Natural Language Processing
自然語言處理

黃瀚萱
Department of Computer Science
National Chengchi University
2020 Fall

Lesson 6

Text Classification

Schedule

Date	Topic
9/16	Introduction
9/23	Linguistic Essentials
9/30	Collocation
10/7	Language Model
10/14	Performance Evaluation and Word Sense Disambiguation
10/21	Text Classification (**HW1 will be assigned**)
10/28	Invited Talk: NLP and Cybersecurity (**Term Project**)
11/4	POS Tagging
11/11	Midterm Exam

Schedule

Date	Topic
11/18	Chinese Word Segmentation
11/25	Word Embeddings
12/2	Neural Networks for NLP
12/9	Parsing
12/16	Discourse Analysis
12/23	Invited Talk
12/30	Final Project Presentation I
1/6	Final Project Presentation II
1/13	Final Exam

Agenda

- Classification tasks
- Classifiers
  - Decision tree
  - Perceptron
  - kNN
- Feature extraction
- Overfitting
- Assignment I and Term Project

Text Classification

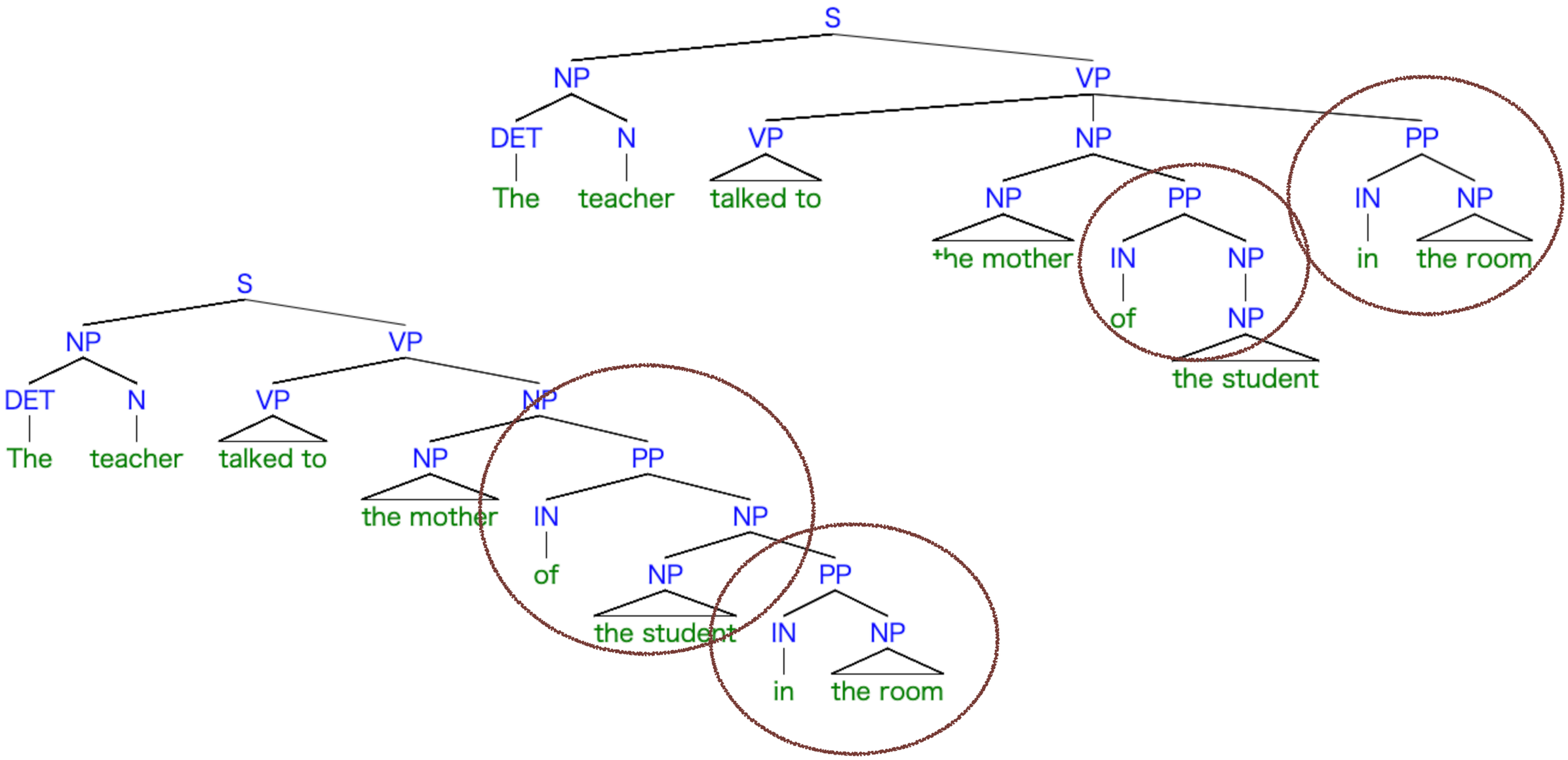
- Text classification (aka text categorization) is the task that assigning objects from a universe to two or more classes (categories).
    - Classify the sentiment polarity of a movie
    - Classify the trustiness of an article
    - Classify the category of a news article
    - Classify the part of speech tag of a word in a sentence
        - Sequence labeling

Classification Tasks

Task	Object	Categories
Disambiguation	context of a word	word's senses
PP attachment	sentence	parse trees
Author detection	document	authors
News Categorization	document	topics
Sentence boundary detection	document / paragraph	Roles of punctuation marks
Part-of-speech tagging	sentence	POS tags

Prepositional Phrase Attachment

- Linking a prepositional phrase to the correct head



Word Sense Disambiguation

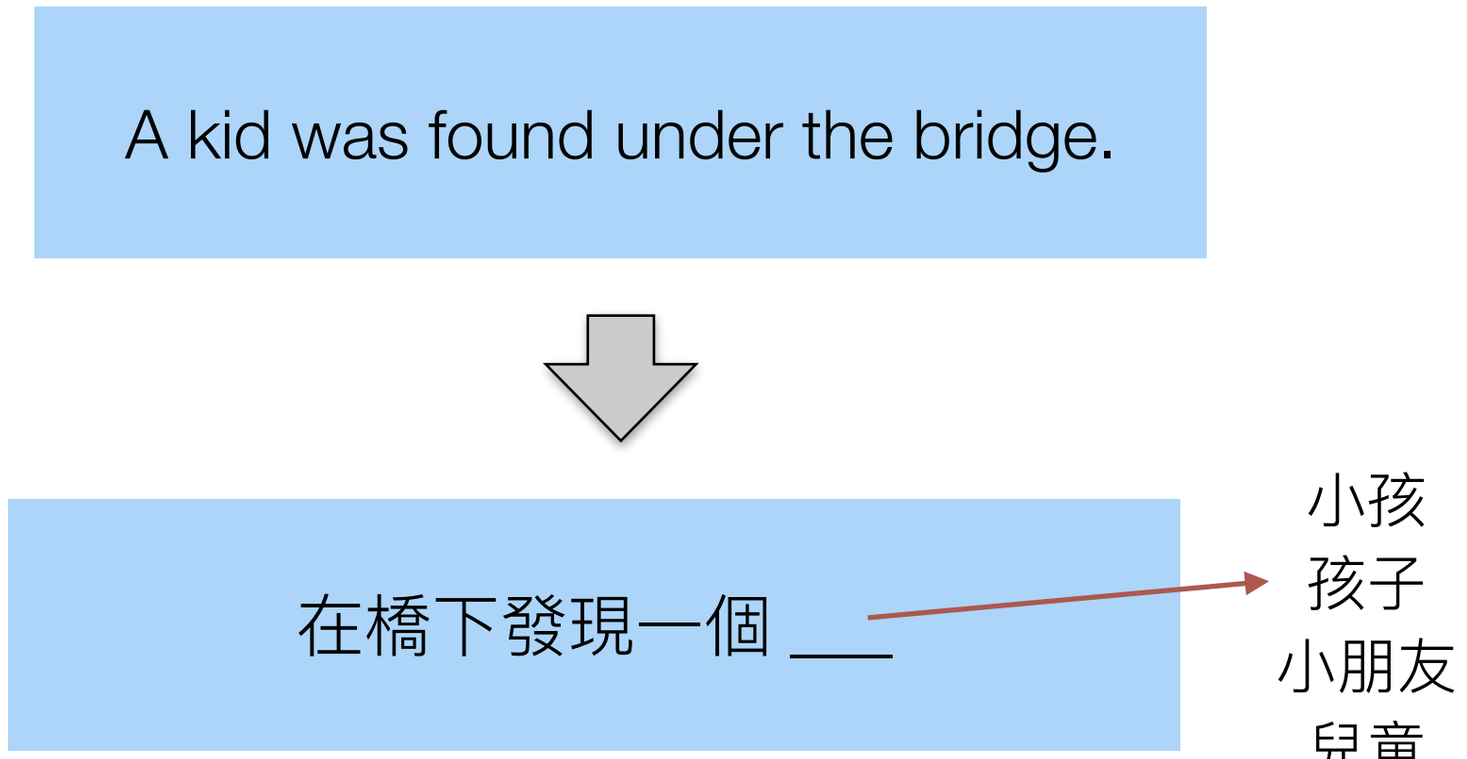
- Choose the most feasible sense from all senses of a word in a sentence.

He sat on the ***bank*** of the river and watched the currents

The rising ground bordering a lake, river, or sea

Natural Language Generation

- NLG can also be formulated as text classification
  - Predicting the next word by selecting the best one from a set of words



Training Stage

Training Data

(Arg1, Arg2)
Instances

Labels

Feature
Extraction

Feature Vectors

Learning Model

Naive Bayes

Logistic Regression

SVM

Neural Network

(Arg1, Arg2)
Instance

New Data

Bag of word

Word embedding

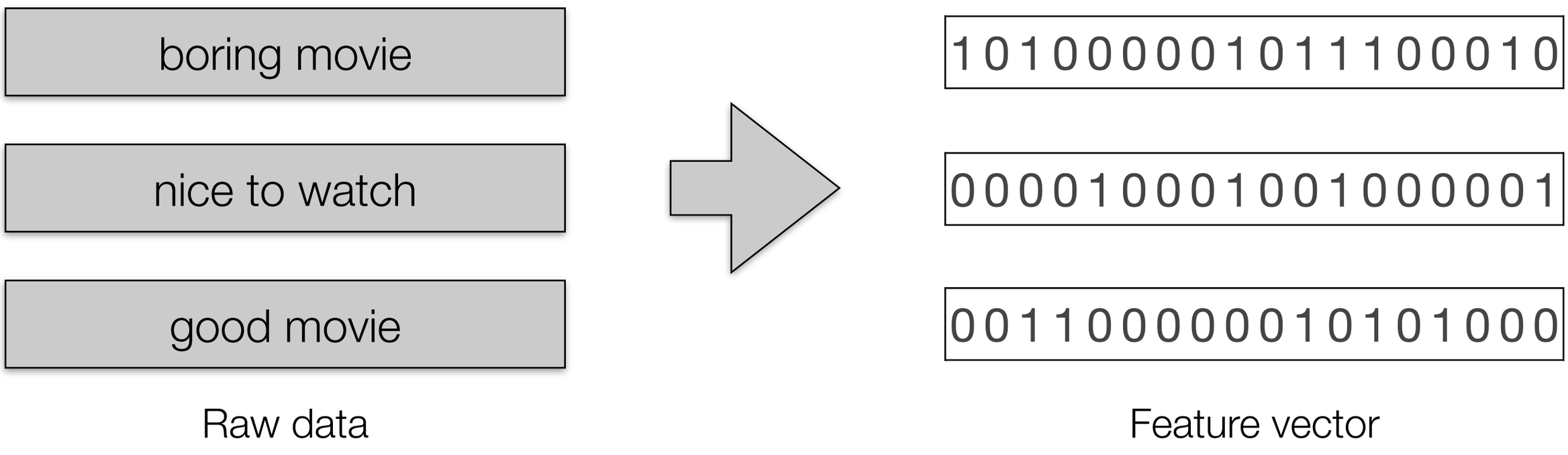
Sentence Encoder

Predicted
Label

Productive Stage

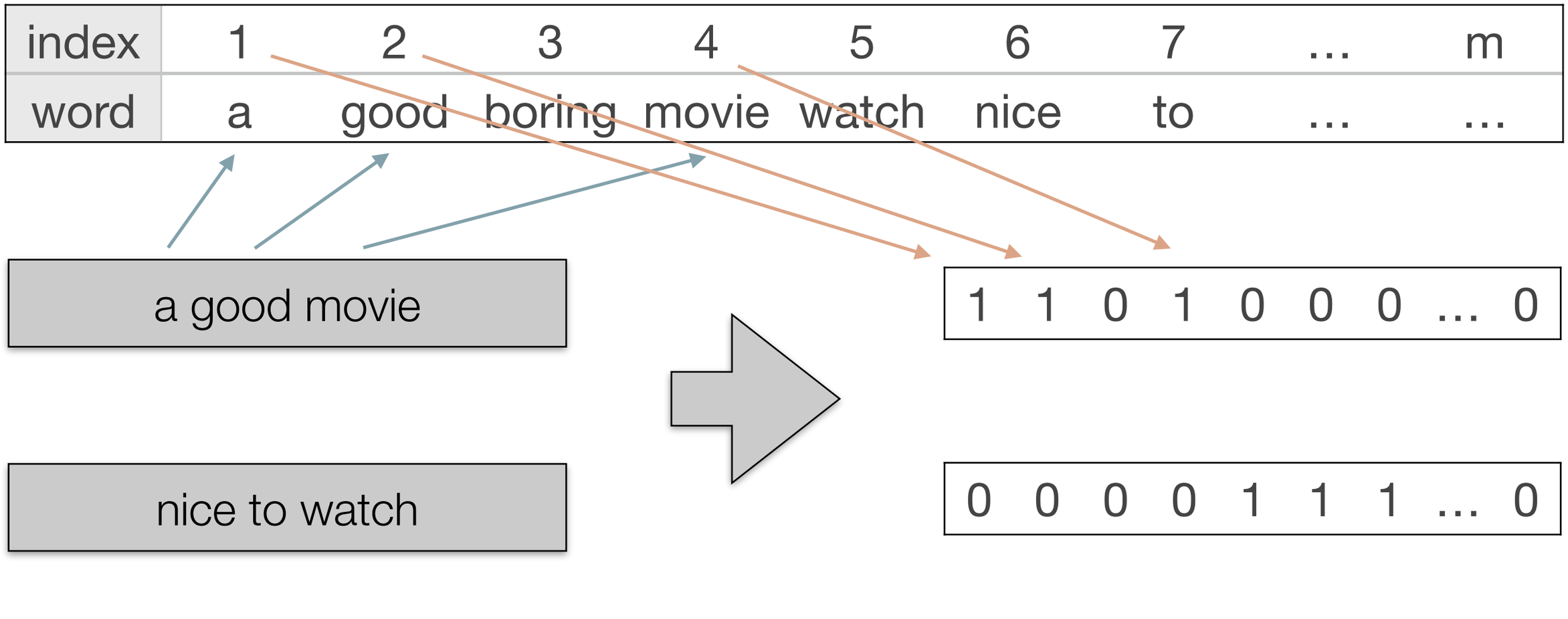
Feature Extraction

- Handcrafted linguistic features
  - Proposed by human
- Directly learning the representation from data
  - Co-optimized by data and the model



Bag of Word Representation

- Each word has a specific cell on the feature vectors
- The presence or absence of each word is denoted by the value of the cell.



Learning Models

- Ngram model
- Naive Bayes model
- Decision tree model
- Perceptrons
- Logistic regression
  - Maximum entropy model
- Neural network
- k nearest neighbors

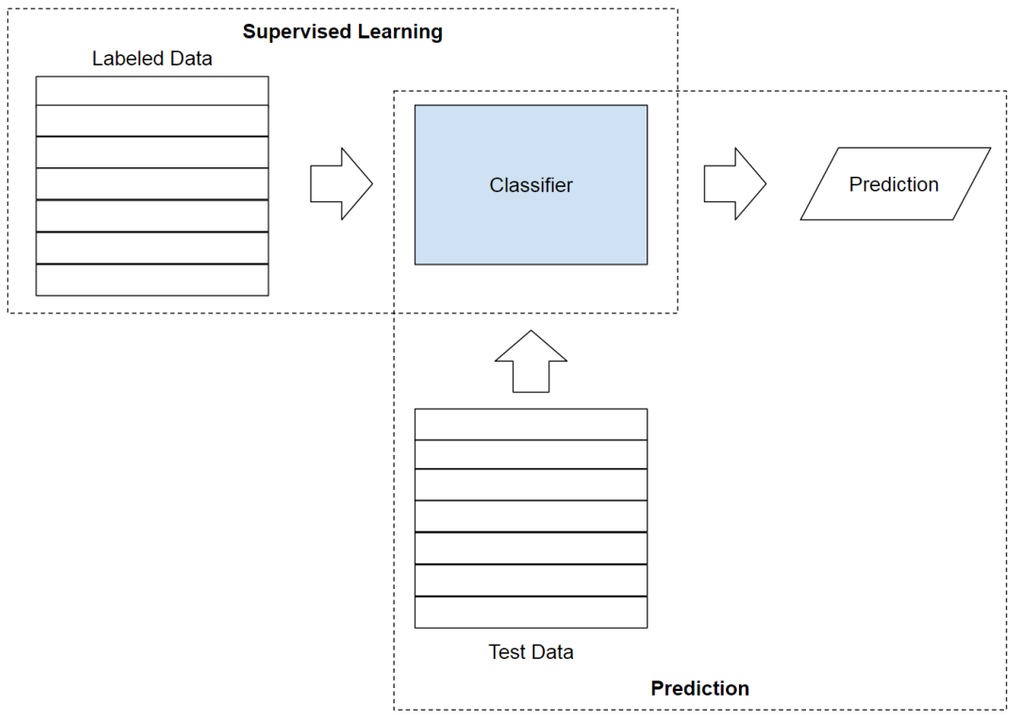
Supervised Learning

- To obtain a model *f*() that outputs y given an input *x*.

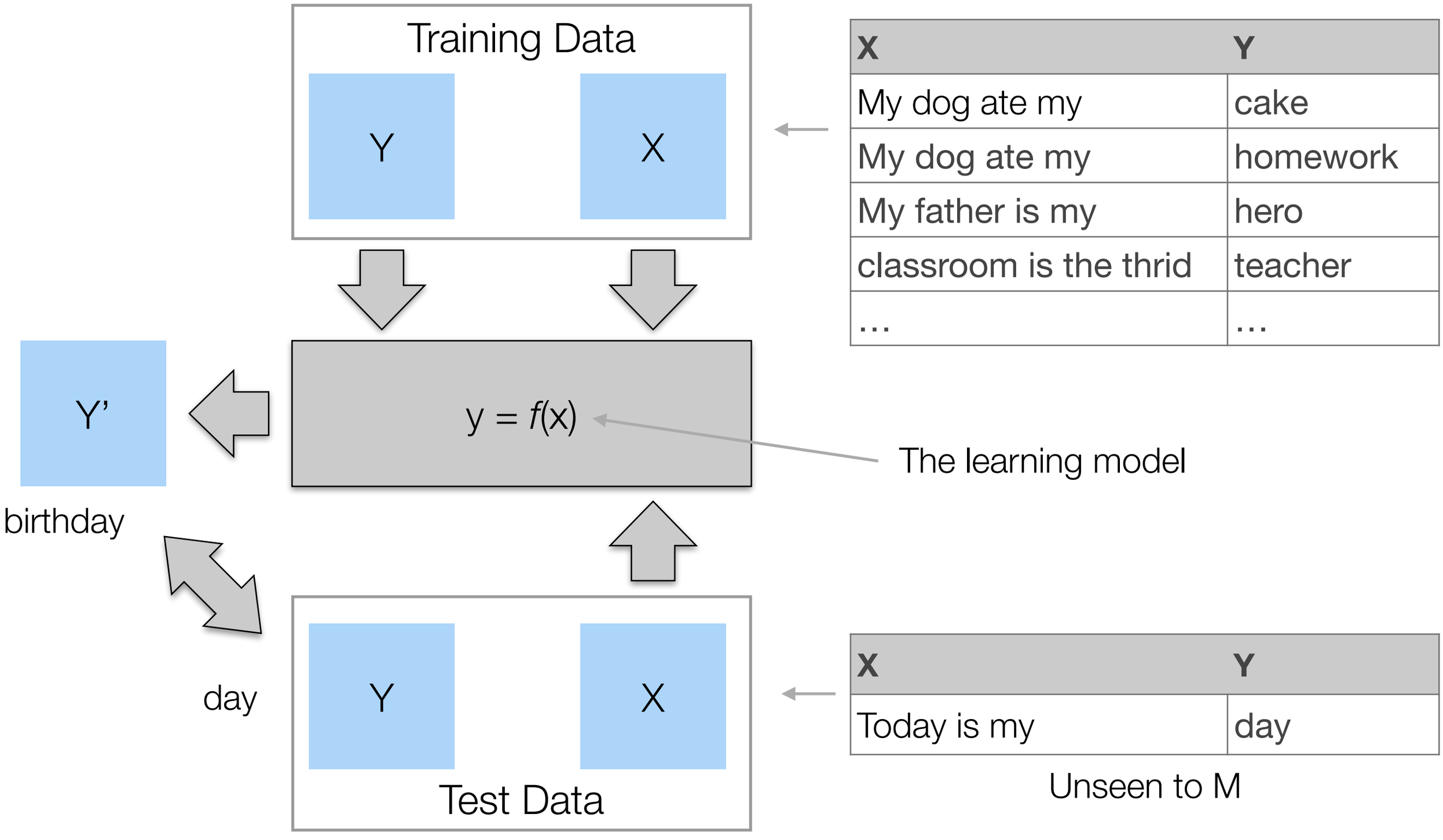
$$y = f(x)$$

- Learning from labeled data

- To capture the relation between (x, y) with a large amount of (x, y) pairs.



Supervised Learning



X	Y
My dog ate my	cake
My dog ate my	homework
My father is my	hero
classroom is the thrid	teacher
…	…

X	Y
Today is my	day

Unseen to M

Naive Bayes Model

- Simple and efficient to train
- Based on the assumption of statistically independency

$$\bar{y} = \arg\max_{y \in \mathbf{y}} P(y|x) = \arg\max_{y \in \mathbf{y}} P(y) \prod_{i=1}^{n} \frac{P(x_i|y)}{P(x_i)}$$

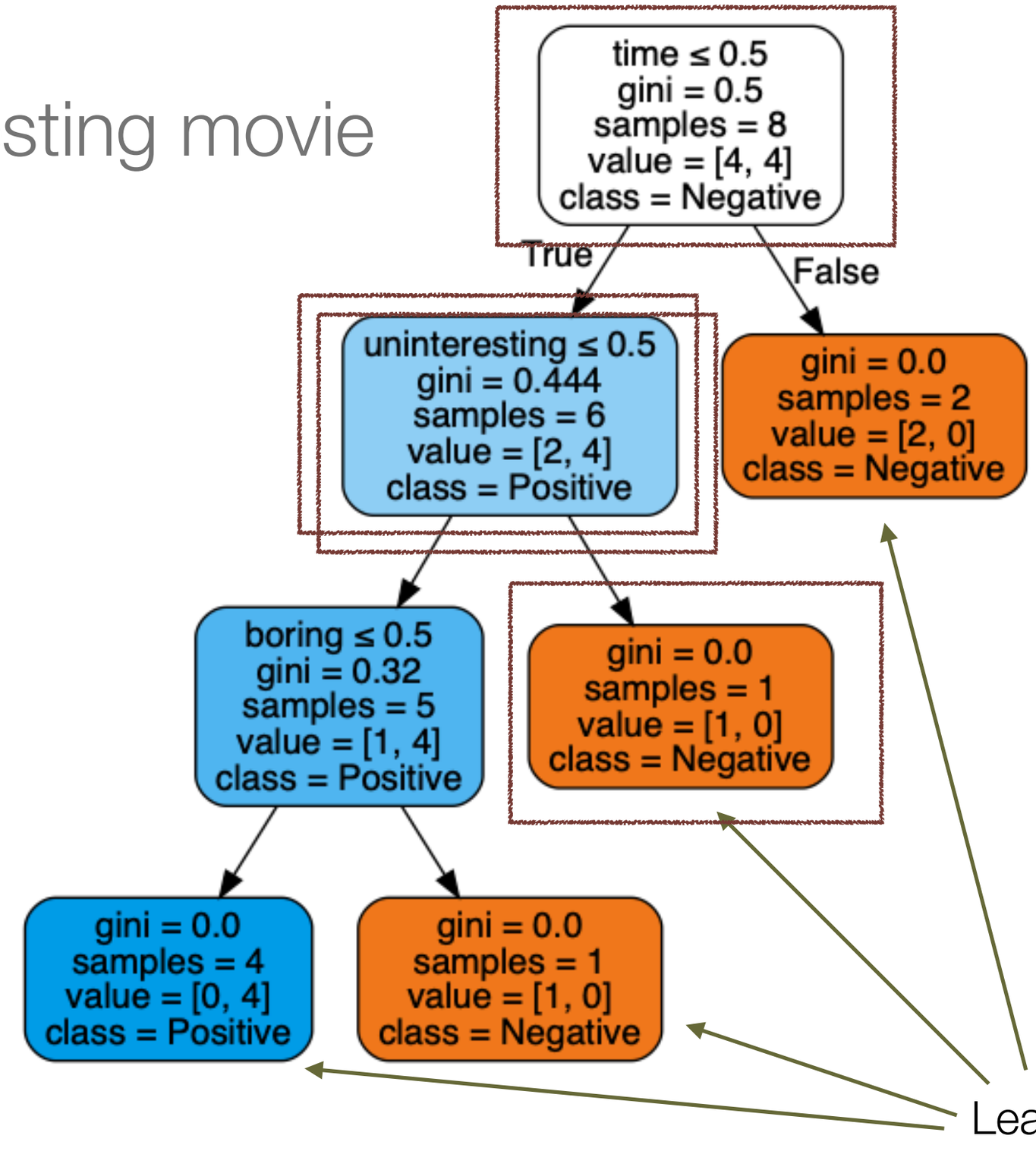
- Additive (Laplace) smoothing

$$P(x_i|y) = \frac{C(x_i, y) + k}{C(y) + |\mathbf{y}| \times k}$$

$0 \leq k \leq 1$ and $|\mathrm{y}|$ is the number of classes.

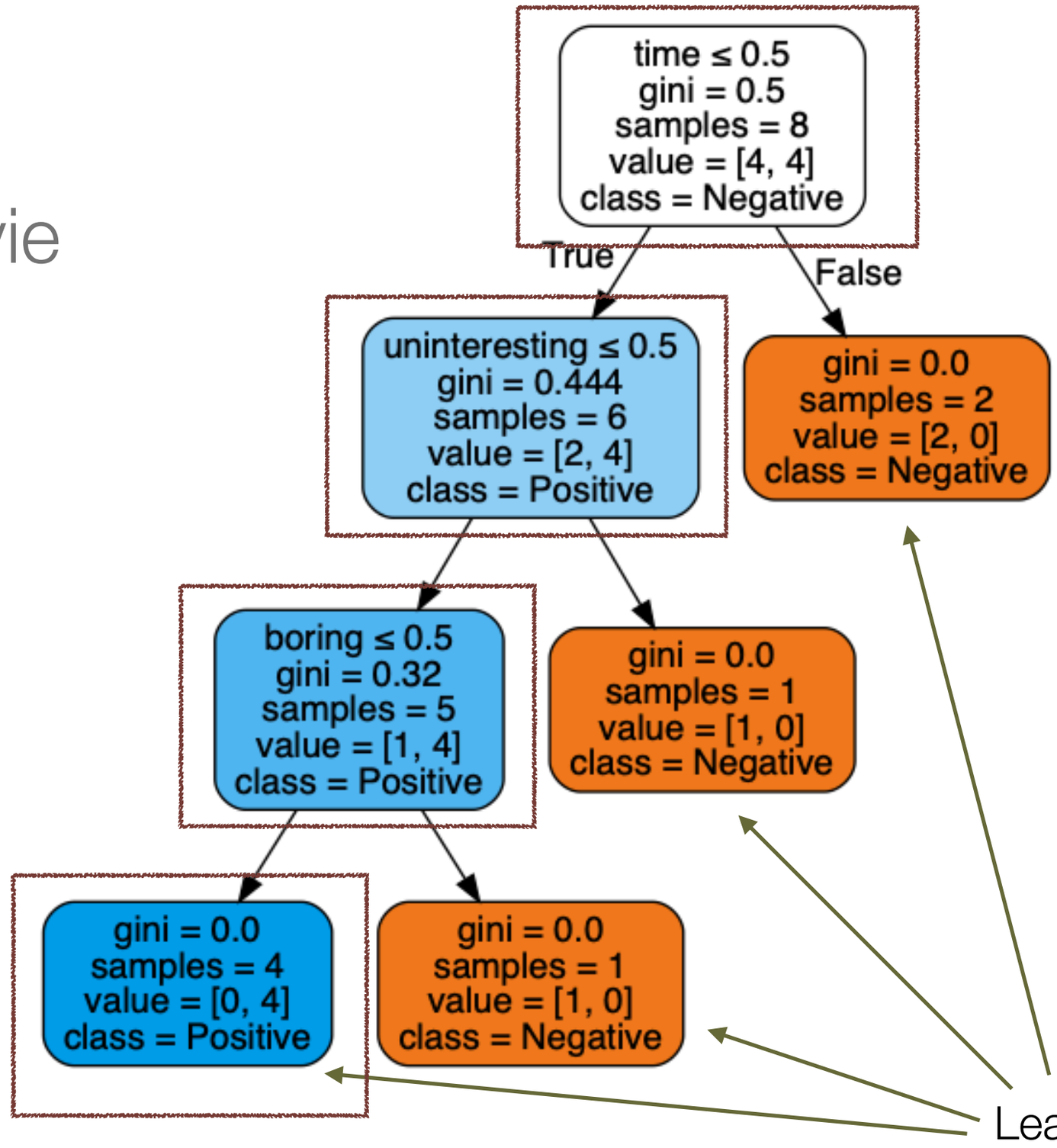
Decision Tree Classifier

Just saw an uninteresting movie



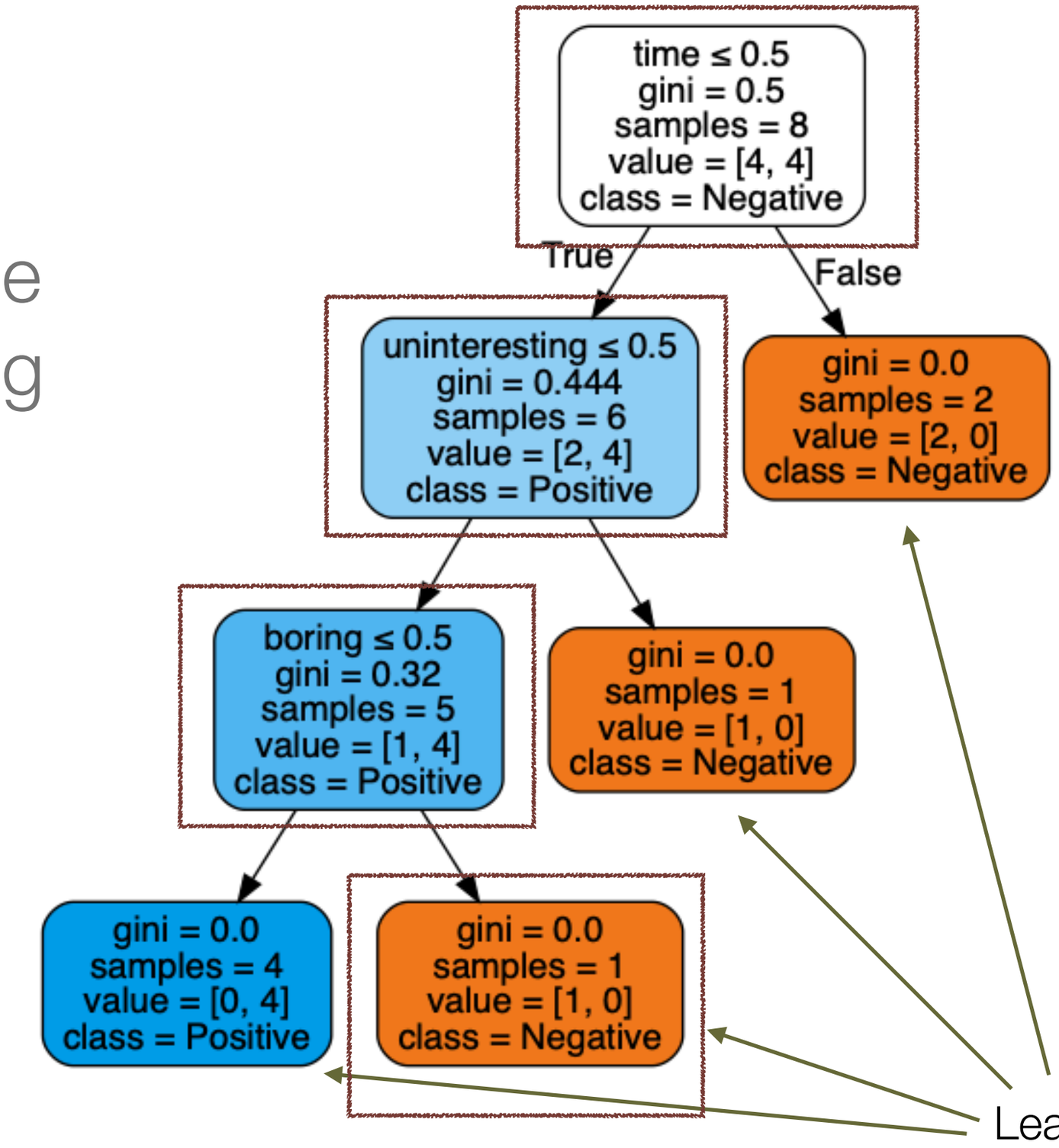
Decision Tree Classifier

It is a good movie



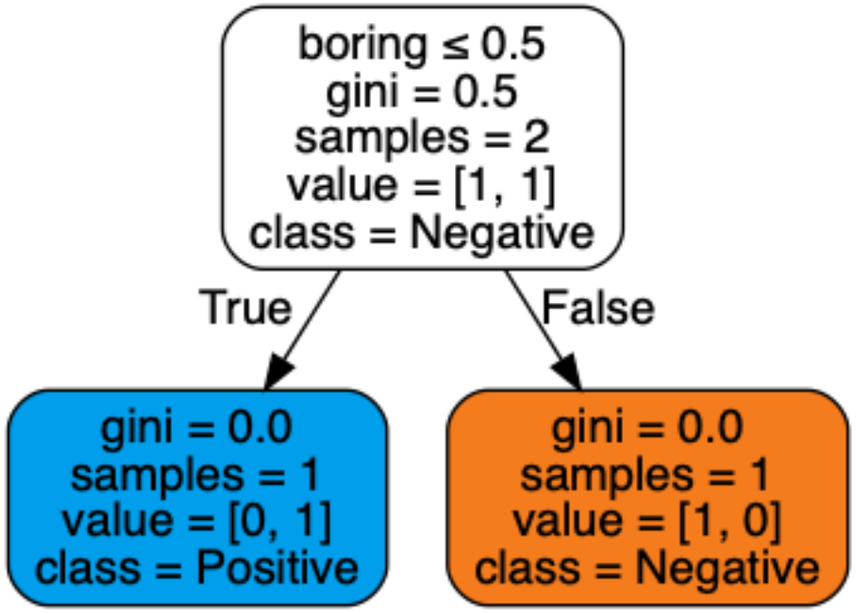
Decision Tree Classifier

It is a good movie
I didn't feel boring

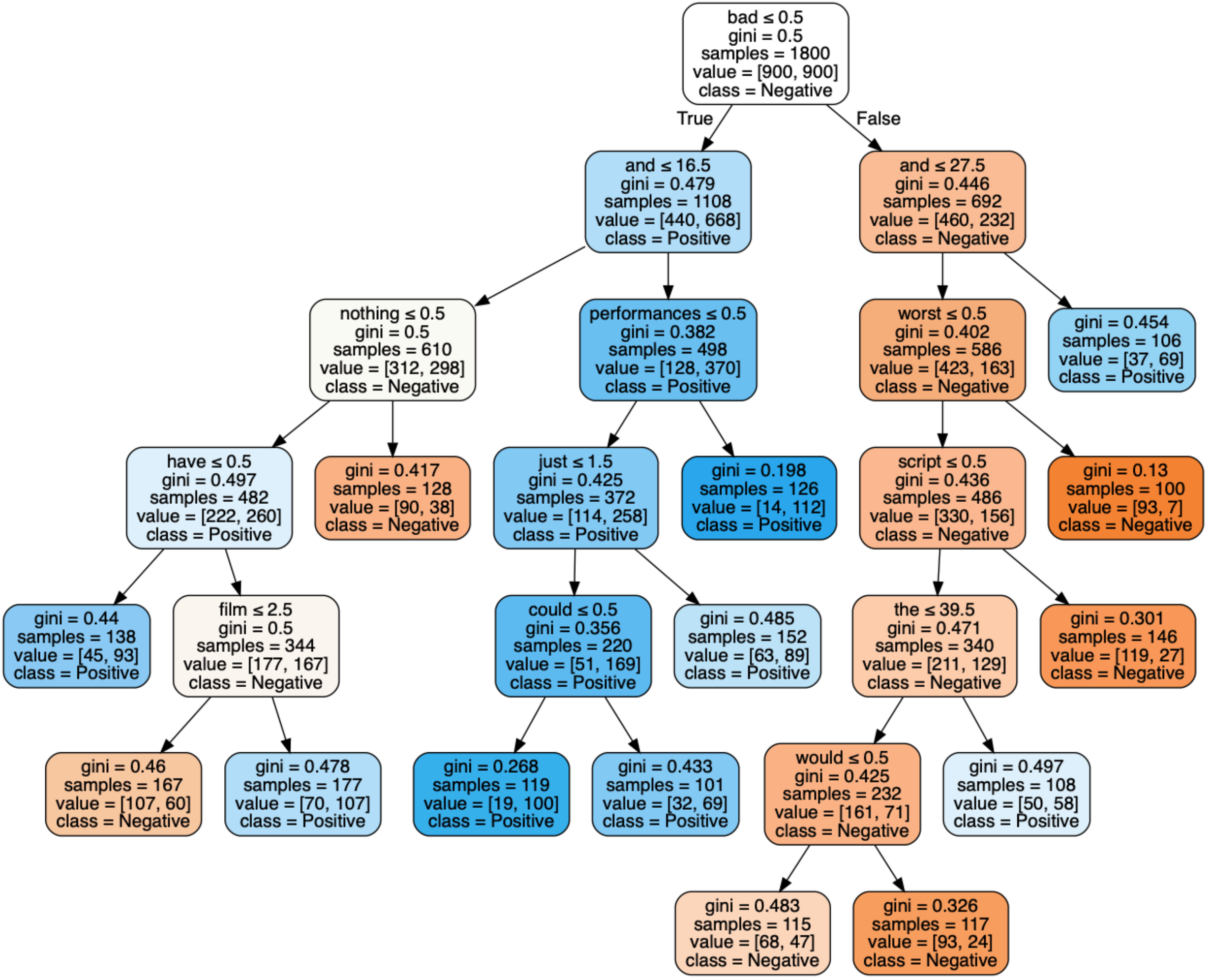


A Small Sample

- a **boring** movie => Negative
- highly recommended => Positive



One advantage of the decision tree classifier is interpretable


bad ≤ 0.5
gini = 0.5
samples = 1800
value = [900, 900]
class = Negative
True
False
and ≤ 16.5
gini = 0.479
samples = 1108
value = [440, 668]
class = Positive
and ≤ 27.5
gini = 0.446
samples = 692
value = [460, 232]
class = Negative
nothing ≤ 0.5
gini = 0.5
samples = 610
value = [312, 298]
class = Negative
performances ≤ 0.5
gini = 0.382
samples = 498
value = [128, 370]
class = Positive
worst ≤ 0.5
gini = 0.402
samples = 586
value = [423, 163]
class = Negative
gini = 0.454
samples = 106
value = [37, 69]
class = Positive
have ≤ 0.5
gini = 0.497
samples = 482
value = [222, 260]
class = Positive
gini = 0.417
samples = 128
value = [90, 38]
class = Negative
just ≤ 1.5
gini = 0.425
samples = 372
value = [114, 258]
class = Positive
gini = 0.198
samples = 126
value = [14, 112]
class = Positive
script ≤ 0.5
gini = 0.436
samples = 486
value = [330, 156]
class = Negative
gini = 0.13
samples = 100
value = [93, 7]
class = Negative
gini = 0.44
samples = 138
value = [45, 93]
class = Positive
film ≤ 2.5
gini = 0.5
samples = 344
value = [177, 167]
class = Negative
could ≤ 0.5
gini = 0.356
samples = 220
value = [51, 169]
class = Positive
gini = 0.485
samples = 152
value = [63, 89]
class = Positive
the ≤ 39.5
gini = 0.471
samples = 340
value = [211, 129]
class = Negative
gini = 0.301
samples = 146
value = [119, 27]
class = Negative
gini = 0.46
samples = 167
value = [107, 60]
class = Negative
gini = 0.478
samples = 177
value = [70, 107]
class = Positive
gini = 0.268
samples = 119
value = [19, 100]
class = Positive
gini = 0.433
samples = 101
value = [32, 69]
class = Positive
would ≤ 0.5
gini = 0.425
samples = 232
value = [161, 71]
class = Negative
gini = 0.497
samples = 108
value = [50, 58]
class = Positive
gini = 0.483
samples = 115
value = [68, 47]
class = Negative
gini = 0.326
samples = 117
value = [93, 24]
class = Negative

Pruning of Decision Tree

- The pruning is necessary because very large trees tend to *overfit* the training data.
    - Making decision based some accidental instances in the training data.
- Reduce the too fined branches in the decision tree
    - Do not split a subtree if too few leaf nodes (sparsity)
    - Limit the height of the whole tree

Properties of the Decision Tree Classifier

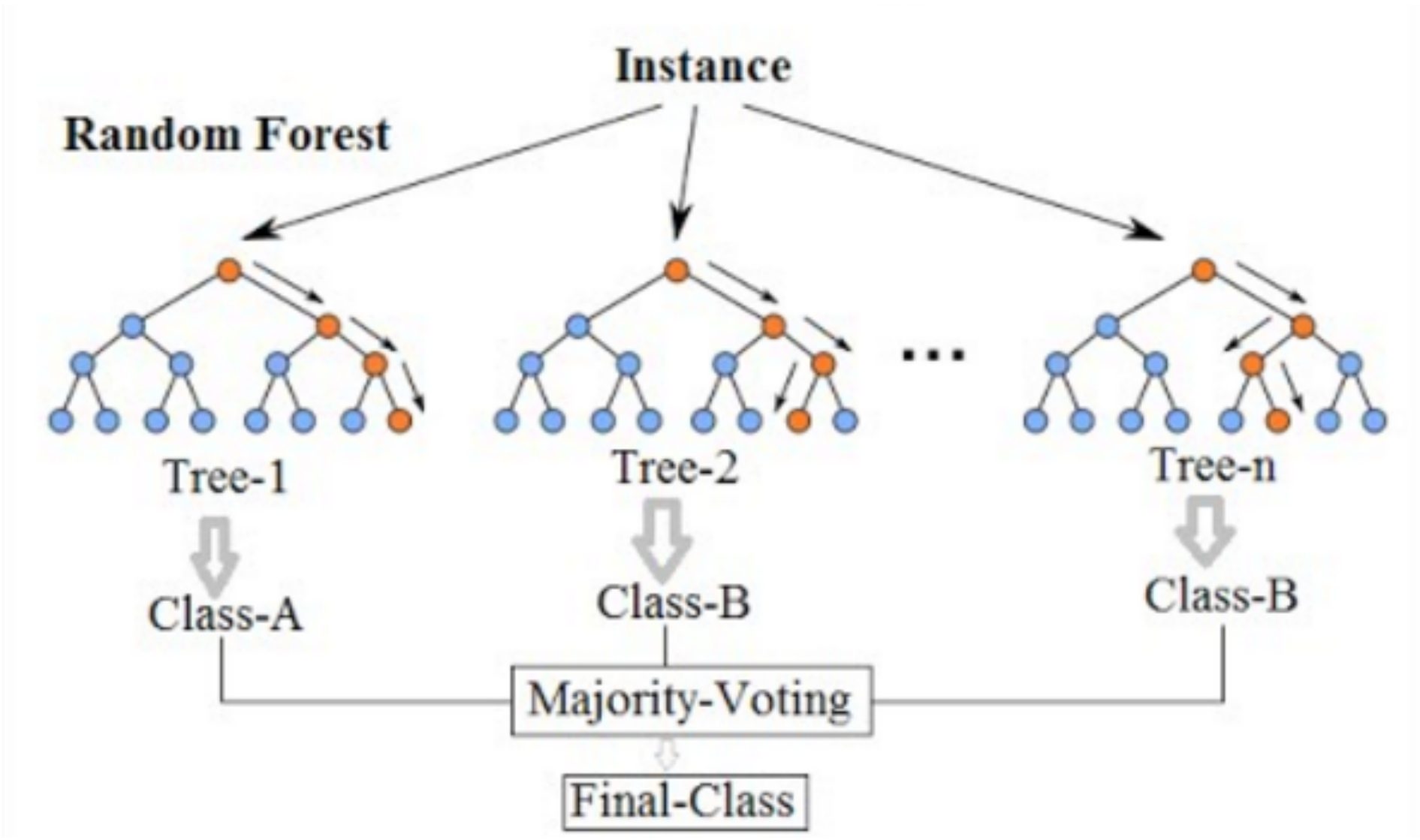
- Training is more time-consuming than naive Bayes model
- Do not rely on the assumption of statistically independency.
- Less generalized; tends to overfit the data
  - Pruning is always mandatory.

Why Decision Trees are Less Generalized

- **Overfitting**
  - Noise and lack of representative instances
  - Tends to happen in large/deep trees
- **Bias**
  - Too many restrictions on target functions
- **Variance**
  - Decision trees have high variance, which means that tiny changes in the training data have the potential to cause large changes in the final result.

Random Forest

- Majority of variations of decision trees.



Building A Variety of Decision Trees

- With randomization
  - Different samples for training each tree
  - Different feature subsets for training each tree
- Building and combining small (shallow) trees
  - Avoid overfitting
- Harder to interpret than a standalone decision tree

Gradient Tree Boosting

- Like random forests, gradient boosting is a set of decision trees
    - Unlike random forest, GB builds trees sequentially.
- Adding a weak learner to improve the existing weak learners.

$$F_{m+1}(x) = F_m(x) + h_m(x) = y$$

$$h_m(x) = y - F_m(x)$$

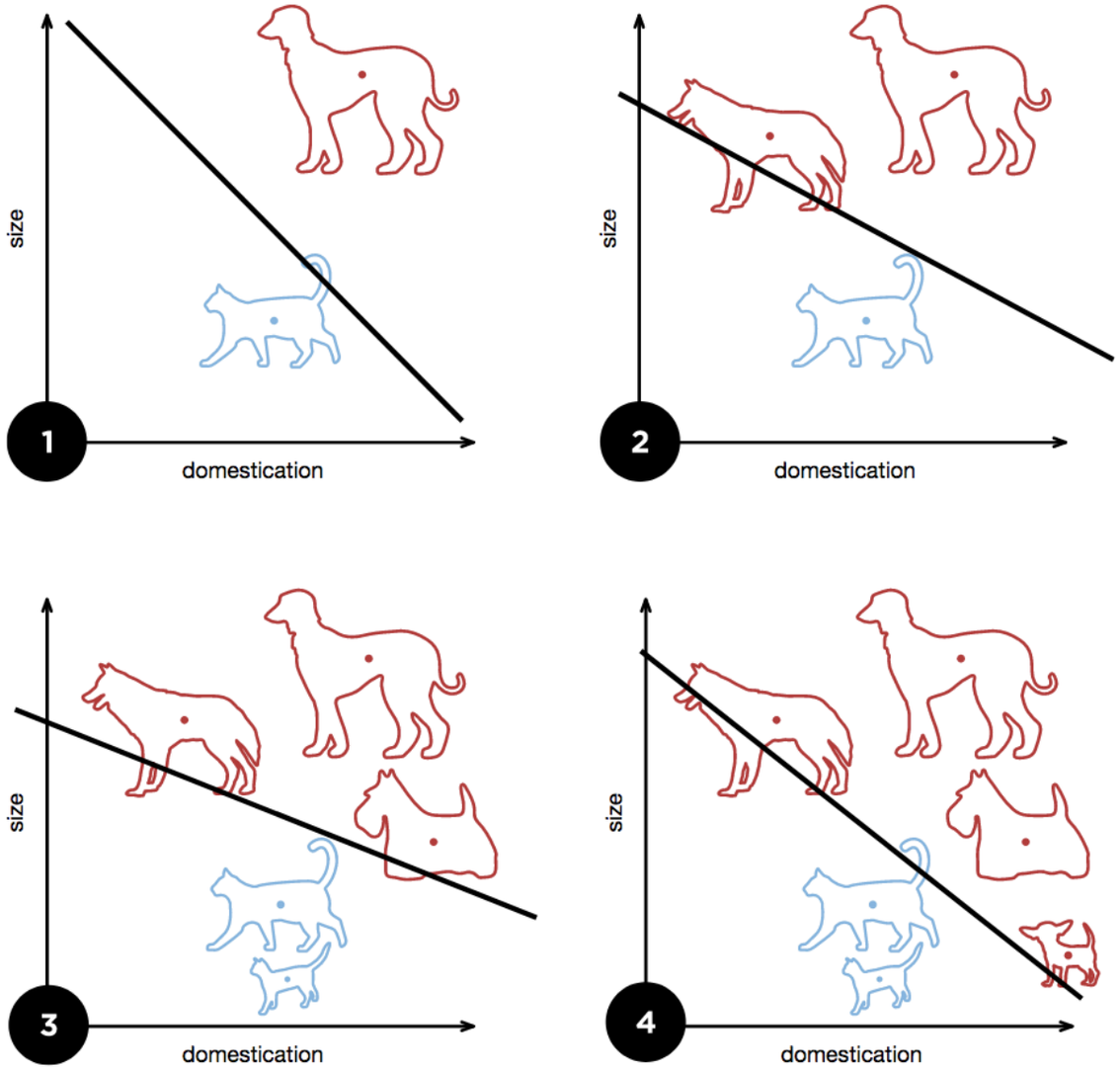
Correct the existing model by adding a new learner

Gradient Tree Boosting vs Random Forest

- Training
  - Random forest builds trees independently.
  - Gradient boosting builds trees depending on previous trees.
- Prediction
  - A RF predicts based on the majority its trees
  - GB predicts based on evaluating its trees along the order
- Performance
  - GB generally outperforms RF and DT
  - More likely to overfit to noise
- Famous implementation
  - XGBoost

Perceptron

- Features: x
- Labels: y
  - 1: Cat
  - 0: Dog



Features $(\mathbf{x})$:
x_1: size
x_2: domestication

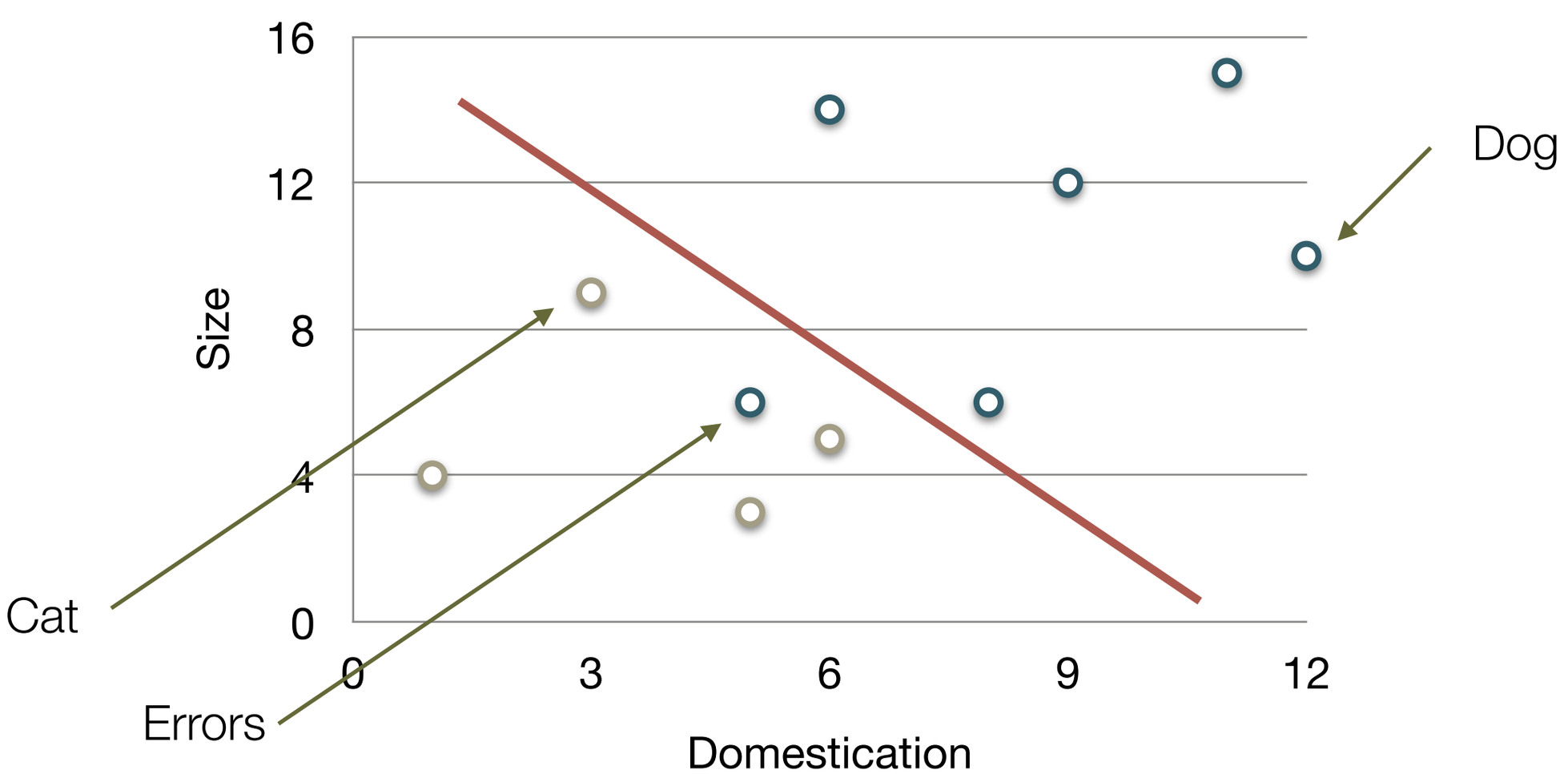
Labels (y):
0: Dog
1: Cat

$$f(x) = \begin{cases} 1 & \text{if } w_1 x_1 + w_2 x_2 + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

Parameters to estimate: w_1, w_2, and b

Finding a Hyperplane Iteratively

- A line for two-dimensional features
- A plane for three-dimensional features
- In most cases, a hyperplane (超平面) for high dimensional features



Training the Perceptron Model

$$D = \{(x_{1,1}, x_{1,2}, x_{1,3}, ..., x_{1,n}, y_1), (x_{2,1}, x_{2,2}, x_{2,3}, ..., x_{2,n}, y_2), \\ ..., (x_{m,1}, x_{m,2}, x_{m,3}, ..., x_{m,n}, y_m)\}$$

For convenient, set $x_{i,0} = 1$ for all input $1 \leq i \leq m$,
and use w_0 as the bias instead of b.

Initialization

$w_j = 0$ for all weights $0 \leq j \leq n$.

Training

Do until convergence

For each instance $(\mathbf{x}_i, y_i)$ in D

Evaluate $f(\mathbf{x}_i)$

$y'_i \leftarrow w_0 x_{i,0} + w_1 x_{i,1} + x_2 x_{i,2} + ... + w_n x_{i,n}$

Update weights

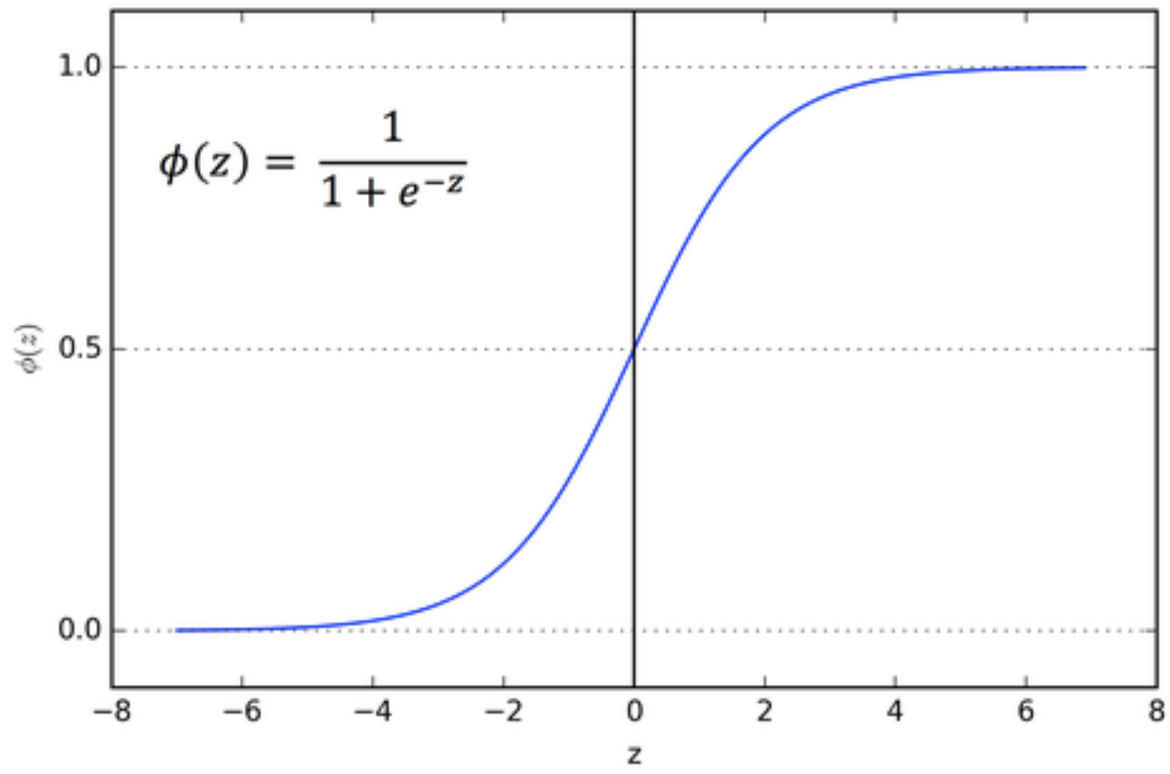
$w'_j \leftarrow w_j + \alpha(y_i - y'_i) x_{i,j}$ for all features $0 \leq j \leq n$.

Logistic Regression

- Estimate the **probability** of a binary outcome $f(y)$ based on input features x.

- Map the output of perceptron $f(x)$ to the range (0, 1) with the logistic (sigmoid) function.

$$
\begin{aligned}
& p(f(x) = 1) \\
& = \frac{1}{1 + e^{-f(\mathbf{x})}} \\
& = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n)}}
\end{aligned}
$$



Logistic Regression vs Naive Bayes Model

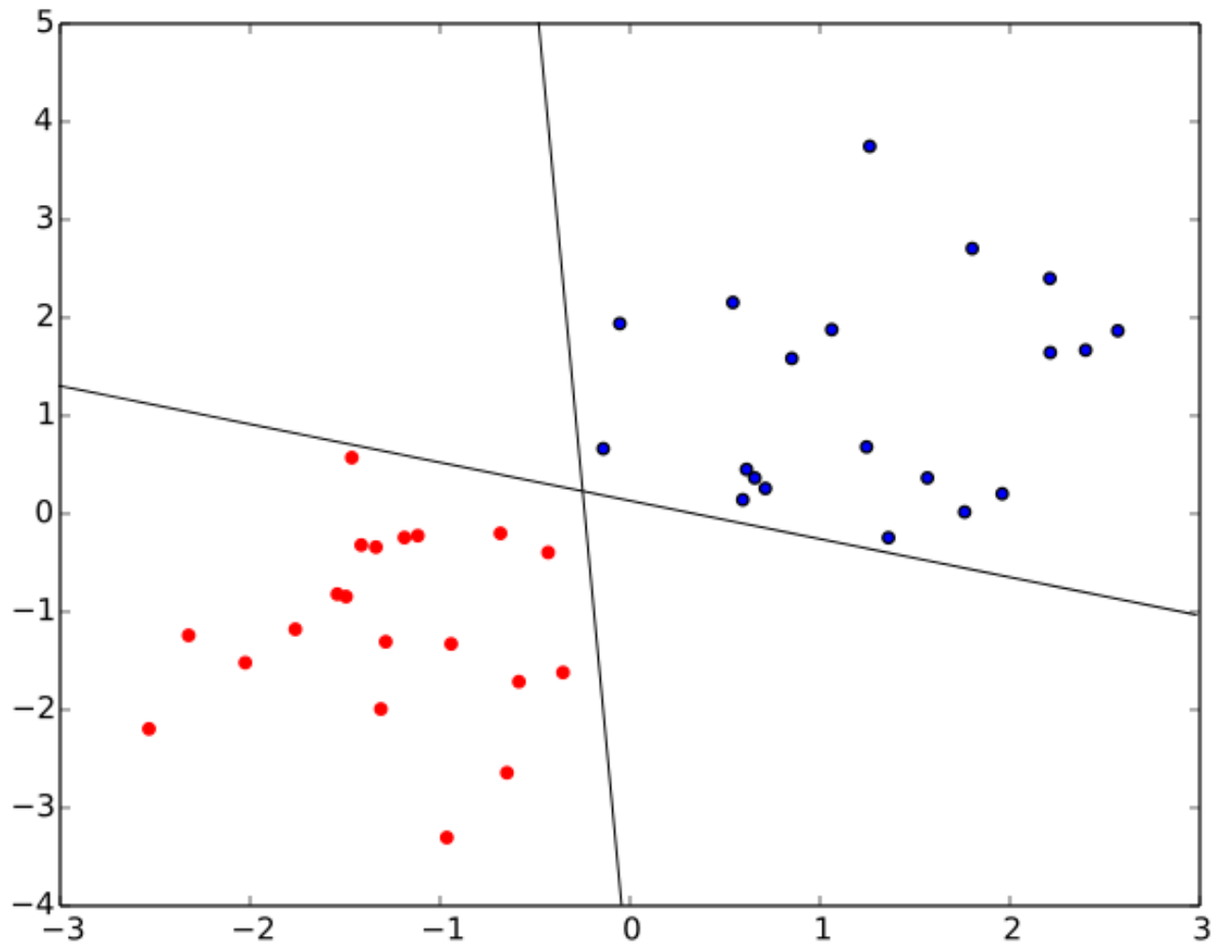
- Logistic regression does not rely on the statistically independency.
    - We can explore a lot of features without the consideration of their dependencies.

- Logistic regression model should be trained iteratively.
    - The training of naive Bayes model is much simpler and less time-consuming.
    - The computation of $P(x_i|y)$ is very easy

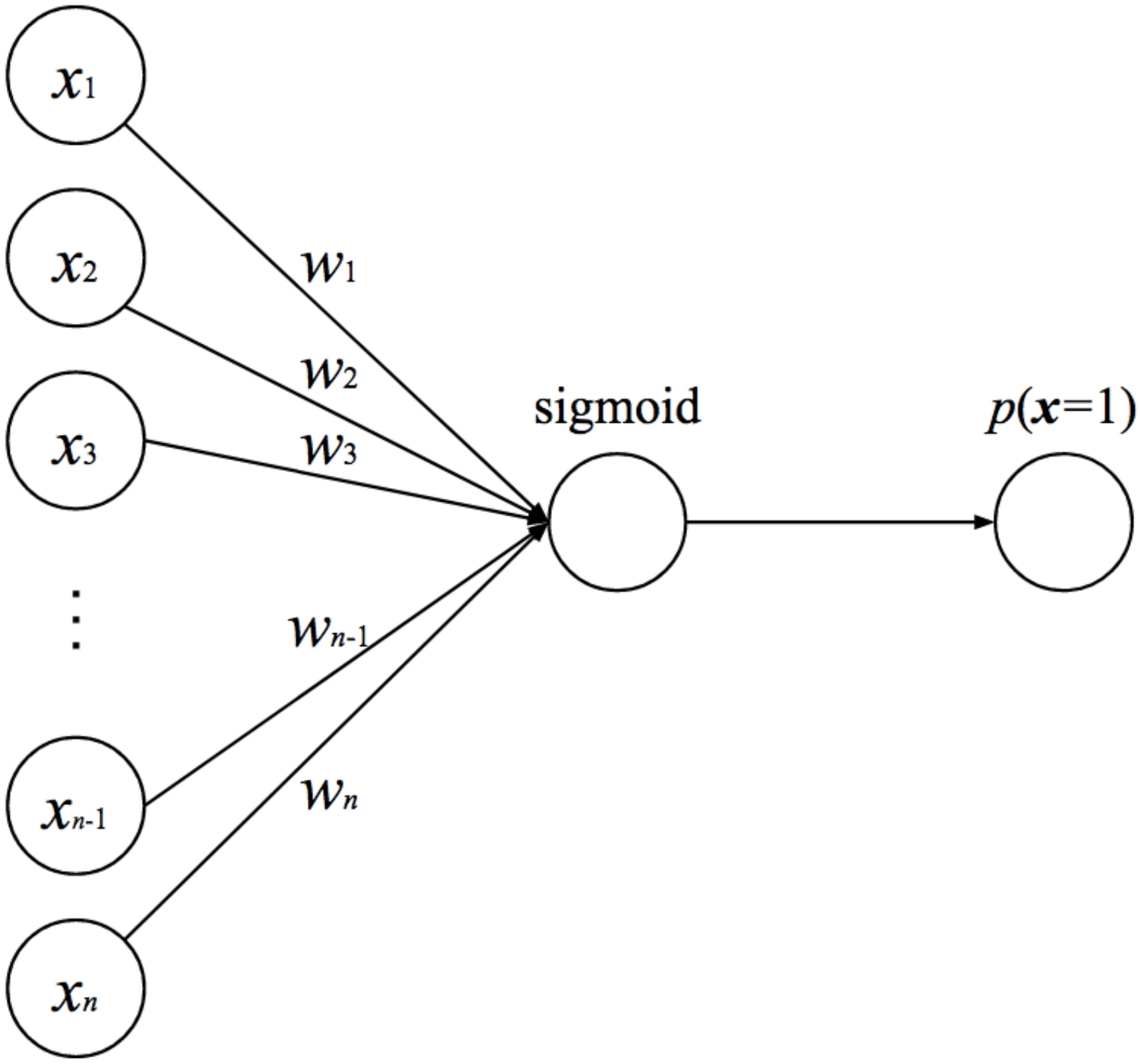
Maximum Margin

- Perceptron cannot choose the best splitting in the training space.

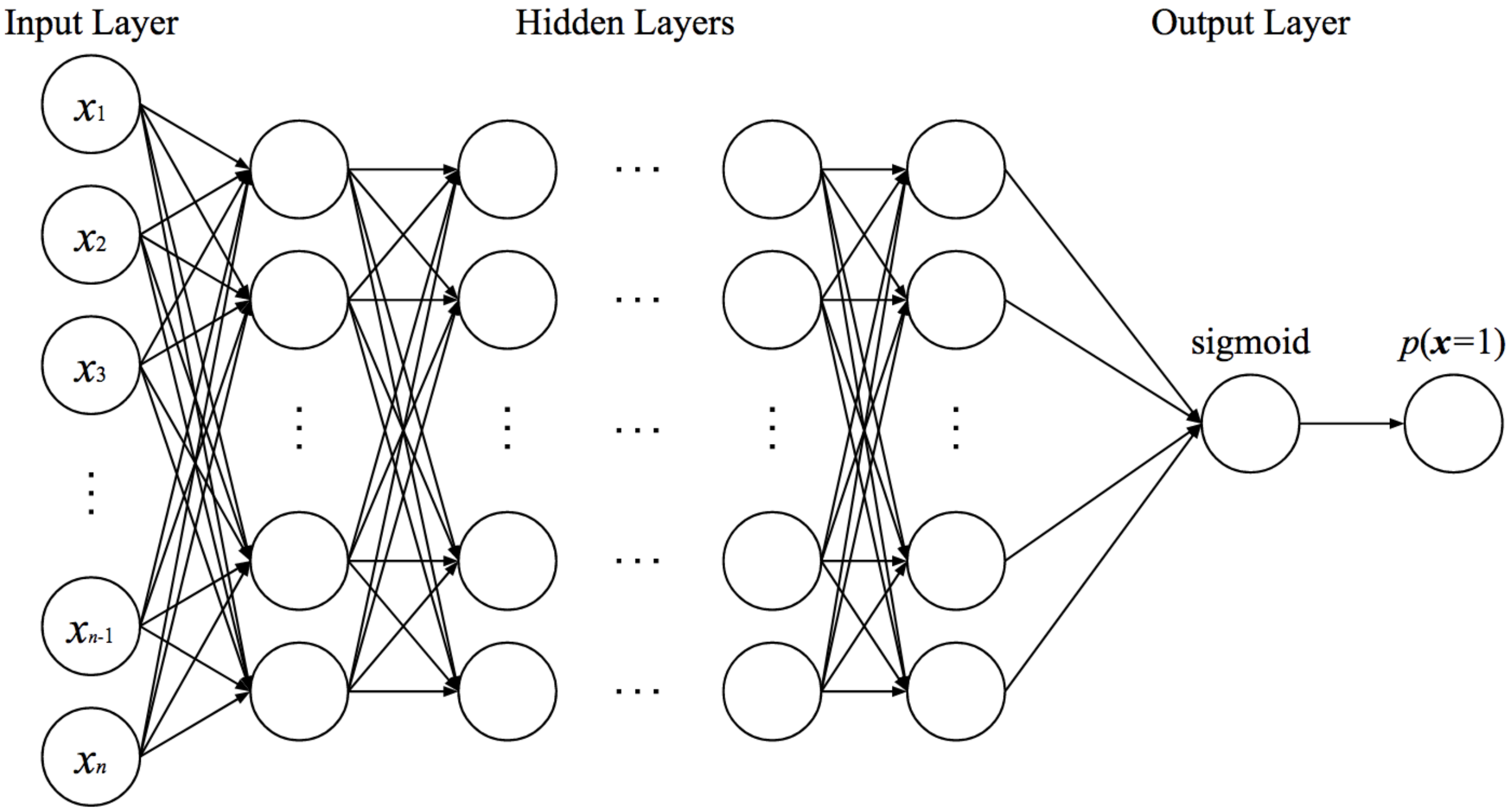
- Maximum margin models like SVM are designed to solve this problem.



Single Layer Neural Network



Feedforward Neural Network

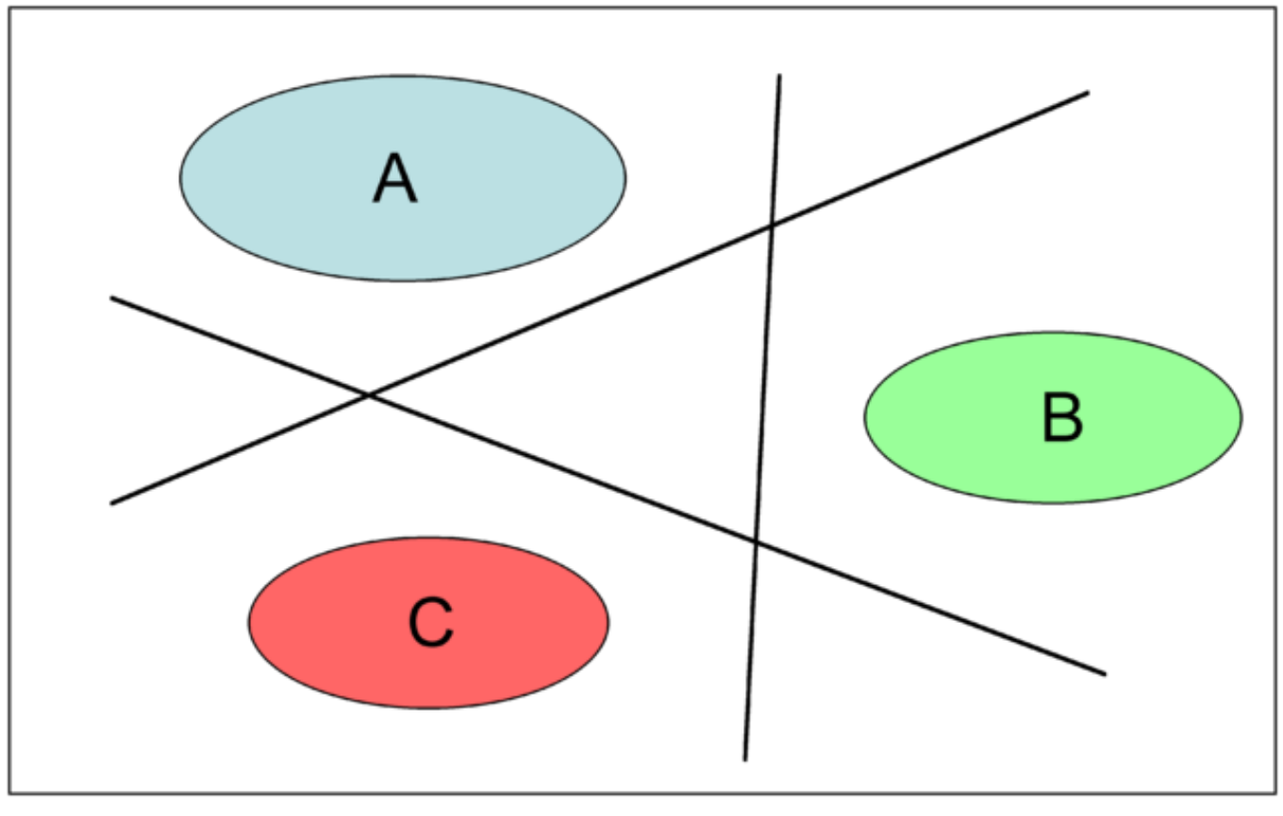


Properties of Deep Neural Network Models

- High expressive power
- Can handle the complex relationships among features.
  - In contrast, logistic regression or single layer perceptron is only to compute the weighted sum of all features.
    - Single layer perceptron cannot handle the relation such as:
      - {*good*} => positive
      - {*not*, *good*} => Negative
- Requiring large amount of data to train

Multiclass Classification

- Multiclass: Cat vs Dog vs Pig
- One vs Rest method
  - Cat or Not-Cat
  - Dog or Not-Dog
  - Pig or Not-Pig

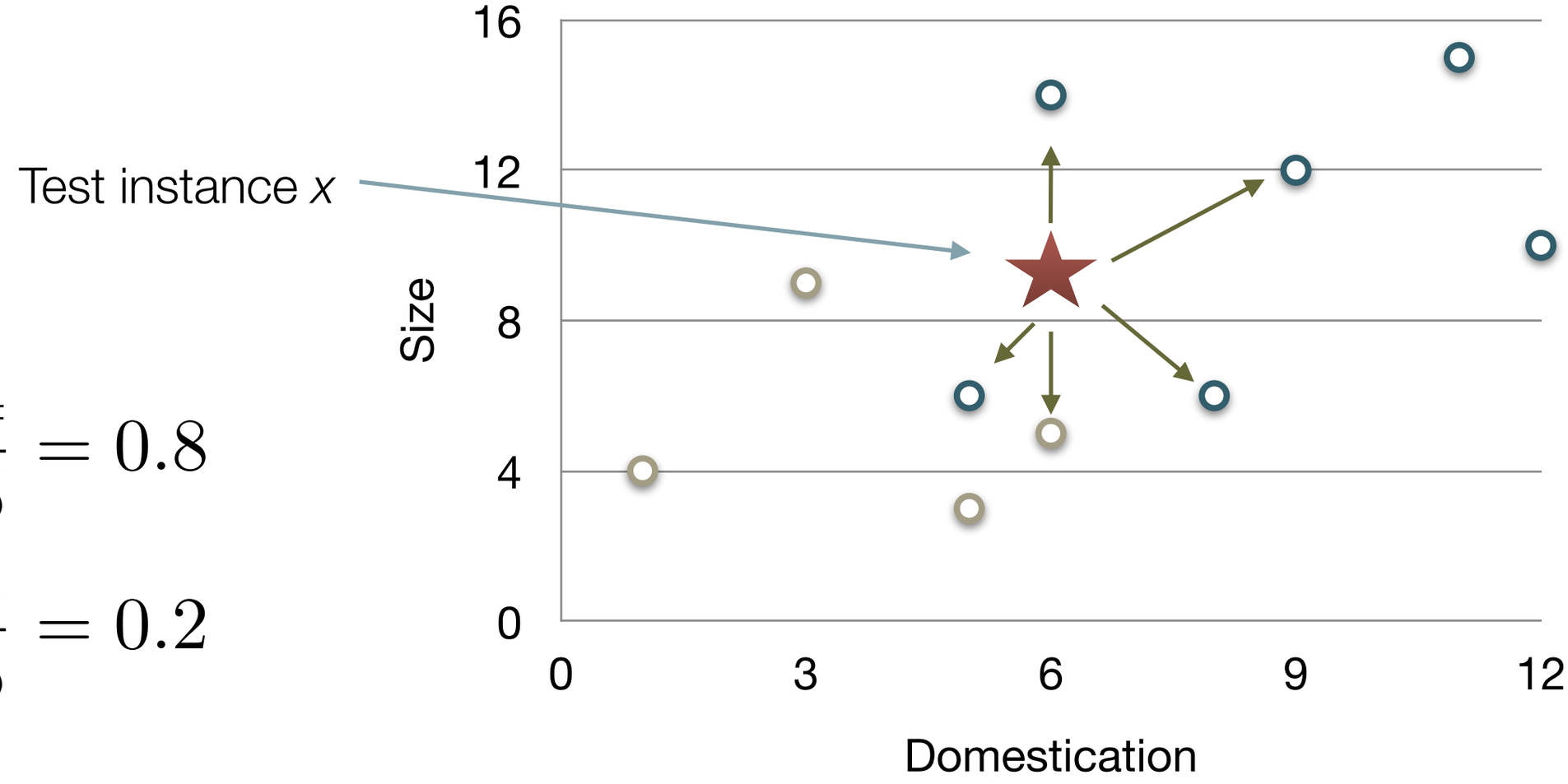


k Nearest Neighbor (*k*NN) Model

- Voting by *k* nearest training instances in the feature space.
- How to calculate the distance between instances?

$$P(\text{dog}|x) = \frac{4}{5} = 0.8$$

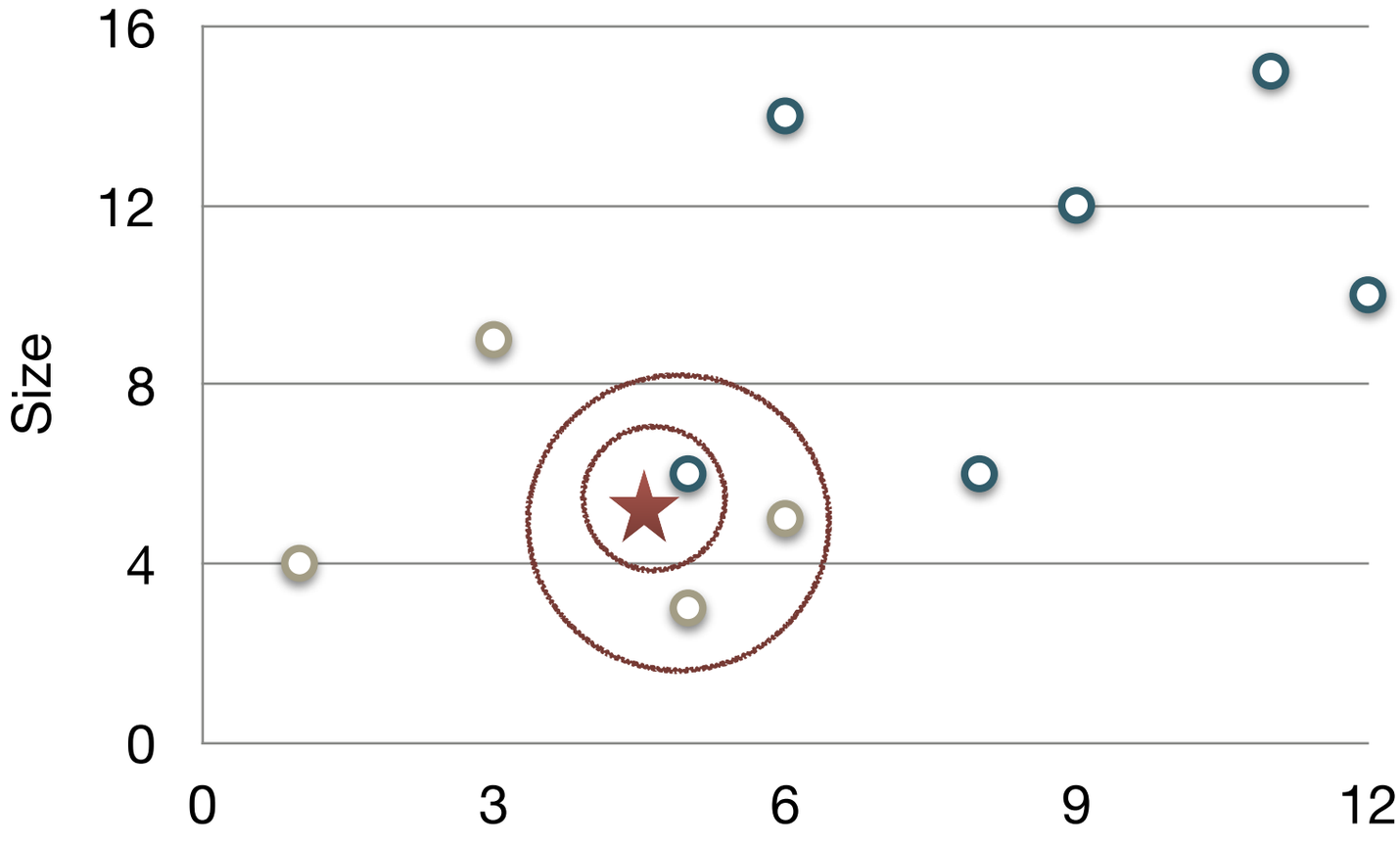
$$P(\text{cat}|x) = \frac{1}{5} = 0.2$$



Basic Idea of *k*NN

- Assign the instance *x* with a label that is the label of a training instance most similar to *x*.

- Extended to *k* most similar articles for better generalization.



Similarity Measurement

- The challenge of *k*NN is how to find a good similarity measurement.
    - Do not use kNN if no good similarity measurement is available.
- Similarity of two documents/sentences
    - Measure the distance between two vectors in a high dimensional space.

Jaccard Coefficient

- Counting the overlapped items in the vector *v* and the vector *u.*
- Normalizing by the length

$$\text{similarity} = \frac{|v \cap u|}{|v \cup u|}$$

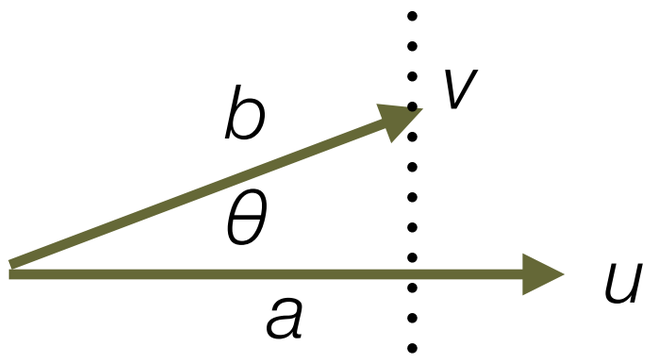
$$\text{similarity}(\text{good to watch}, \text{good movie})$$

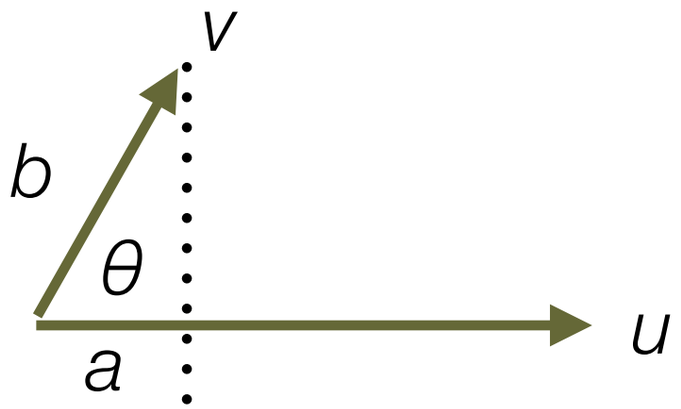
$$= \frac{|\{\text{good}\}|}{|\{good, to, watch, movie\}|} = \frac{1}{4}$$

Cosine Similarity

- The cosine of the angle θ between two vectors.
  - cos(0°) = 1
  - cos(90°) = 0
  - cos(180°) = -1
- Similar vectors have a smaller θ and a greater cos(θ)

$$\cos(\theta) = \frac{a}{b}$$





Cosine Similarity Calculation

- In a vector space with m dimensions, the cosine similarity of two vectors v and u can be calculated as follows.

$$\text{similarity} = \cos(\theta) = \frac{\vec{v} \cdot \vec{u}}{|\vec{v}||\vec{u}|}$$

← Overlapped items in two vectors

← Normalized by their lengths

$$\vec{v} \cdot \vec{u} = \sum_{i=1}^{m} v_i u_i \qquad |\vec{v}| = \sqrt{\sum_{i=1}^{m} {v_i}^2}$$

Dot product (内積)
of the two vectors

Length of a vector

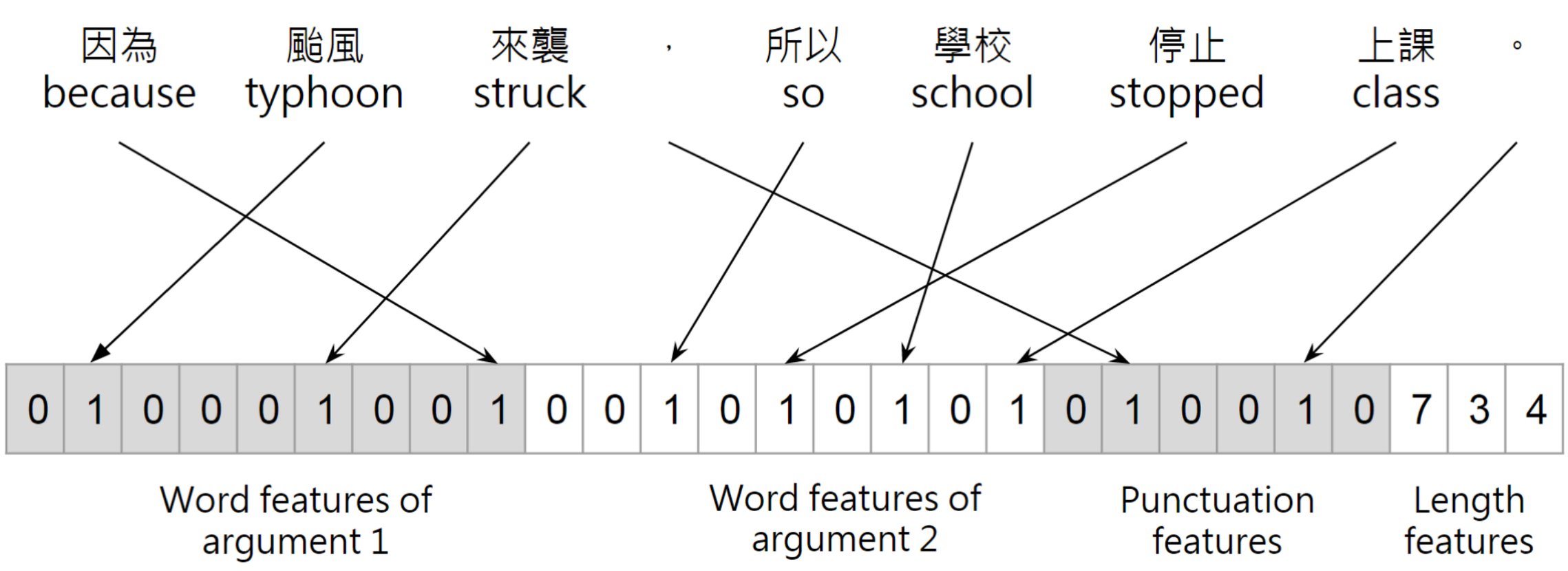
Properties of *k*NN

- Simple and no “training” at all.
  - The model can be updated frequently by adding more labeled instances.
- Less accurate in general
  - Performance is entirely dependent on the similarity measurement.

Feature Extraction

Feature Representation

- Each dimension in the vector indicates a certain type of feature.



Drawback of the Bag-of-Word Representation

- The bag-of-words scheme is very useful in many NLP and information retrieval tasks.
- One of its limitations is that it does not preserve the word order information.
- As a result, the relations among the words in an argument are missing from the feature vector.
- The bag-of-words representations of following two sentences are identical.
  - The meanings in these two sentences are indistinguishable.

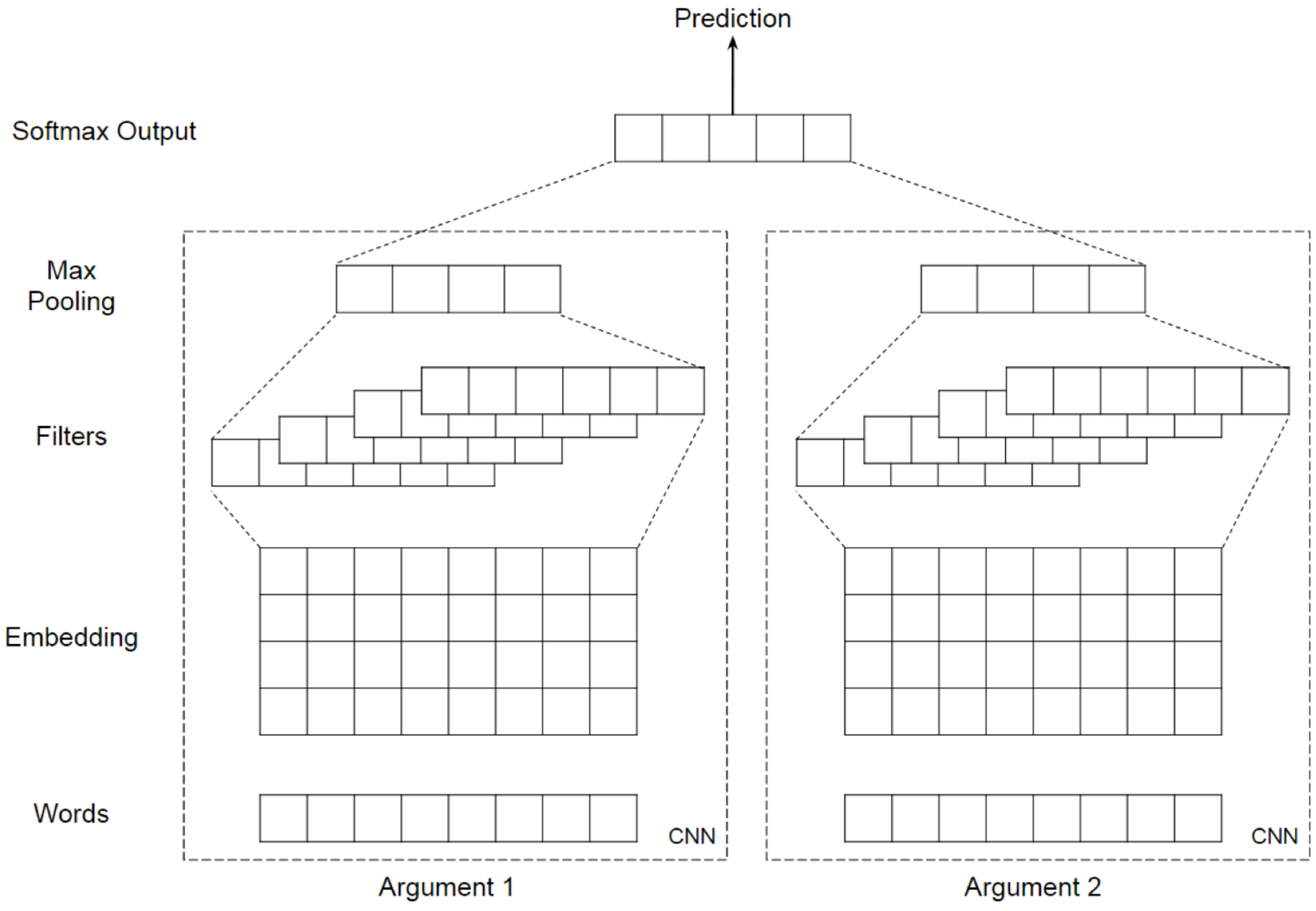
The cat catches the mouse.

The mouse catches the cat.

Neural Network-based Representation

- Deep neural networks like CNN and RNN are capable of extracting the word order information.
- Word embeddings
  - Representing each word in a dense vector.
    - Related words share similar representation in the vector space.
      - cosine(*tiger*, *lion*) > cosine(*tiger*, *communication*)
- Sentence embeddings or document embeddings
  - Representing a piece of text in a dense vector.

CNN Model for Discourse Relation Recognition



Linguistics Features

- Subword level
  - Phonetics, Character
- Word level
  - Word, N-grams, Part-of-Speech (POS), Discourse connective, Collocated words
- Syntax level
  - Constitution parsing, Dependency parsing, Topic-comment structure
- Others
  - Lengths
  - Punctuation Marks

Phonetics Features

- Initials, finals, and tones of the first character and the last character in a fragment.

- The syllabic feature useful in the speech recognition is unavailable in the written text.

- The pronunciation of each Chinese character as the phonetics information.

- The pronunciation combination between the last character of a fragment and the first character of the next fragment provides clues to the relationship between successive units.

Character Features

- In English, verb affixes like prefix and suffix provide some morphological information.
  - un-do, un-install
  - look-ing, watch-ing
- Also useful in Chinese because Chinese word segmentation can be skipped.
- Facebook fastText
  - Word embedding with subword information

Word Features

- The bag-of-word scheme.
- Some studies also include the n-grams of words for capturing the short-term word order information.
  - bag of words = bag of unigrams
  - bag of bigrams, bag of trigrams, and so on.
- Chinese word segmentation should be performed due to no delimiter between Chinese words.

Word Grouping

- Merge related words into one surface form
  - De-Capitalization
    - Dog, dog => dog
  - Stemming
    - dogs, dog => dog
  - Lemmatization
    - do, did, does, done => do
- Stopword removing

Hypernyms

- Word features tend to suffer from the sparsity issue.
- To reduce the sparsity, related words can be clustered to a group by consulting an ontology such as Chinese WordNet or Tongyici Cilin.
  - The words 汽車 (‘car’), 火車 (‘train’), and 摩托車 (‘motorcycle’) can be mapped to the concept 交通工具 (‘vehicle’) since vehicle is a hypernym of them.
- The word sequence is converted to a sequence of concepts. Similar to the Word feature, the bag-of-word scheme or deep neutral networks can extract the information from the sequences of concepts.

Part-of-Speech Tags

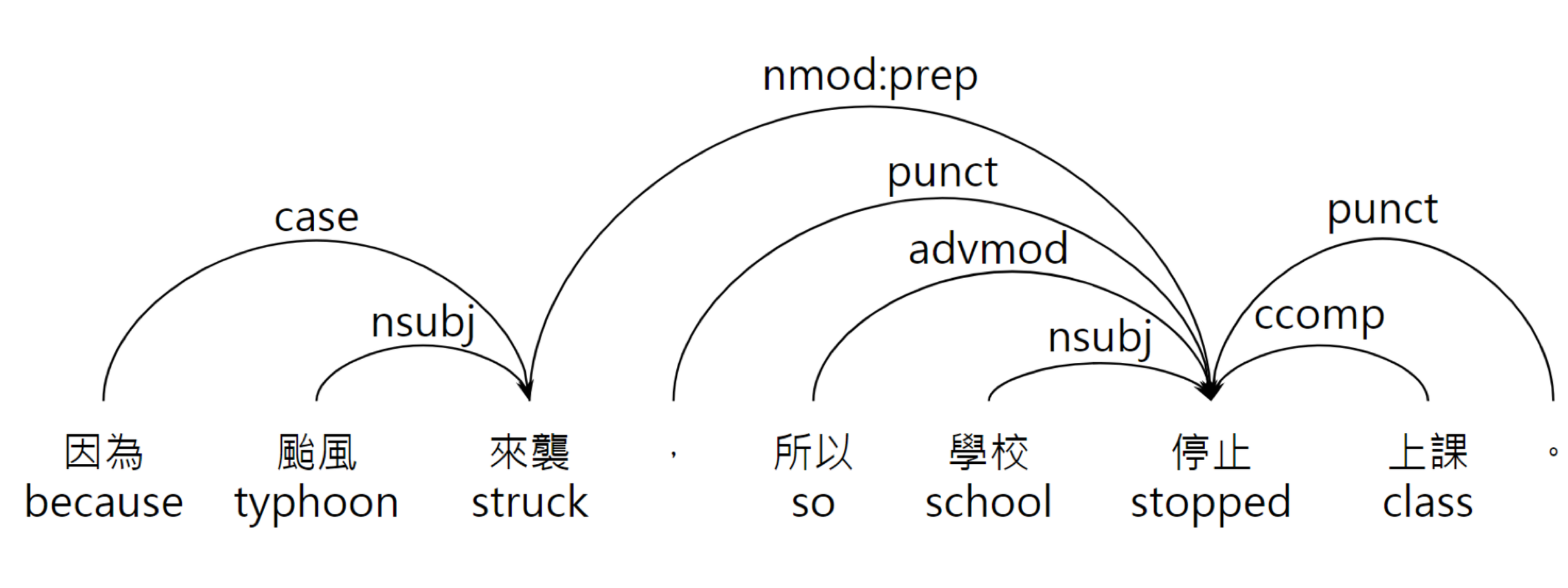
- Another manner for word clustering is part-of-speech (POS) tagging.

- POS tagger like Stanford CoreNLP (Manning et al., 2014) can be employed to label each word in a unit with its POS tag.

- Similar to the Word features, the bag-of-word scheme or deep network networks can extract the information from the sequences of POS tags.

Dependency Parse Tree

- To captures the relationships among the words in a sentence, the results of dependency parsing provide useful information.
- The two units have a closer connection if some dependencies occur across them.



Constitution Parse Tree

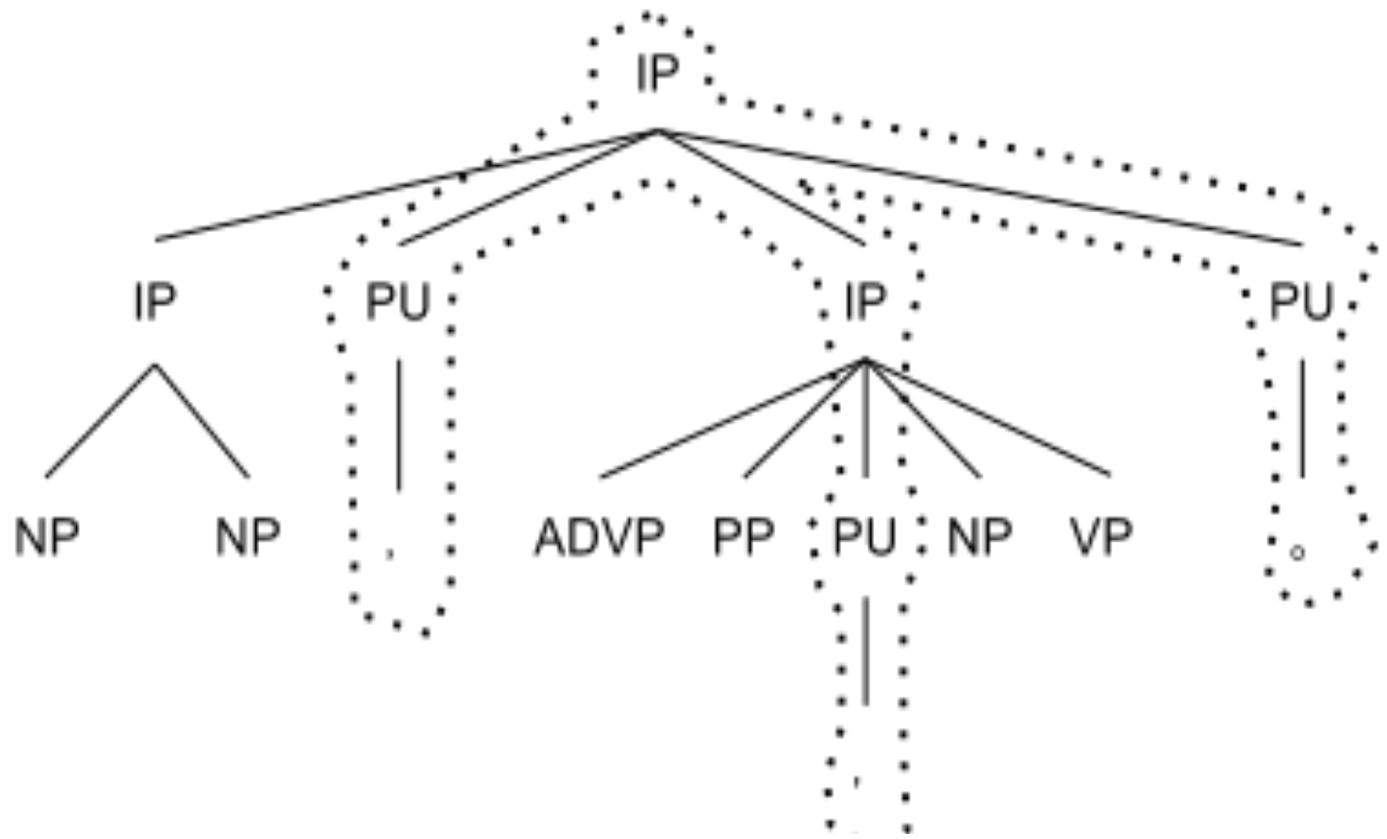
- To capture syntactic information, constituency parsing is performed to generate the syntactic trees of the units, and syntactic information such as the categories of a tree node, its parent, its left sibling, and its right sibling are taken as features.

- In each sentence, the sub-tree of the upper three levels and the paths from the root to each punctuation node in the sentence can also be extracted as features.

Constitution Parse Tree

- The sub-tree of the upper three levels, which represent the fundamental composition of the sentence.

- All the path from root to each punctuation.



Connectives

- In explicit discourse relation recognition, connectives in the units are extremely important.

- The connectives can be further mapped to a type of discourse relation by dictionary lookup, and the types of the connectives in the two arguments are considered as features.

- The type of *but* and *however* is Comparison, and the that of *meanwhile* and *while* is Temporal.

Other Features

- Length: the lengths of input data
  - Number of characters
  - Number of words
  - Number of clauses
  - Number of sentences
- Position
  - The position of the word in the sentence
  - The position of the sentence in the paragraph/document
- Punctuation: The punctuations in argument 1 and argument 2 are regarded as features. Most common punctuations include commas, periods (full-stops), question marks, and exclamation marks.

Analysis of the Classification Results

- Exploring the behavior of the model
- Gaining insights for improving the model
- Strategies
    - Confusion matrix
        - A general approach working for all classification models
    - Finding important features
        - Works for linear models
    - Finding decision rules
        - Works for decision trees

Confusion Matrix

	Predicted as Dog	Predicted as Cat
Actual Dog	34	11
Actual Cat	5	18

Confusion Matrix

- Extend to multi-way classification

	Predicted as Political	Predicted as World	Predicted as Society	Predicted as Sports
Actual Political	245	76	120	15
Actual World	26	92	17	10
Actual Society	106	55	312	24
Actual Sports	6	32	17	89

Important Features

- We can exam the important features from simple classifiers like the logistic regression model

- View the most important features by selecting the features with maximum/minimum coefficients.

Most Contributing Features

- A logistic regression model with only word (unigram) features
- View the features with highest and lowest weights for movie review sentiment analysis

Word	Weight
fun	0.468551
seen	0.464473
great	0.435241
well	0.421405
back	0.386793
overall	0.35757
change	0.356973
quite	0.340318
perfectly	0.338054
pulp	0.317991

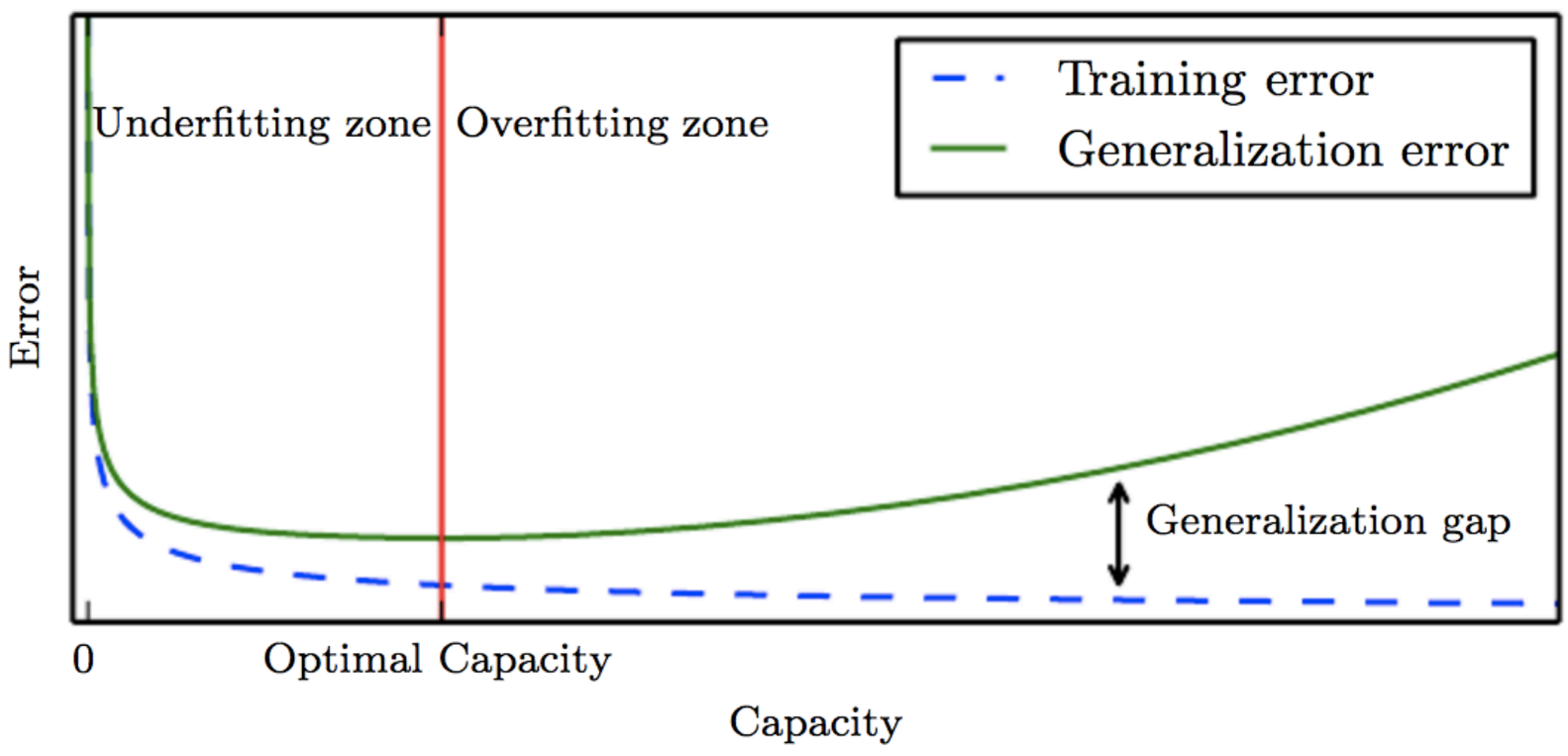
Word	Weight
meekness	0
relatable	0
albany	0
boastfully	0
cronfronting	0
crowning	0
daper	0
deterent	0
exuberhant	0
freewill	0

Word	Weight
bad	-0.755196
unfortunately	-0.634278
worst	-0.614966
nothing	-0.498185
script	-0.469493
waste	-0.467703
only	-0.467102
poor	-0.43059
boring	-0.430199
plot	-0.420406

Overfitting

Training Performance vs Validation Performance

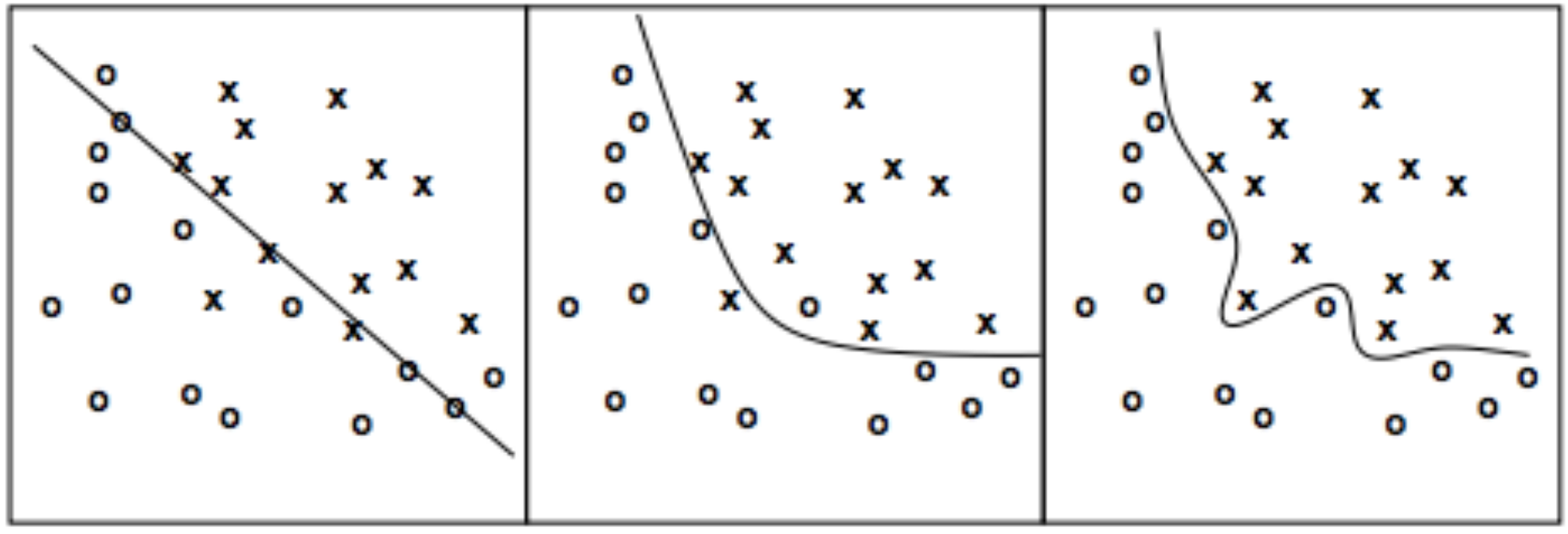
- Training performance
  - Used to measure model's ability to fit the data.
- Testing (validation) performance
  - Used to measure the performance in real applications.



Overfitting

- Training performance >> testing performance

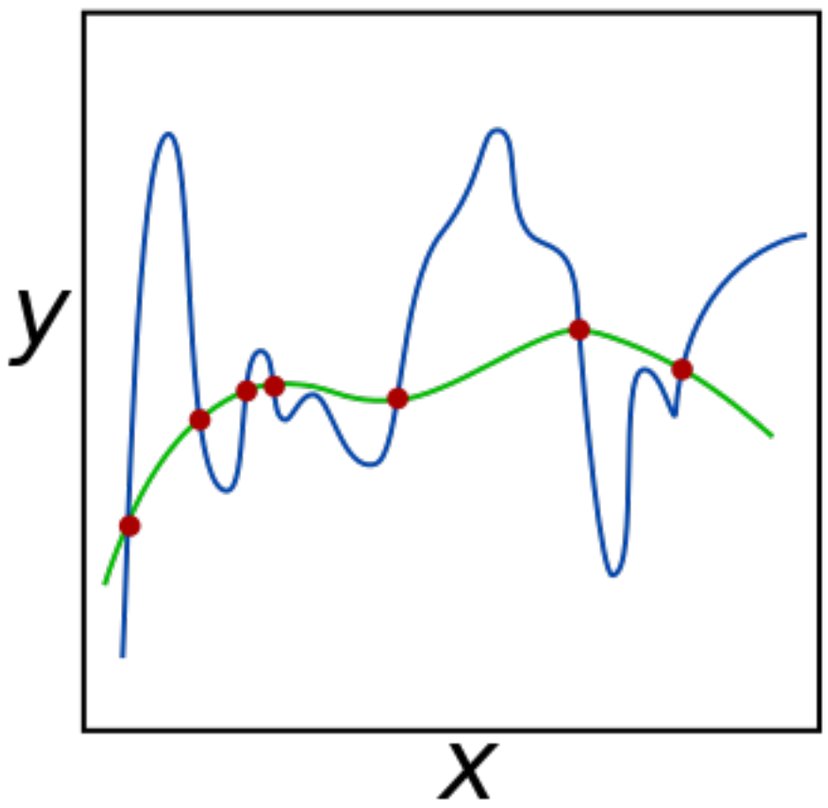
"The most likely hypothesis is the simplest one consistent with the data."



Data Sparseness

- Reasons of overfitting
  - Too specific features
  - Too complex model
- Solutions
  - More training data
  - Reduce the complexity of the model
  - Feature selection
  - Regularization
-

$$D = \{ (x_{1,1}, x_{1,2}, x_{1,3}, ..., x_{1,n}, y_1), (x_{2,1}, x_{2,2}, x_{2,3}, ..., x_{2,n}, y_2), ..., (x_{m,1}, x_{m,2}, x_{m,3}, ..., x_{m,n}, y_m)\}$$
, where $n \gg m$

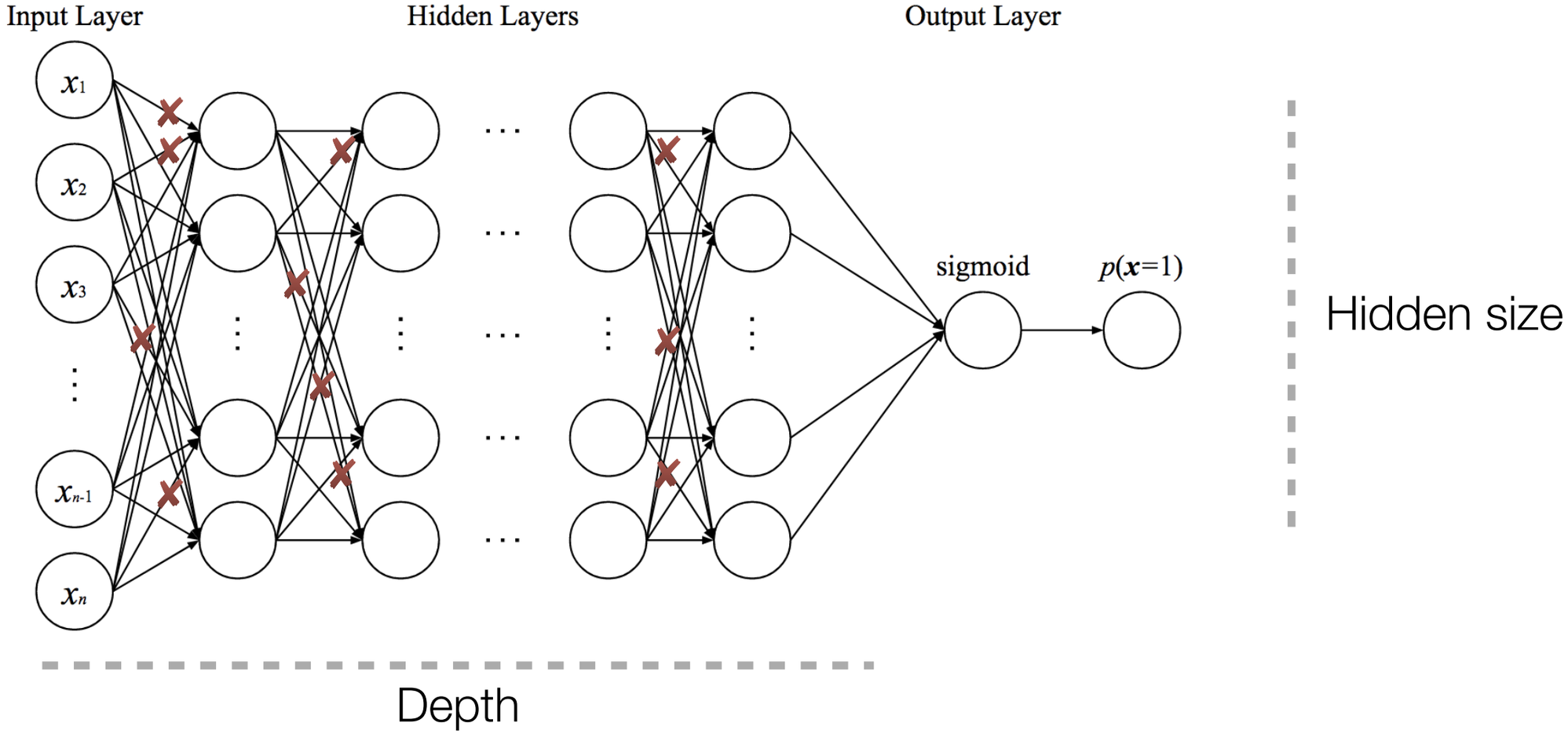


Dealing with Sparsity

- Feature selection
  - Removing unimportant features
- Naive Bayes
  - Smoothing
- Decision tree
  - Pruning
- Logistic regression
  - Increasing regularization by decreasing the hyperparameter c

Dealing with Sparsity

- Neural network
  - Decreasing the depth of the neural network
  - Decreasing the size of hidden layers
  - Dropout



Feature Selection

- Removing low frequency features
  - One-time appearing words
- Verifying the correlation between a feature and the labels.
  - Chi-square test
- Feature selection by model

Reducing the Features with Non-zero Coeffiecents

- The more variables are used, the more likely the model overfits the training data.

- The more nonzero coefficients are in the model, the more difficult it is to interpret of the model.

- For explanation, a model with 3 features that achieves an accuracy of 90% is better than a model with 1000 features that achieves an accuracy of 92%.

Regularization

- Regularization is an approach in which we add to the error term a penalty that gets larger as number of nonzero coefficients gets larger.

- Then we minimize the combined error and penalty.

- The more importance we place on the penalty term, the more we discourage large coefficients.

Lasso Regularization

- Least absolute shrinkage and selection operator
- L_1 norm for regularization
  - Manhattan distance

$$||W||_1 = |w_1| + |w_2| + |w_3| + ... + |w_n|$$

- Loss function with L_1 regularization

$$L_1 = (y - y')^2 - \lambda||W||_1 = (y - y')^2 - \lambda \sum_{i=1}^{n} |w_i|$$

Ridge Regression

- L_2 norm (squared norm) for regularization
  - Squared Euclidean distance

$$||W||_2^2 = w_1^2 + w_2^2 + w_3^2 + ... + w_n^2$$

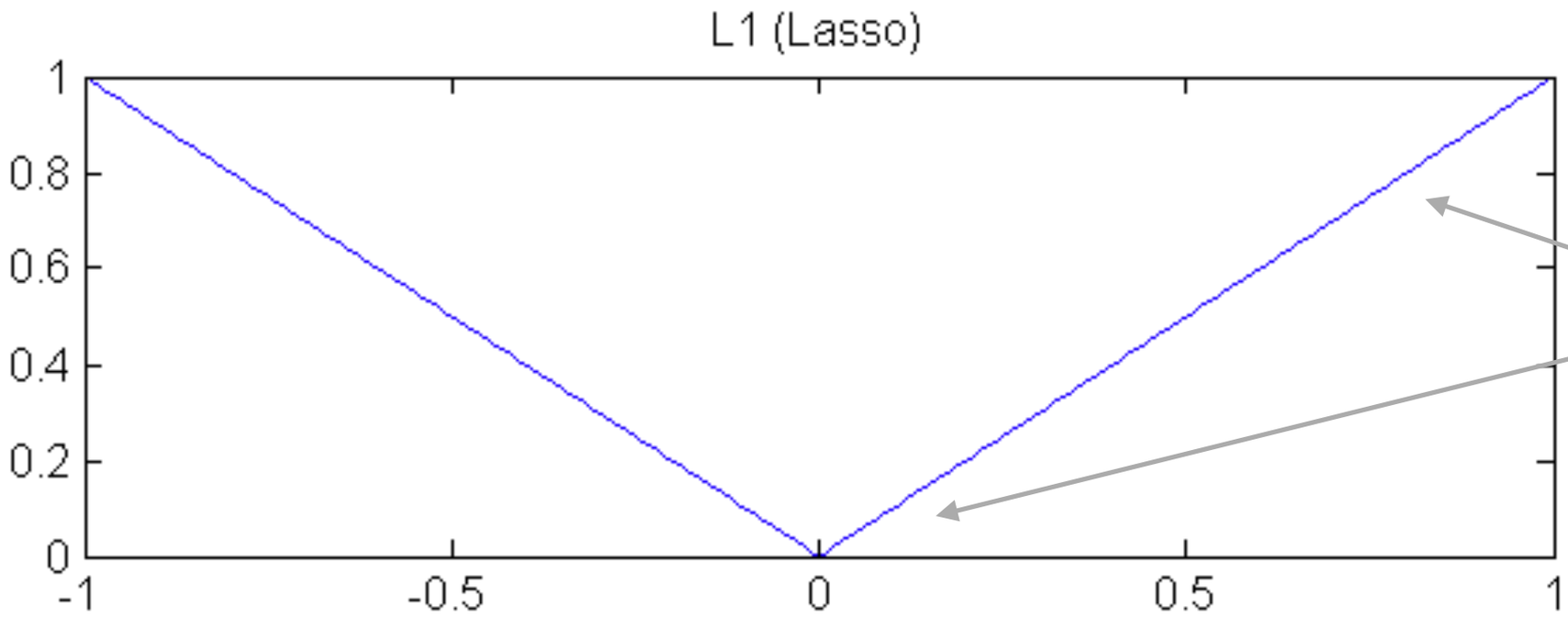
- Loss function with L_2 regularization

$$Loss_{R_{L_2}} = (y - y')^2 - \lambda||W||_2^2 = (y - y')^2 - \lambda \sum_{i=1}^{n} w_i^2$$

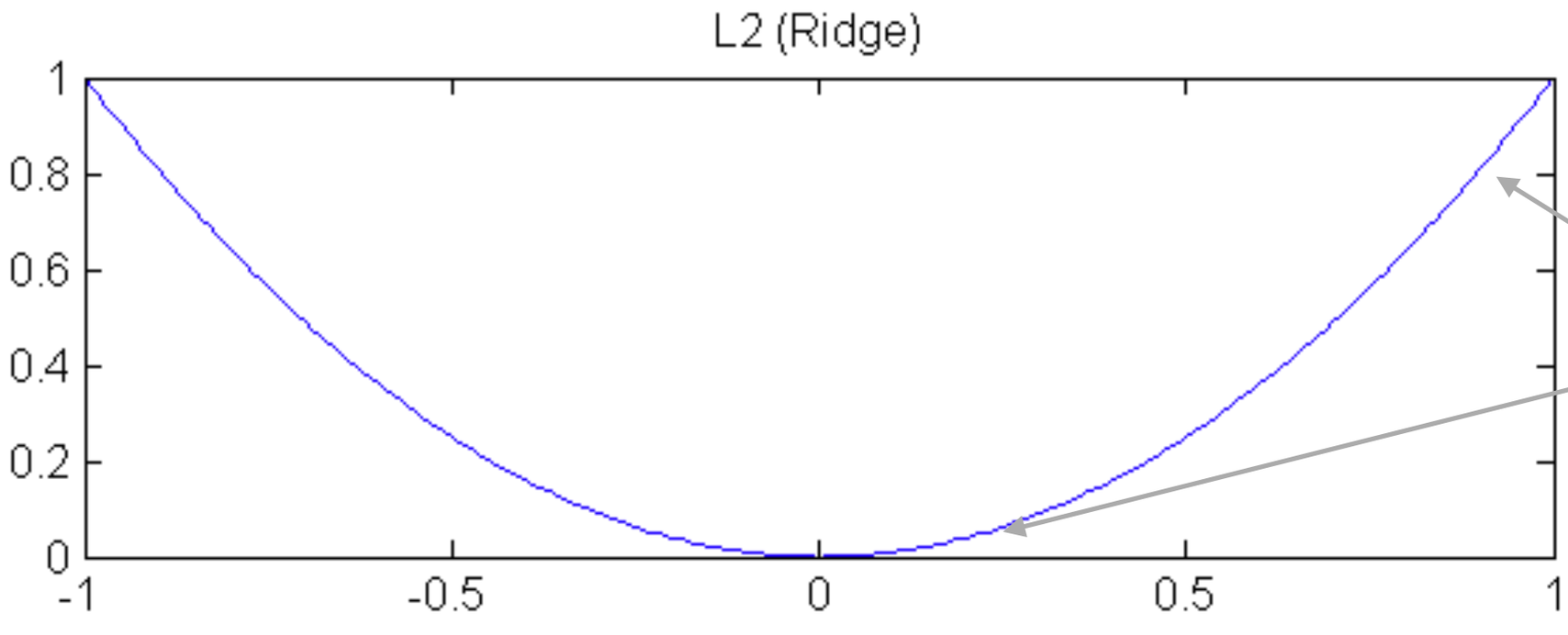
L1 vs L2 Regularization Methods

- L2 severely punishes for high parameter wights, but once the value is close enough to zero, their effect becomes insignificant.
    - Prefer to decrease value of a parameter with high weight by 1 than to decrease the value of ten parameters that already have relatively low weights by 0.1 each.
- L1 punishes **uniformly** for both low and high values.
    - To decrease all the non-zero parameter values toward 0.
    - Resulting a sparse solution (many parameters are 0)

L1 vs L2 Regularization Methods



Uniform penalty for each value



Higher penalty for the higher value

Elastic-Net

- A regularization method that combines both L_1 and L_2

$$Loss_{R_{EN}} = (y - y')^2 - (\lambda_1 ||W||_1 + \lambda_2 ||W||_2^2)$$