Examine train data

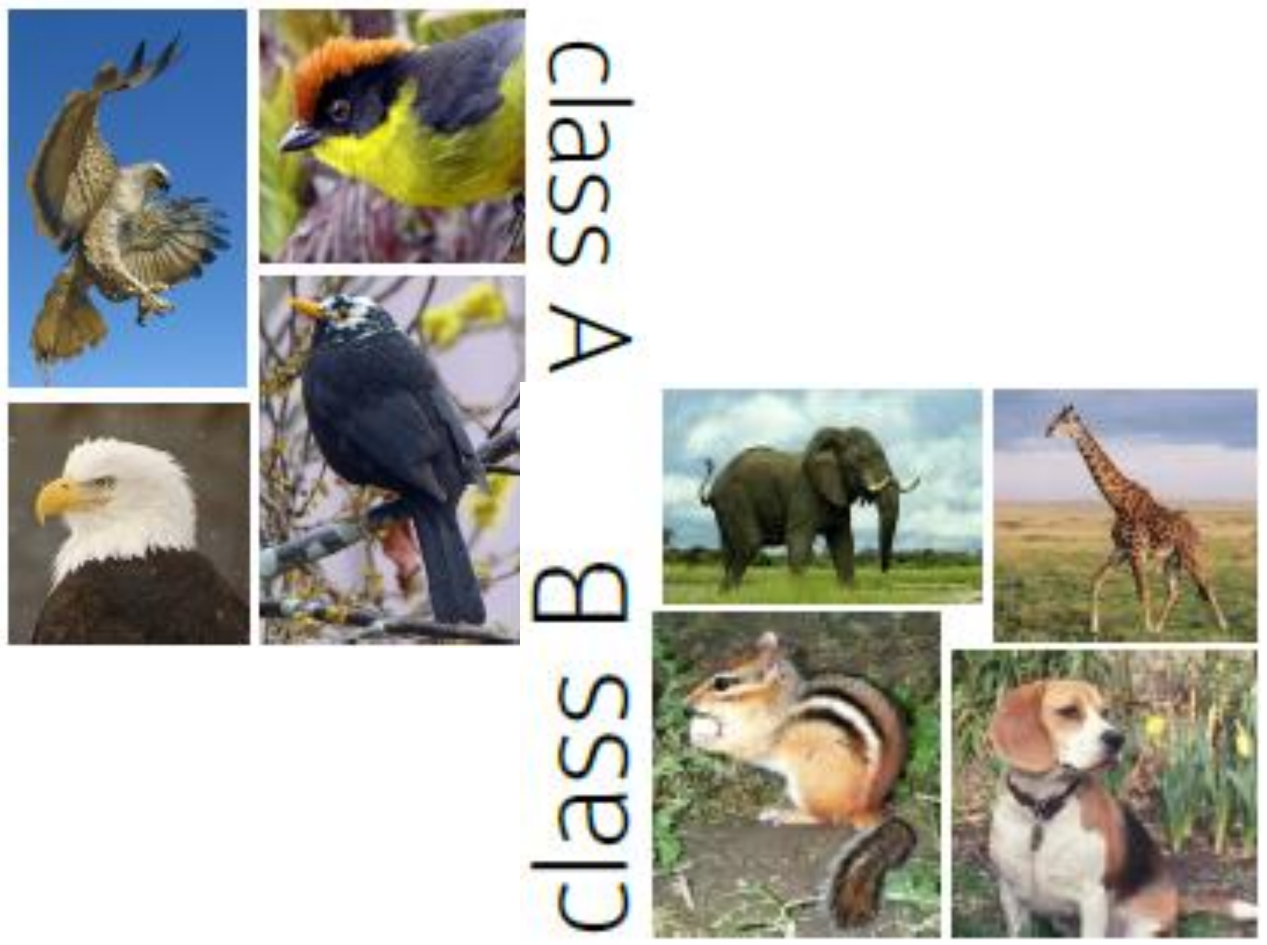
Training data



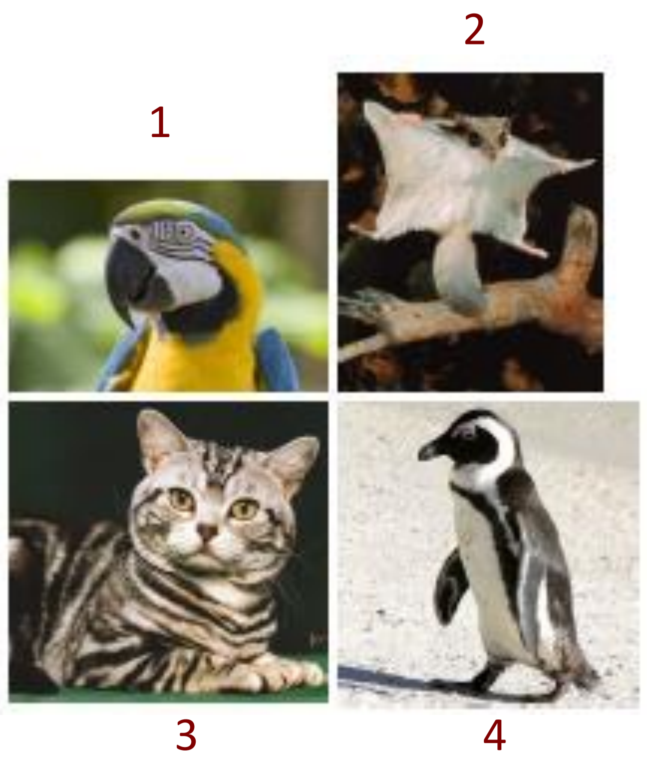
Hal Daume III

Make a guess!

Training data



Test data

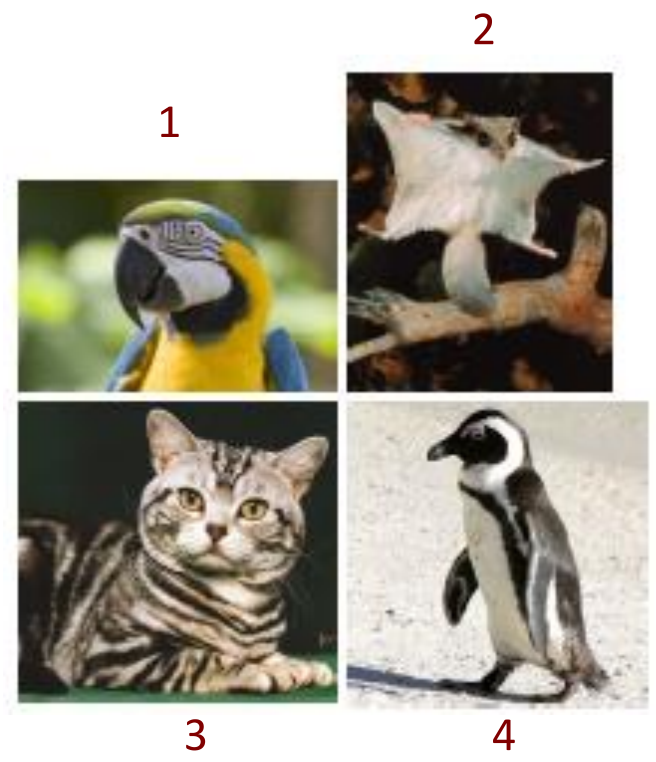


Hal Daume III

Inductive bias

- Most probably you guessed either
  - ABBA ($p = 0.6 - 0.7$) or
  - AABB ($p = 0.3 - 0.4$)

Test data



Hal Daume III

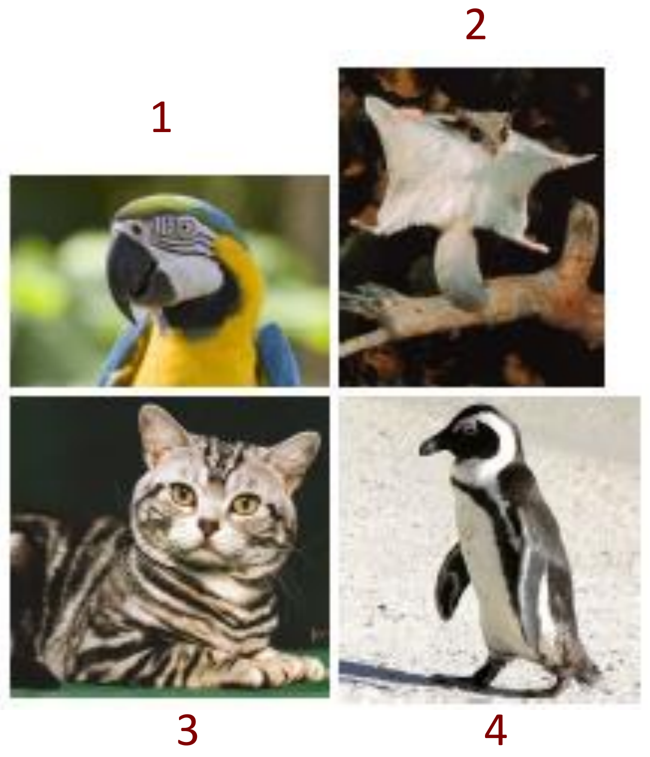
Inductive bias

- Most probably you guessed either
  - ABBA ($p = 0.6 - 0.7$) or
  - AABB ($p = 0.3 - 0.4$)

- The first choice relates to the distinction: “bird” vs. “non-bird” and the second to the distinction “fly” “no-fly”

- The preference for one distinction over the other is a bias inherent to the learner and in the context of machine learning it is called inductive bias
  - in the absence of data that narrow down the relevant concept, what type of solutions are we more likely to prefer?

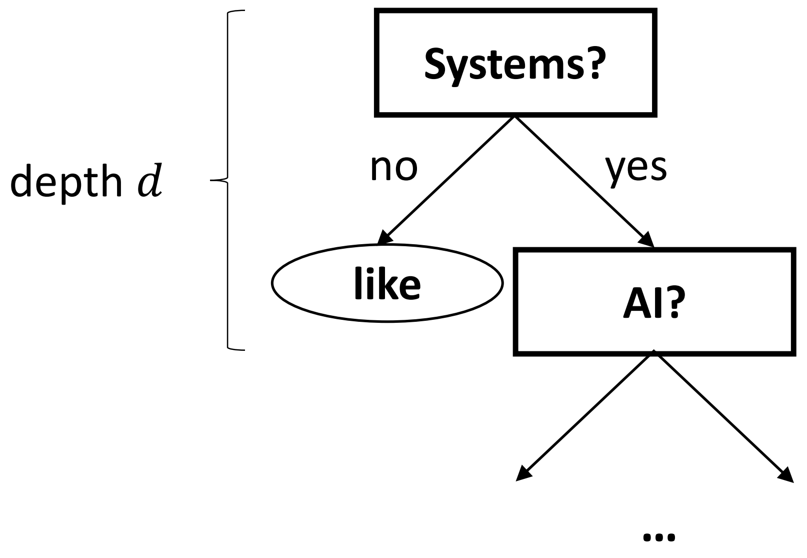
Test data



Hal Daume III

Inductive bias

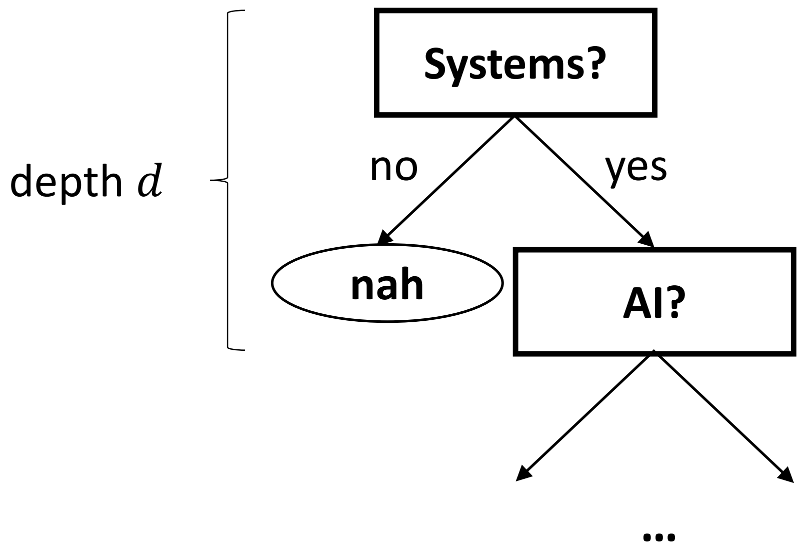
- Now consider a certain family of DT models that expand up to a certain depth level.
  - These DTs are called shallow DTs
- What is the inductive bias of shallow DTs?



Inductive bias

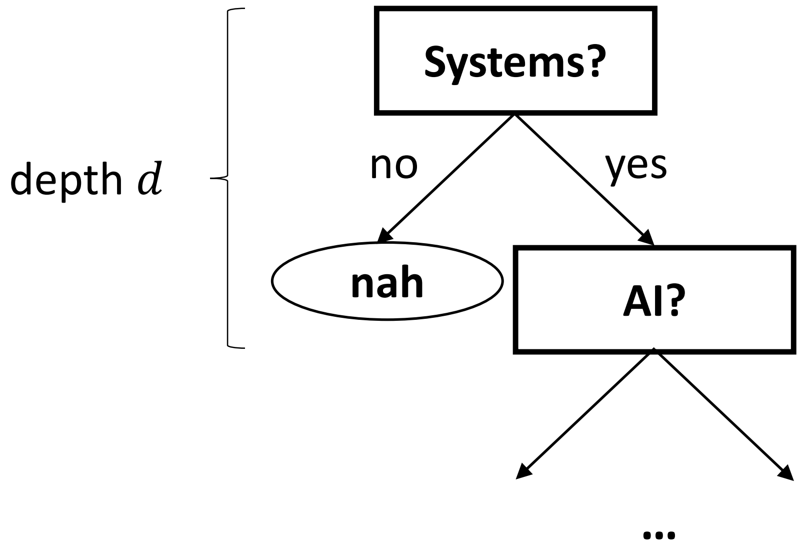
- Now consider a certain family of DT models that expand up to a certain depth level.
  - These DTs are called shallow DTs

- What is the inductive bias of shallow DTs?
  - it is the fact that they are allowed to ask a limited number of questions (small number of features)



Inductive bias

- Now consider a certain family of DT models that expand up to a certain depth level.
    - These DTs are called shallow DTs
- What is the inductive bias of shallow DTs?
    - it is the fact that they are allowed to ask a limited number of questions (small number of features)
- A shallow DT would be ideal for modeling a function “students only like AI courses”. Only one feature is needed (Is AI course?)
- A shallow DT would be bad at modeling a function “If a student liked an odd number of courses so far he will definitely like the next one”



Why learning might fail!

- Inductive bias

- Noise in the training data
  - feature level
  - label level

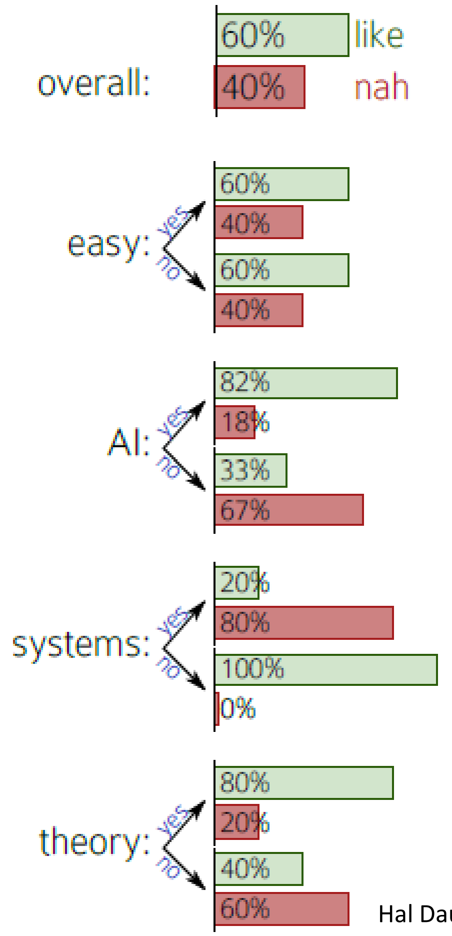
- Insufficient feature vectors

- Some examples may not have a single correct answer

Note: inductive bias source of error is fundamentally different than the other three sources of error. Inductive bias is inherent to the model. Maybe if you switch to a different model you will solve the problem.

Underfitting & Overfitting

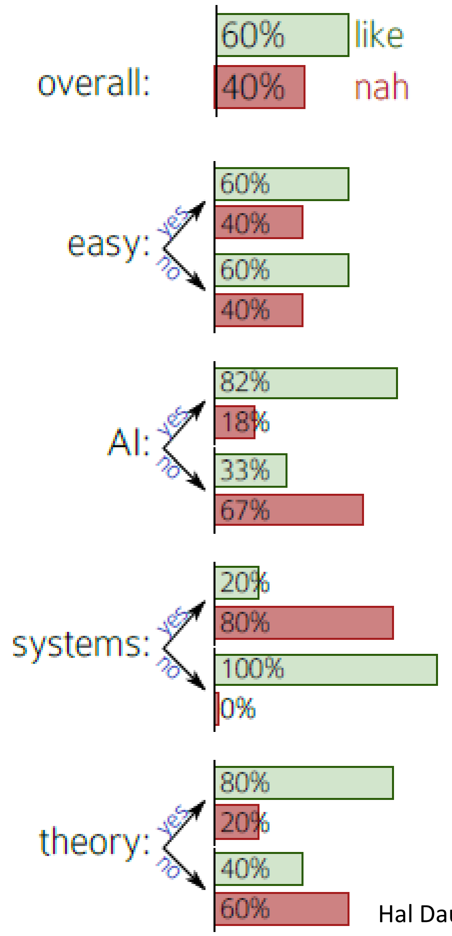
- One extreme case: the DT model is totally empty! no questions asked! What would be the decision rule?



Hal Daume III

Underfitting & Overfitting

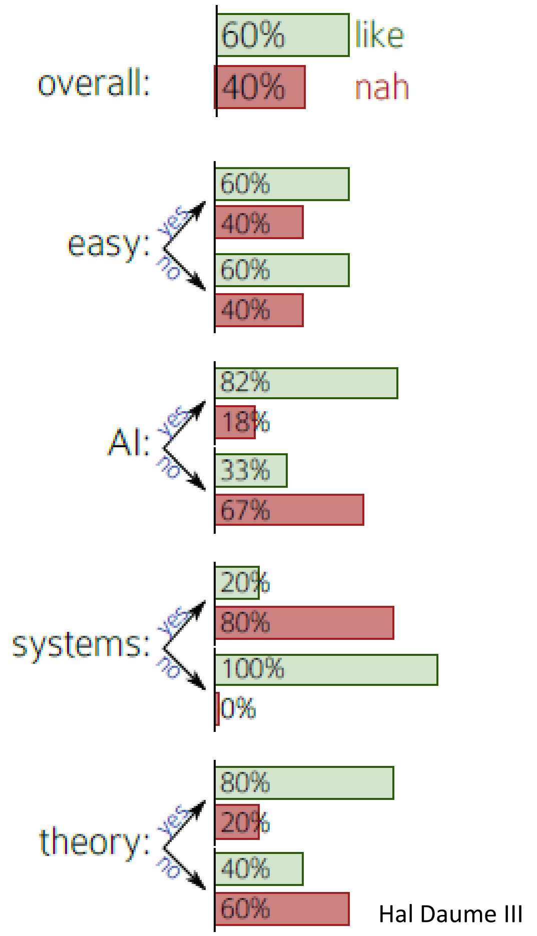
- One extreme case: the DT model is totally empty! no questions asked! What would be the decision rule?
    - based on overall histogram: for every input give the output "like"
- Another extreme case: the DT model is full. Every possible question is asked.
    - there may be leaves with no associated training data
    - but also leaves for every training sample
- What is the training error in the first case?
- What is the training error in the second case?



Hal Daume III

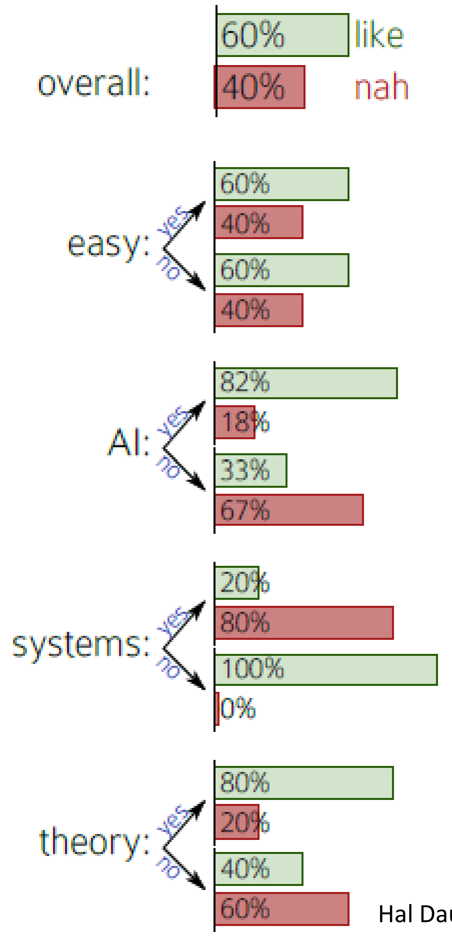
Underfitting & Overfitting

- One extreme case: the DT model is totally empty! no questions asked! What would be the decision rule?
  - based on overall histogram: for every input give the output “like”
- Another extreme case: the DT model is full. Every possible question is asked.
  - there may be leaves with no associated training data
  - but also leaves for every training sample
- What is the training error in the first case?
  - 8/20 -> 40%
- What is the training error in the second case?



Underfitting & Overfitting

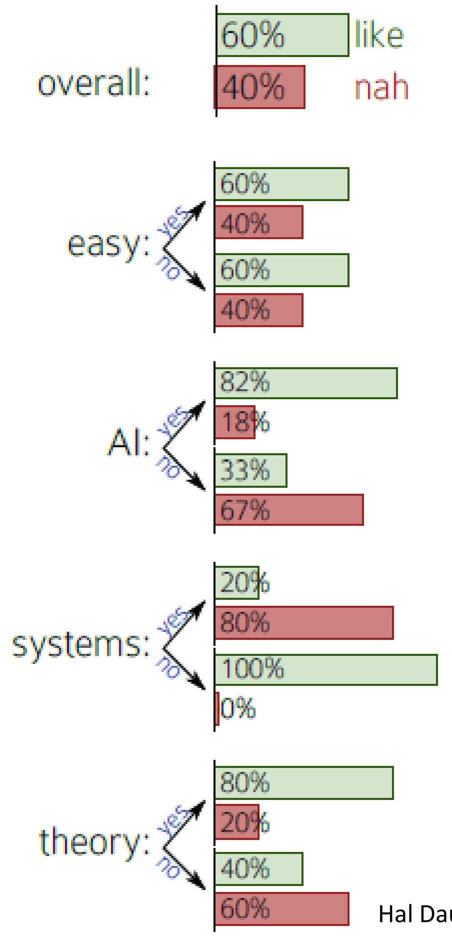
- One extreme case: the DT model is totally empty! no questions asked! What would be the decision rule?
  - based on overall histogram: for every input give the output “like”
- Another extreme case: the DT model is full. Every possible question is asked.
  - there may be leaves with no associated training data
  - but also leaves for every training sample
- What is the training error in the first case?
  - 8/20 -> 40%
- What is the training error in the second case?
  - 0%



Hal Daume III

Underfitting & Overfitting

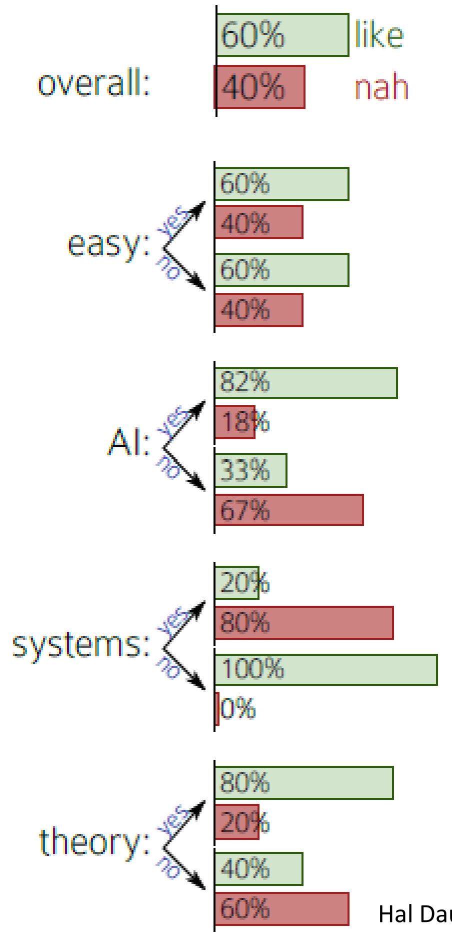
- One extreme case: the DT model is totally empty! no questions asked! What would be the decision rule?
    - based on overall histogram: for every input give the output "like"
- Another extreme case: the DT model is full. Every possible question is asked.
    - there may be leaves with no associated training data
    - but also leaves for every training sample
- What is the expected error in unseen data for the first case?
- What is the expected error for unseen data in the second case?



Hal Daume III

Underfitting & Overfitting

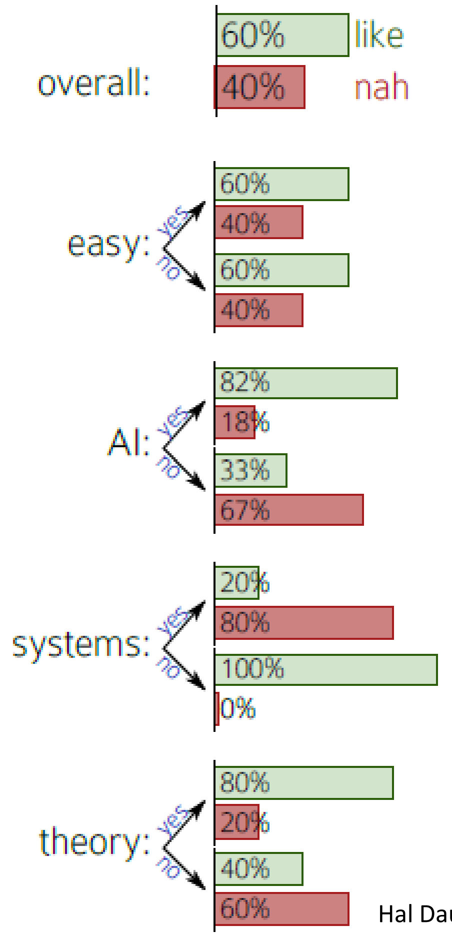
- One extreme case: the DT model is totally empty! no questions asked! What would be the decision rule?
  - based on overall histogram: for every input give the output “like”
- Another extreme case: the DT model is full. Every possible question is asked.
  - there may be leaves with no associated training data
  - but also leaves for every training sample
- What is the expected error in unseen data for the first case?
  - we might expect the same error: 40%
- What is the expected error for unseen data in the second case?



Hal Daume III

Underfitting & Overfitting

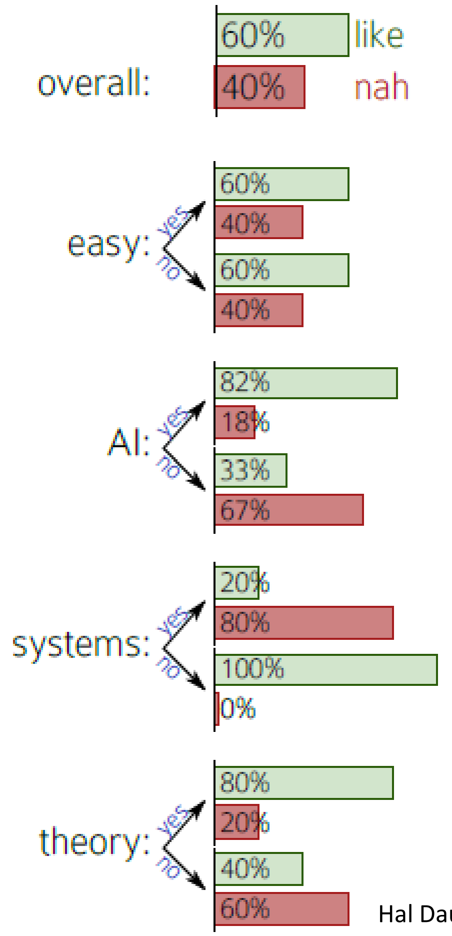
- One extreme case: the DT model is totally empty! no questions asked! What would be the decision rule?
  - based on overall histogram: for every input give the output "like"
- Another extreme case: the DT model is full. Every possible question is asked.
  - there may be leaves with no associated training data
  - but also leaves for every training sample
- What is the expected error in unseen data for the first case?
  - we might expect the same error: 40%
- What is the expected error for unseen data in the second case?
  - ~40%. If assume that 2/10 test samples match training samples then the error is ~40%, same as the empty DT



Hal Daume III

Underfitting & Overfitting

- This example illustrates the key concepts of **underfitting** and **overfitting**
- When you have the opportunity to learn and you do not (underfitting)
  - even one more feature would lead to huge advancements in the performance of the DT model
  - the model is too simple
- When you pay too much attention to the peculiarities of the training data and are not able to generalize (overfitting)
  - the model is too complicated



Hal Daume III

Hyperparameter tuning

- In every ML problem one chooses a method (e.g., a DT) to model its data.
- Every model comes with certain number of associated parameters.
  - for DT: the number of questions, the order of questions, the decision at leaves are all parameters of the model
  - The goal is to data-drive the selection of these parameters
- Some models come with additional parameters that control other parameters of the model. For instance, the depth d of a DT is such a hyperparameter.
- The adjustment of the hyperparameters can also affect the inductive bias of the model!
- How should we tune those parameters to avoid overfitting (e.g., large d) and underfitting (e.g., small d)?

Hyperparameter tuning

- We have already mentioned training and test data.

- One idea would be to tune those parameters based on the performance of the algorithm on test data. Would that be an option?

Hyperparameter tuning

- We have already mentioned training and test data.

- One idea would be to tune those parameters based on the performance of the algorithm on test data. Would that be an option?

!!! NEVER !!!

Hyperparameter tuning

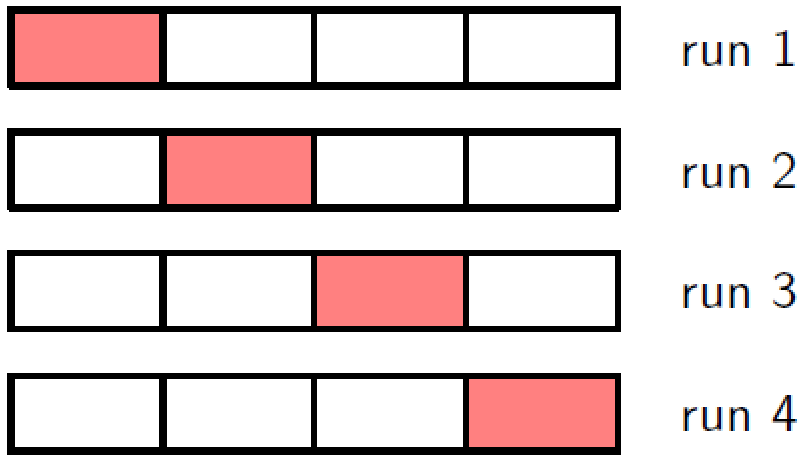
- Instead:
  - split your data into: training – validation – test data
  - For each parameter setting under consideration
    - train the model on training data
    - evaluate the model on validation data
  - For the above set of models choose the one that minimizes the loss function on validation data
  - Evaluate the chosen model on test data

Note: you can re-train the chosen model (with the particular hyperparameter set) on the training + validation dataset.

Hyperparameter tuning

- Instead:
  - split your data into: training – validation – test data
  - For each parameter setting under consideration
    - train the model on training data
    - evaluate the model on validation data
  - For the above set of models choose the one that minimizes the loss function on validation data
  - Evaluate the chosen model on test data

4-fold cross-validation



Bishop

Note: you can re-train the chosen model (with the particular hyperparameter set) on the training + validation dataset.

Note: instead of single fold validation you better follow a k-fold cross validation

Feature pruning

- Carefully selecting and curating your features could be of paramount importance for the whole ML project!
- How should I decide if I can discard a feature or keep it?
- Binary case: discard features that either never happen (always zero) or happen almost every time (almost always 1)
- Real case:

Feature pruning

- Carefully selecting and curating your features could be of paramount importance for the whole ML project!
- How should I decide if I can discard a feature or keep it?
- Binary case: discard features that either never happen (always zero) or happen almost every time (almost always 1)
- Real case: features with very low variance
- **Careful: how often a binary feature happens or what is the ideal cutoff variance of a real valued feature may be hyperparameters of the whole approach!**
- Great variety of state of the art methods that deal with feature selection

Feature normalization

- There are two basic types of normalization
  - feature normalization: adjust across all samples of particular feature
  - sample normalization: adjust only specific sample
- Common feature normalization schemes with $n = 1, \dots, N$ samples of L −dimension feature vector:
  - std scaling:
    - centering: $\hat{x}_{n,l} \leftarrow x_{n,l} - \mu_l, l = 1, \dots, L$
    - scaling: $\hat{x}_{n,l} \leftarrow \hat{x}_{n,l} / \sigma_l$
    - μ_l and σ_l are the mean and standard deviation of the lth feature
  - Absolute scaling:
    - scaling: $x_{n,l} \leftarrow x_{n,l} / r_l$
    - where $r_l = \max_l |x_{n,l}|$
  - Min-Max scaling
    - scaling: $x_{n,l} \leftarrow x_{n,l} - q_l / r_l - q_l$
    - where $r_l = \max_l |x_{n,l}|$ and $q_l = \min_l |x_{n,l}|$
- Common sample normalization scheme with N samples
  - scaling: $x_n \leftarrow x_n / \|x_n\|$

Evaluating model performance

- So far we have talked only about *accuracy*

- For binary classifiers there are many performance measures that you can choose based on the problem (e.g., a medical problem)

- There are two major types of binary classification problems
  - X versus Y
  - X versus not-X (detector: detecting and event rather than finding a true distinction between X and Y)

- For detection there are often more appropriate success metrics than accuracy
  - e.g., precision / recall measure

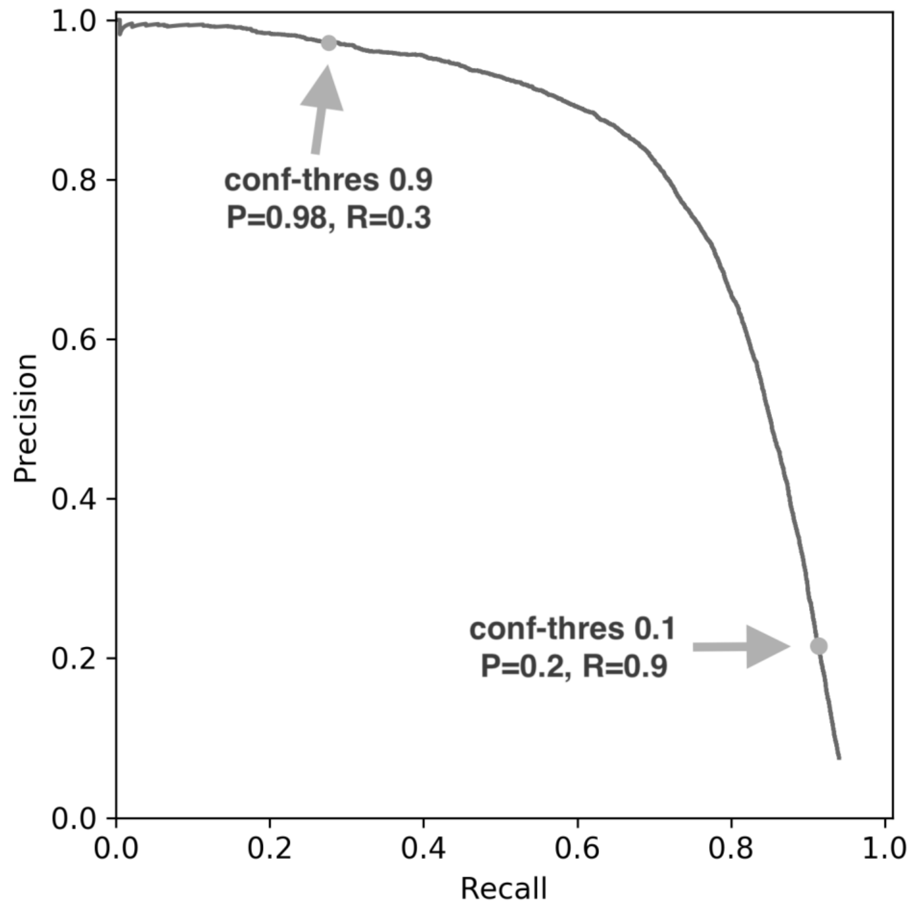
Precision / Recall

- $precision\ = \frac{TP}{TP+FP}$ or how many of the Xs that you found are actually Xs?
- $recall\ = \frac{TP}{TP+FN}$ or how many of the Xs that are out there did you find?
- A good mnemonic example:

$$\textit{Do you promise to } \underbrace{\textit{tell the whole truth}}_{\text{high recall}} \textit{ and } \underbrace{\textit{nothing but the truth}}_{\text{high precision}}\textit{?}$$

Precision / recall curves

- Suppose that you want to detect spam emails and apart of the final decision of the model that you train for that you also get a confidence score.
- If you sort the confidence scores for all your samples then you can choose how aggressively you are going to cut out emails by choosing the appropriate threshold for confidence score.
- by changing the threshold you can trace out a precision/recall curve. You can compare A and B classifiers by comparing respective curves. A dominates B if its curve is always higher than B.



F score

- Sometimes a single performance measure (instead of two) is more convenient.

- In order to combine the precision and recall measures we use their harmonic mean:

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

- If you think that precision is more important than recall then you can use the *weighted* F-score:

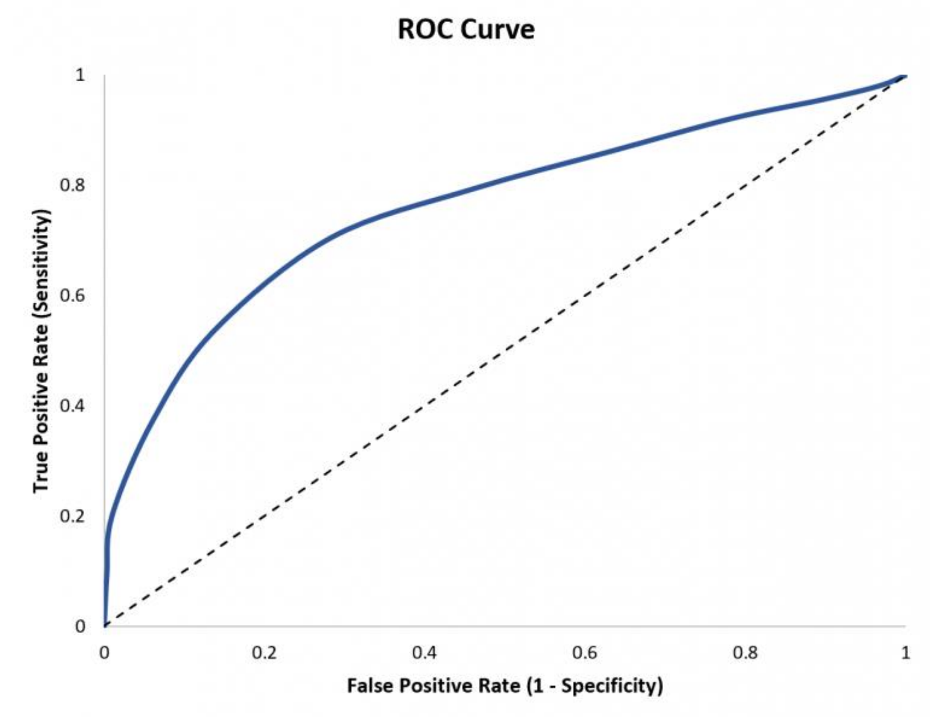
$$F = \frac{(1 + \beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$$

for $\beta = 1$ it reduces to the standard F score which is also called $F1$ score.

- **Note:** precision and recall (and hence f-score) depend crucially on which class is considered the thing you wish to find!!!

Sensitivity and Specificity

- sensitivity is exactly the same as recall.
- Specificity is about *not* finding the things that one doesn't want to find. Specificity is precision on the negation of the task at hand.
- You can compute curves for sensitivity and specificity much like those for precision and recall.
- The typical plot, referred to as the **receiver operating characteristic** (or **ROC curve**) plots the sensitivity against 1 – specificity



- Given an ROC curve, you can compute the **area under the curve** (or **AUC**) metric, which also provides a meaningful single number for a system's performance

ARISTOTLE UNIVERSITY OF THESSALONIKI

FACULTY OF ENGINEERING

Questions?

Pattern Recognition & Machine Learning

Our First Model

Fall Semester