# Natural Language Processing

## Text classification

Yulia Tsvetkov

yuliats@cs.washington.edu

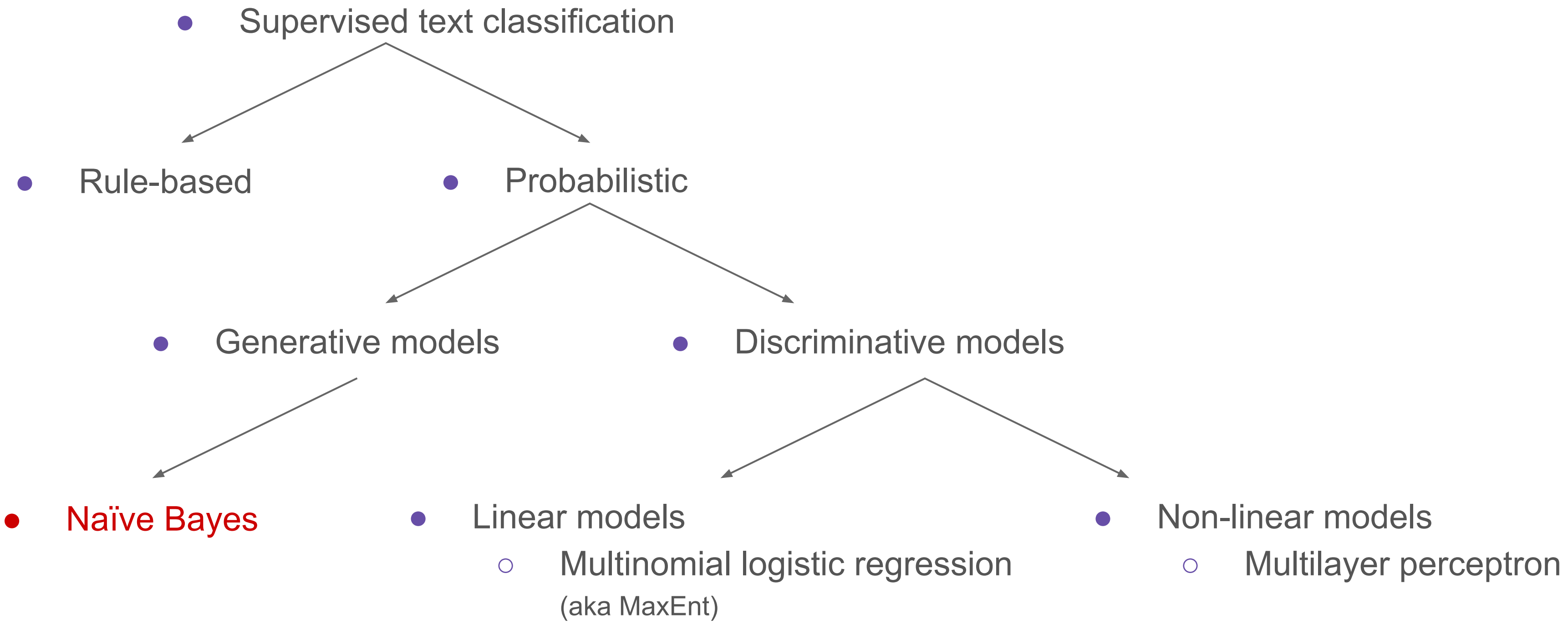
PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

# Announcements

- No quiz next week!
- No class on Friday this week!
- Please start your HW1 early

# Recap: Naive Bayes

# We consider various models for classification



- Supervised text classification
- Rule-based
- Probabilistic
- Generative models
- Discriminative models
- Naïve Bayes
- Linear models
  - Multinomial logistic regression (aka MaxEnt)
- Non-linear models
  - Multilayer perceptron

# Generative and discriminative models

- Generative text classification: Learn a model of the joint P(X, y), and find

$$\hat{y} = \underset{\tilde{y}}{\operatorname{argmax}}\, P(X, \tilde{y})$$

- Discriminative text classification: Learn a model of the conditional P(y | X), and find

$$\hat{y} = \underset{\tilde{y}}{\operatorname{argmax}}\, P(\tilde{y} | X)$$

# Generative and discriminative models

## Toy Example: Spam Email Detection

### Generative models

**Learn:** P(email, class)

**Approach:** Model what spam and legiti emails look like

**Example (Naive Bayes):**

- Learn P("free" | spam) = 0.8
- Learn P("money" | spam) = 0.7
- Learn P("free" | legitimate) = 0.1
- Learn P("money" | legitimate) = 0.2
- Learn P(spam) = 0.3

**Classify new email:** "free money"

P(email, spam) = P(“free”l | spam) x P(“money”l | spam) × P(spam) = (0.8 × 0.7) × 0.3

P(email, legit) = P(“free”l | legit) x P(“money”l | legit) × P(legit) = (0.1 × 0.2) × 0.7

### Discriminative models

**Learn:** P(class | email), given the email, what is the probability of each class

Only classify, not able to generate new documents

# Generative text classification: naïve Bayes

- Simple classification method
  - based on the Bayes rule
- Relies on very simple (naïve) representation of a documents
  - Conditional independence assumption:
    the features are conditionally independent, given the target class
    (hence the name "naïve")
  - bag-of-words, no relative order
- A good baseline for more sophisticated models

Andrew Y. Ng and Michael I. Jordan, On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes, Advances in Neural Information Processing Systems 14 (NIPS), 2001.

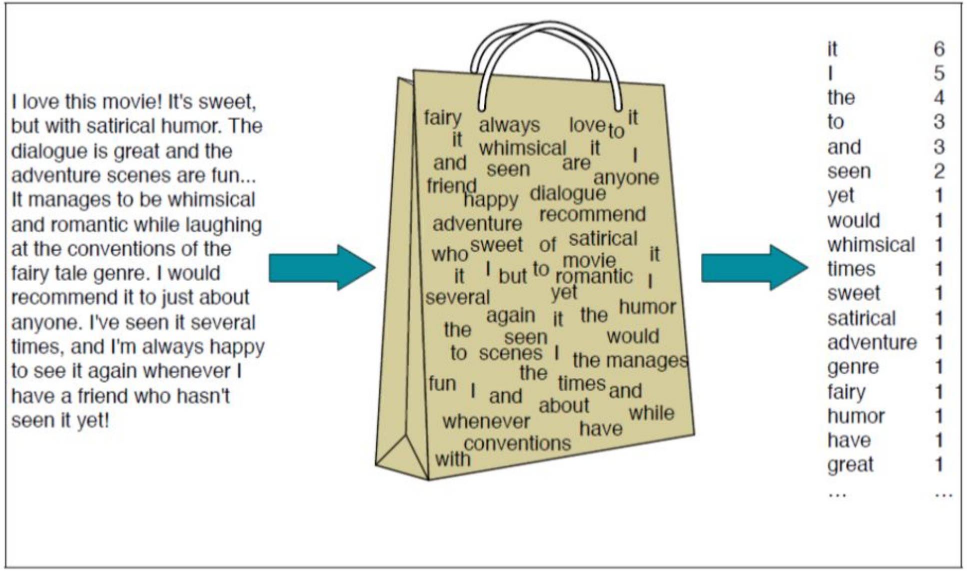
# Naïve Bayes

## Sentiment analysis: movie reviews

- Given a document d (e.g., a movie review)
- Decide which class c it belongs to: positive, negative, neutral
- Compute P(c | d) for each c
  - P(positive | d), P(negative | d), P(neutral | d)
  - select the one with max P

# Bag-of-Words (BOW)

- Given a document d (e.g., a movie review) – how to represent d ?



I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

| | |
|---|---|
| it | 6 |
| I | 5 |
| the | 4 |
| to | 3 |
| and | 3 |
| seen | 2 |
| yet | 1 |
| would | 1 |
| whimsical | 1 |
| times | 1 |
| sweet | 1 |
| satirical | 1 |
| adventure | 1 |
| genre | 1 |
| fairy | 1 |
| humor | 1 |
| have | 1 |
| great | 1 |
| ... | ... |

**Figure 7.1** Intuition of the multinomial naive Bayes classifier applied to a movie review. The position of the words is ignored (the *bag of words* assumption) and we make use of the frequency of each word.

Figure from J&M 3rd ed. draft, sec 7.1

# Naïve Bayes

- Given a document d and a class c, use Bayes' rule:

$$\underset{\substack{\big\downarrow \\ \text{posterior}}}{P(c|d)} = \frac{P(d|c)P(c)}{P(d)}$$

# Naïve Bayes

- Given a document d and a class c, Bayes' rule:

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

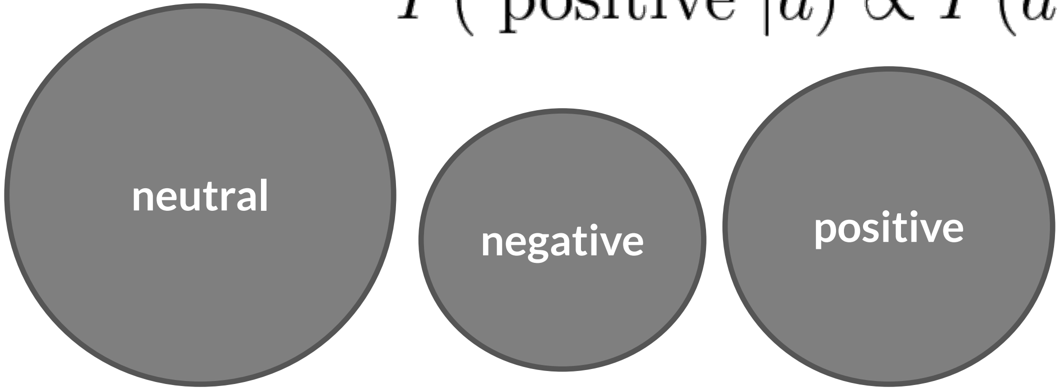
$$\underbrace{P(\text{`positive'}|d)}_{\text{posterior}} \underset{\text{Proportional to}}{\propto} \underbrace{P(d|\text{`positive'})}_{\text{likelihood}}\underbrace{P(\text{`positive'})}_{\text{prior}}$$

# Naïve Bayes

- Given a document d and a class c, Bayes' rule:

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

$$P(\text{`positive'}|d) \propto P(d|\text{`positive'})\underbrace{P(\text{`positive'})}_{\text{prior}}$$



# Naïve Bayes

- Given a document d and a class c, Bayes’ rule:

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

$$P(\text{‘positive’}|d) \propto \underset{\substack{\Big\downarrow \\ \text{likelihood}}}{P(d|\text{‘positive’})}\,P(\text{‘positive’})$$

# Naïve Bayes independence assumptions

$$P(d|\text{‘positive’})$$

$$P(w_1, w_2, \ldots, w_n|c)$$

- **Bag of Words assumption**: Assume word order doesn't matter
- **Conditional Independence**: Assume the feature probabilities $P(w_i|c_j)$ are independent given the class $c$

  (Naive: think of an example, “boring” and “sleep”)

$$P(w_1, w_2, \ldots, w_n|c) = P(w_1|c) \times P(w_2|c) \times P(w_3|c) \times \ldots \times P(w_n|c)$$

# Document representation

I love this movie. It's sweet but with satirical humor. The dialogue is great and the adventure scenes are fun… it manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

→ **bag of words (BOW)** →

| | |
|---|---|
| it | 6 |
| I | 5 |
| the | 4 |
| to | 3 |
| and | 3 |
| seen | 2 |
| yet | 1 |
| would | 1 |
| whimsical | 1 |
| times | 1 |
| sweet | 1 |
| satirical | 1 |
| adventure | 1 |
| genre | 1 |
| fairy | 1 |
| humor | 1 |
| have | 1 |
| great | 1 |
| … | … |

# Document representation

I love this movie. It's sweet but with satirical humor. The dialogue is great and the adventure scenes are fun… it manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

→ **bag of words (BOW)** →

| | |
|---|---|
| it | 6 |
| I | 5 |
| the | 4 |
| to | 3 |
| and | 3 |
| seen | 2 |
| yet | 1 |
| would | 1 |
| whimsical | 1 |
| times | 1 |
| sweet | 1 |
| satirical | 1 |
| adventure | 1 |
| genre | 1 |
| fairy | 1 |
| humor | 1 |
| have | 1 |
| great | 1 |
| … | … |

$$P(d|c) = P(w_1, w_2, \ldots, w_n|c) = \prod_i P(w_i|c)$$

# Generative text classification: Naïve Bayes

$$\mathrm{C}_{NB} = \underset{c}{\operatorname{argmax}} P(c|d) = \underset{c}{\operatorname{argmax}} \frac{P(d|c)P(c)}{P(d)} \propto$$

Bayes rule

$$\underset{c}{\operatorname{argmax}} P(d|c)P(c) =$$

same denominator

$$\underset{c}{\operatorname{argmax}} P(w_1, w_2, \ldots, w_n|c)P(c) =$$

representation

$$\underset{c_j}{\operatorname{argmax}}\ P(c_j) \prod_i P(w_i|c)$$

conditional independence

# Underflow prevention: log space

- Multiplying lots of probabilities can result in floating-point underflow

  P(review | positive) = $(0.1)^{50}$

  $= 10^{-50}$

  = 0.0000000000000000000000000000000000000000000000001

- Since log(xy) = log(x) + log(y)
  - better to sum logs of probabilities instead of multiplying probabilities
- Class with highest un-normalized log probability score is still most probable

$$\mathrm{C}_{NB} = \underset{c_j}{\operatorname{argmax}}\ P(c_j) \prod_i P(w_i|c)$$

$$\mathrm{C}_{NB} = \underset{c_j}{\operatorname{argmax}}\ log(P(c_j)) + \sum_i log(P(w_i|c))$$

# Learning the multinomial naïve Bayes

- How do we learn (train) the NB model?

# Learning the multinomial naïve Bayes

- How do we learn (train) the NB model?
- We learn $P(c)$ and $P(w_i|c)$ from training (labeled) data

$$\mathrm{C}_{NB} = \underset{c_j}{\operatorname{argmax}}\; log(\underline{P(c_j)}) + \sum_i log(\underline{P(w_i|c)})$$

# Parameter estimation for NB

- Parameter estimation during training
- Concatenate all documents with category c into one mega-document
- Use the frequency of $w_i$ in the mega-document to estimate the word probability

$$\mathrm{C}_{NB} = \underset{c_j}{\operatorname{argmax}}\ log(P(c_j)) + \textstyle\sum_i log(P(w_i|c))$$

$$\hat{P}(c_j) = \frac{doccount(C = c_j)}{N_{doc}}$$

$$\hat{P}(w_i|c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$$

# Parameter estimation for NB

$$\hat{P}(w_i|c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$$

- fraction of times word $w_i$ appears among all words in documents of topic $c_j$

- Create mega-document for topic $j$ by concatenating all docs in this topic
  - Use frequency of $w$ in mega-document

# Problem with Maximum Likelihood

- What if we have seen no training documents with the word “fantastic” and classified in the topic positive?

# Problem with Maximum Likelihood

- What if we have seen no training documents with the word “fantastic” and classified in the topic positive?

$$\hat{P}(\text{“}fantastic\text{”}|c = \text{positive}) = \frac{count(\text{“}fantastic\text{”}, \text{positive})}{\sum_{w \in V} count(w, \text{positive})} = 0$$

- Zero probabilities cannot be conditioned away, no matter the other evidence!

$$\underset{c_j}{\text{argmax}}\ P(c_j) \prod_i P(w_i|c)$$

# Laplace (add-1) smoothing for naïve Bayes

$$\hat{P}(w_i|c_j) = \frac{count(w_i, c_j) + 1}{\sum_{w \in V}(count(w, c_j) + 1)}$$

# Laplace (add-1) smoothing for naïve Bayes

$$\hat{P}(w_i|c_j) = \frac{count(w_i, c_j) + 1}{\sum_{w \in V}(count(w, c_j) + 1)}$$

$$= \frac{count(w_i, c_j) + 1}{(\sum_{w \in V}(count(w, c_j)) + |V|}$$

- Note about log space

# Multinomial naïve Bayes : learning

- From training corpus, extract *Vocabulary*
- Calculate $P(c_j)$ terms
    - For each $c_j$ do
        - $docs_j \leftarrow$ all docs with class = $c_j$
        - $P(c_j) \leftarrow \dfrac{|docs_j|}{total\ \#\ documents}$

# Multinomial naïve Bayes : learning

- From training corpus, extract *Vocabulary*
- Calculate $P(c_j)$ terms
  - For each $c_j$ do
    - $docs_j \leftarrow$ all docs with class = $c_j$
    - $P(c_j) \leftarrow \dfrac{|docs_j|}{total\ \#\ documents}$
- Calculate $P(w_i | c_j)$ terms
  - $Text_j \leftarrow$ single doc containing all $docs_j$
  - For each word $w_i$ in *Vocabulary*
    - $n_i \leftarrow$ # of occurrences of $w_i$ in $Text_j$
    - $P(w_j | c_j) \leftarrow \dfrac{n_i + \alpha}{n + \alpha |Vocabulary|}$

# Over the next couple of classes, we'll investigate:

1. How do we "digest" text into a form usable by a function?

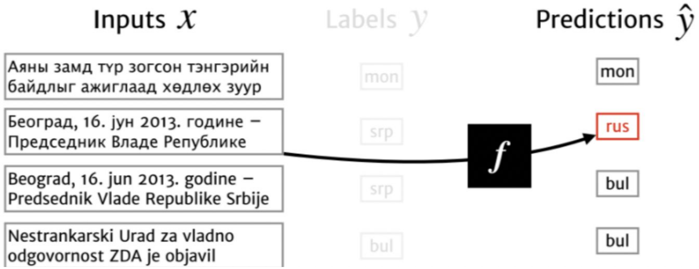
   (Keywords for this section: features, feature extraction, feature selection, representations)

2. What kinds of strategies might we use to create our function *f*?

   (Keyword for this section: models)

3. How do we evaluate our function *f*?

   (Keyword for this section: … evaluation)



Аяны замд түр зогсон тэнгэрийн байдлыг ажиглаад хөдлөх зуур

Београд, 16. јун 2013. године – Председник Владе Републике

Beograd, 16. jun 2013. godine – Predsednik Vlade Republike Srbije

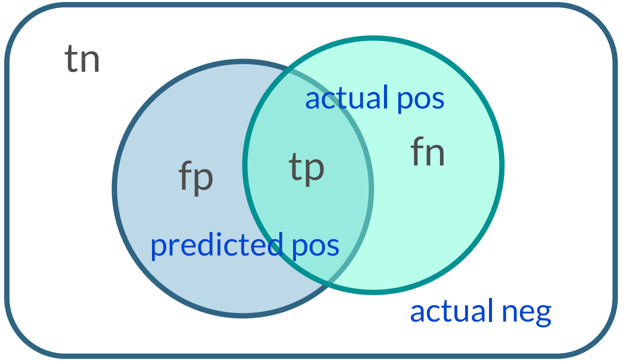
Nestrankarski Urad za vladno odgovornost ZDA je objavil

# How do we evaluate our function *f*?

# Classification evaluation

- Contingency table: model's predictions are compared to the correct results
  - a.k.a. confusion matrix

| | actual pos | actual neg |
|---|---|---|
| predicted pos | true positive (tp) | false positive (fp) |
| predicted neg | false negative (fn) | true negative (tn) |



# Classification evaluation

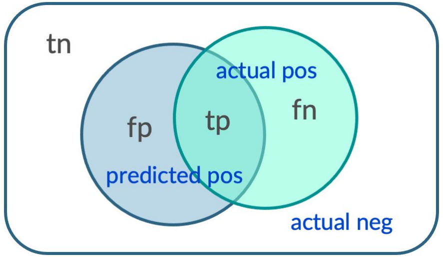
- Borrowing from Information Retrieval, empirical NLP systems are usually evaluated using the notions of precision and recall

# Classification evaluation

- Precision (P) is the proportion of the selected items that the system got right in the case of text categorization
- Important when minimizing false positive (classifying a non-spam as spam)
- Example: In a spam email classifier, precision tells you how many of the emails flagged as spam were actually spam.

$$\text{precision} = \frac{true\ positives}{true\ positives + false\ positives} = \frac{tp}{tp + fp}$$

| | actual pos | actual neg |
|---|---|---|
| predicted pos | true positive (tp) | false positive (fp) |
| predicted neg | false negative (fn) | true negative (tn) |

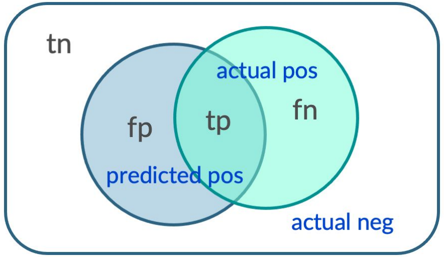


# Classification evaluation

- Recall (R) is the proportion of actual items that the system selected in the case of text categorization
- Important when minimizing false negatives. E.g. medical diagnoses, abusive text
- Example: failing to identify a disease is more harmful than a false alarm

$$\text{recall} = \frac{true\ positives}{true\ positives + false\ negatives} = \frac{tp}{tp + fn}$$

| | actual pos | actual neg |
|---|---|---|
| predicted pos | true positive (tp) | false positive (fp) |
| predicted neg | false negative (fn) | true negative (tn) |



# Classification evaluation

- We often want to trade-off precision and recall
  - typically: the higher the precision the lower the recall
  - can be plotted in a precision-recall curve
- It is convenient to combine P and R into a single measure
  - one possible way to do that is F1 measure
- Harmonic mean of precision and recall
- Useful when you want to balance precision and recall

$$F_\beta = \frac{(\beta^2+1)PR}{\beta^2 P+R} \qquad \text{for } \beta=1,\ F_1 = \frac{2PR}{P+R}$$

# Classification evaluation

- Additional measures of performance: accuracy and error
  - accuracy is the proportion of items the system got right
  - error is its complement

$$\text{accuracy} = \frac{tp+tn}{tp+fp+tn+fn}$$

| | actual pos | actual neg |
|---|---|---|
| predicted pos | true positive (tp) | false positive (fp) |
| predicted neg | false negative (fn) | true negative (tn) |

# Micro- vs. macro-averaging

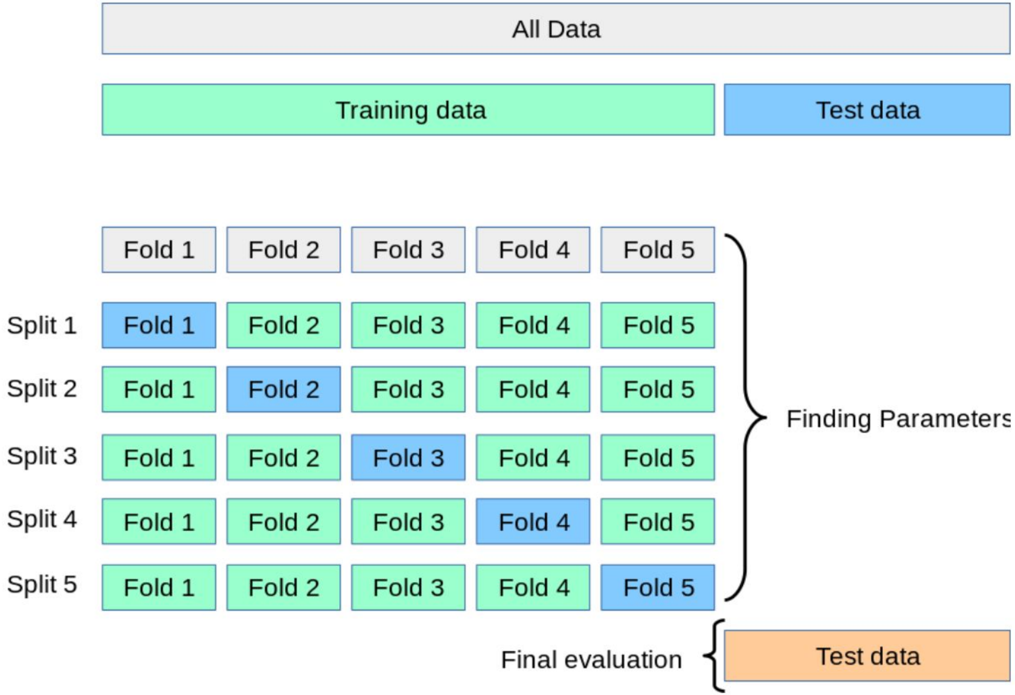
If we have more than one class, how do we combine multiple performance measures into one quantity?

- Macroaveraging
  - Compute performance for each class, then average.
- Microaveraging
  - Collect decisions for all classes, compute contingency table, evaluate.

# Classification common practices

- Divide the training data into k folds (e.g., k=10)
- Repeat k times: train on k-1 folds and test on the holdout fold, cyclically
- Average over the k folds’ results

# K-fold cross-validation



All Data

Training data | Test data

| | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 |
|---|---|---|---|---|---|
| Split 1 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 |
| Split 2 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 |
| Split 3 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 |
| Split 4 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 |
| Split 5 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 |

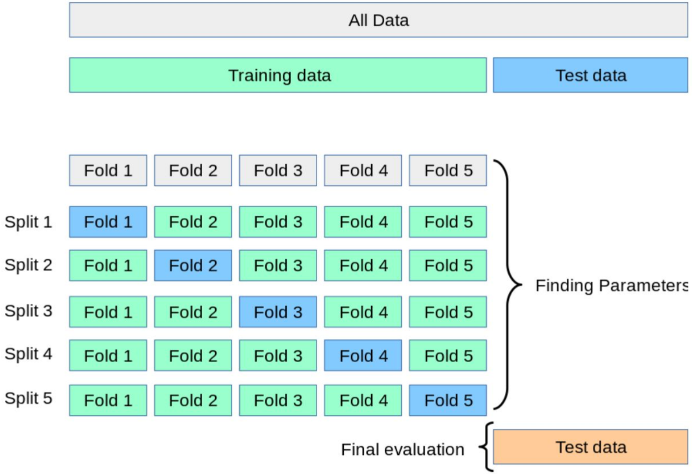
Finding Parameters

Final evaluation

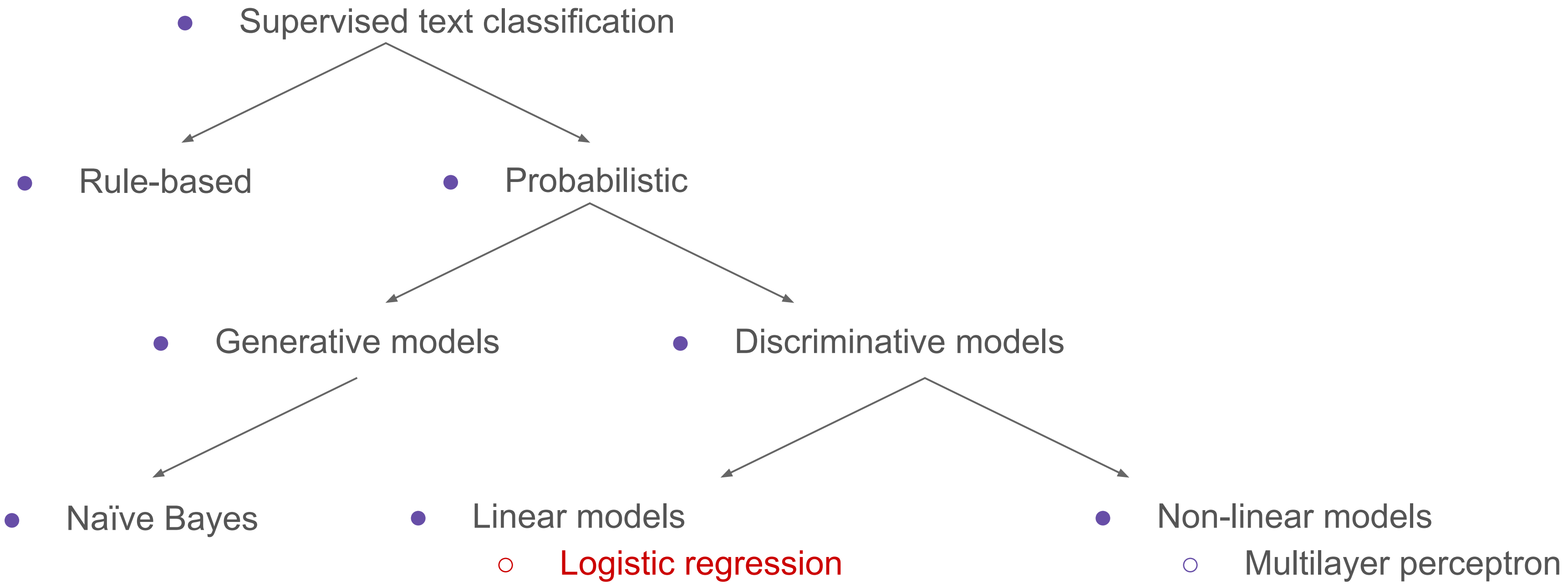
Test data

# K-fold cross-validation

- Metric: P/R/F1 or Accuracy
- Unseen test set
  - avoid overfitting ('tuning to the test set')
  - more conservative estimate of performance
- Cross-validation over multiple splits
    - Handles sampling errors from different datasets
  - Pool results over each split
  - Compute pooled dev set performance



# Next class: Logistic regression



- Supervised text classification
- Rule-based
- Probabilistic
- Generative models
- Discriminative models
- Naïve Bayes
- Linear models
  - Logistic regression
- Non-linear models
  - Multilayer perceptron