Natural Language Processing

Text classification, Methodology, Logistic Regression

Yulia Tsvetkov

yuliats@cs.washington.edu

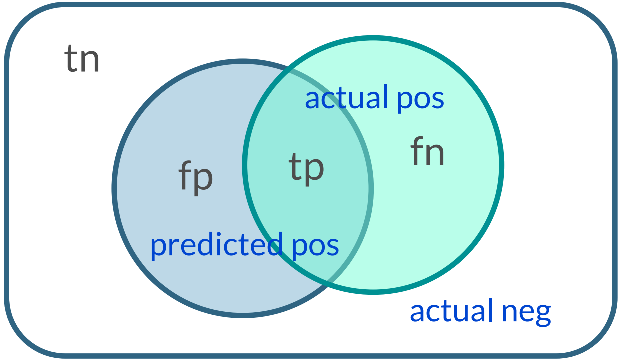
PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

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)



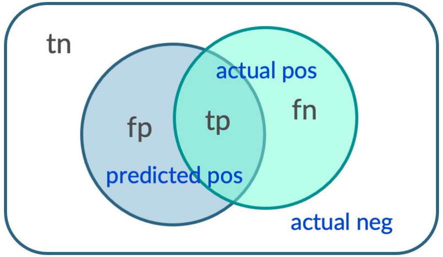
Classification evaluation

- 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
  - it is the % of documents classified as “positive” by the system which are indeed “positive” documents
- Reported per class or average

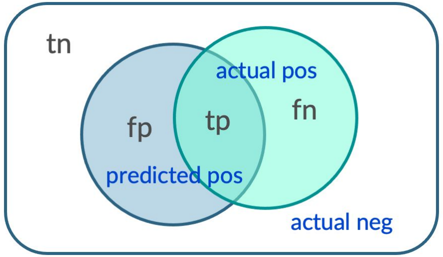
$$\text{precision} = \frac{true\ positives}{true\ positives + false\ positives} = \frac{tp}{tp + fp}$$



Classification evaluation

- Recall (R) is the proportion of actual items that the system selected in the case of text categorization
  - it is the % of the “positive” documents which were actually classified as “positive” by the system
- Reported per class or average

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



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 F measure

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

Classification evaluation

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

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

Micro- vs. macro-averaging

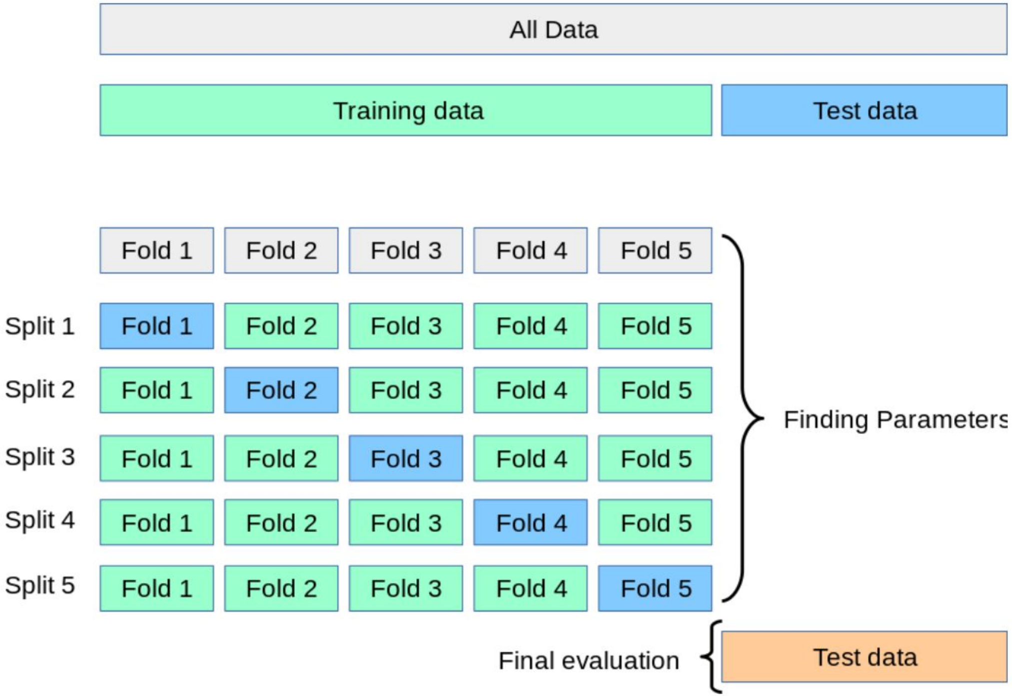
If we have more than one class, how do we combine multiple performance measures into one quantity?

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

Classification common practices

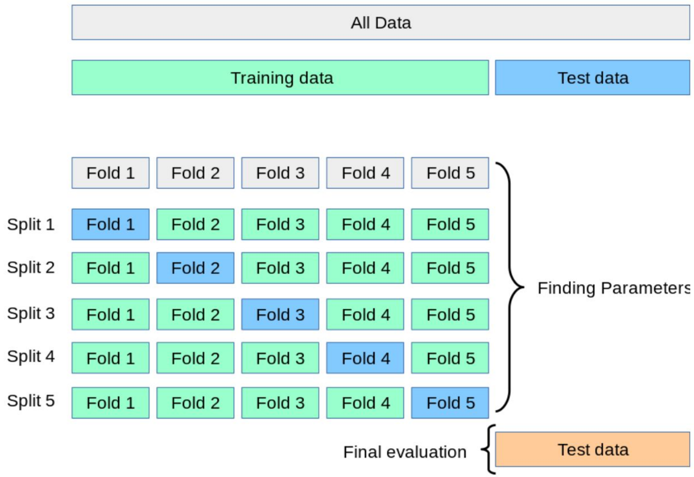
- Divide the training data into k folds (e.g., $k=10$)
- Repeat k times: train on $k-1$ folds and test on the holdout fold, cyclically
- Average over the k folds' results

K-fold cross-validation



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
      - Naïve Bayes
    - Discriminative models
      - Linear models
        - Multinomial logistic regression (aka MaxEnt)
      - Non-linear models
        - Multilayer perceptron

Logistic regression classifier

- Important analytic tool in natural and social sciences
- Baseline supervised machine learning tool for classification
- Is also the foundation of neural networks

Next class: Logistic regression

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

Readings

- J&M Chapter 5 https://web.stanford.edu/~jurafsky/slp3/5.pdf

Logistic regression classifier

- Important analytic tool in natural and social sciences
- Baseline supervised machine learning tool for classification
- Is also the foundation of neural networks

Text classification

Input:

- a document d (e.g., a movie review)
- a fixed set of classes $C = \{c_1, c_2, \dots c_j\}$ (e.g., positive, negative, neutral)

Output

- a predicted class $\hat{y} \in C$

Binary classification in logistic regression

- Given a series of input/output pairs:
  - $(x^{(i)}, y^{(i)})$

- For each observation $x^{(i)}$
  - We represent $x^{(i)}$ by a feature vector $\{x_1, x_2, \ldots, x_n\}$
  - We compute an output: a predicted class $\hat{y}^{(i)} \in \{0,1\}$

Features in logistic regression

- For feature $x_i \in \{x_1, x_2, \ldots, x_n\}$, weight $w_i \in \{w_1, w_2, \ldots, w_n\}$ tells us how important is x_i
  - x_i = "review contains 'awesome'": w_i = +10
  - x_j = "review contains horrible": w_j = -10
  - x_k = "review contains 'mediocre'": w_k = -2

Logistic Regression for one observation x

- Input observation: vector $x^{(i)} = \{x_1, x_2, \ldots, x_n\}$
- Weights: one per feature: $W = [w_1, w_2, \ldots, w_n]$
  - Sometimes we call the weights $\theta = [\theta_1, \theta_2, \ldots, \theta_n]$
- Output: a predicted class $\hat{y}^{(i)} \in \{0,1\}$

multinomial logistic regression: $\hat{y}^{(i)} \in \{0,1, 2, 3, 4\}$

How to do classification

- For each feature x_i, weight w_i tells us importance of x_i
  - (Plus we'll have a bias b)
  - We'll sum up all the weighted features and the bias

$$z = \left(\sum_{i=1}^{n} w_i x_i\right) + b$$

$$z = w \cdot x + b$$

If this sum is high, we say $y=1$; if low, then $y=0$

But we want a probabilistic classifier

We need to formalize “sum is high”

- We’d like a principled classifier that gives us a probability, just like Naive Bayes did
- We want a model that can tell us:
  - $p(y=1|x;\theta)$
  - $p(y=0|x;\theta)$

The problem: z isn't a probability, it's just a number!

- z ranges from $-\infty$ to ∞

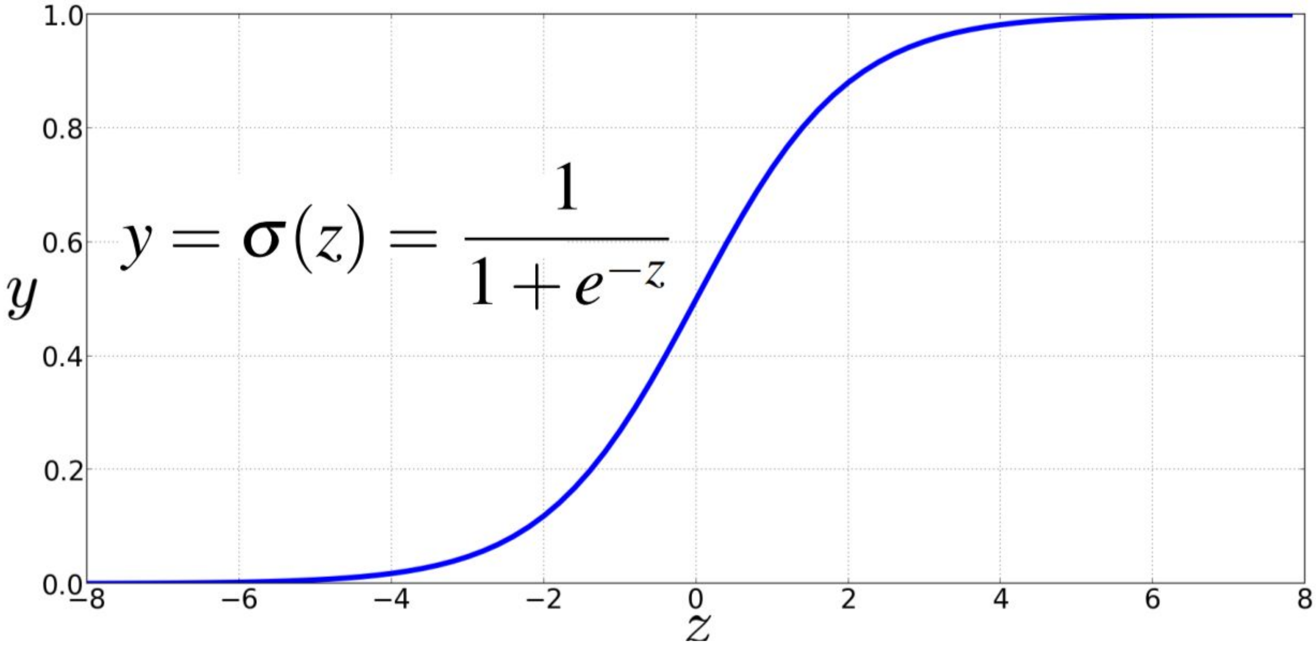
$$z = w \cdot x + b$$

- Solution: use a function of z that goes from 0 to 1

"sigmoid" or "logistic" function

$$y = \sigma(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+\exp(-z)}$$

The very useful sigmoid or logistic function



Idea of logistic regression

- We'll compute w·x+b
- And then we'll pass it through the sigmoid function:

$$\sigma(w \cdot x + b)$$

- And we'll just treat it as a probability

Making probabilities with sigmoids

$$
\begin{aligned}
P(y=1) &= \sigma(w \cdot x + b) \\
&= \frac{1}{1+\exp(-(w \cdot x + b))}
\end{aligned}
$$

Making probabilities with sigmoids

$$\begin{aligned} P(y=1) &= \sigma(w \cdot x + b) \\ &= \frac{1}{1+\exp(-(w \cdot x + b))} \end{aligned}$$

$$\begin{aligned} P(y=0) &= 1 - \sigma(w \cdot x + b) \\ &= 1 - \frac{1}{1+\exp(-(w \cdot x + b))} \\ &= \frac{\exp(-(w \cdot x + b))}{1+\exp(-(w \cdot x + b))} \end{aligned}$$

By the way:

$$
\begin{aligned}
P(y=0) &= 1-\sigma(w\cdot x+b) \\
&= 1-\frac{1}{1+\exp(-(w\cdot x+b))} \\
&= \frac{\exp(-(w\cdot x+b))}{1+\exp(-(w\cdot x+b))}
\end{aligned}
$$

$$= \sigma(-(w\cdot x+b))$$

Because

$$1-\sigma(x)=\sigma(-x)$$

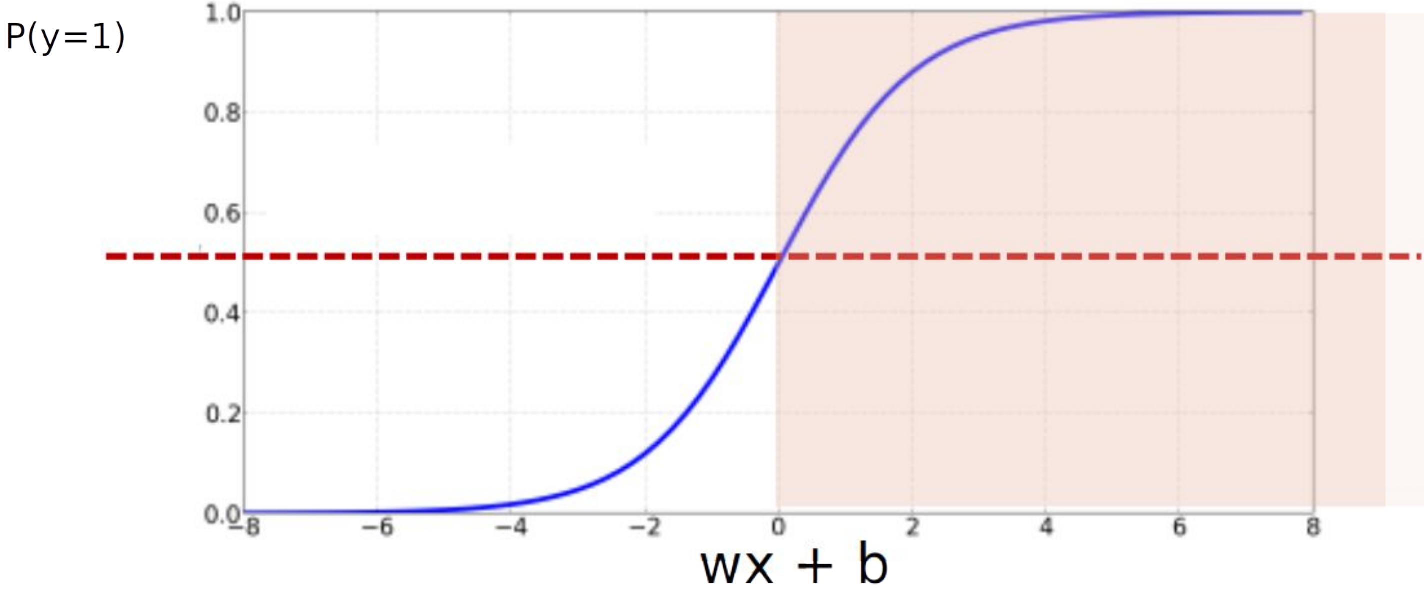
Turning a probability into a classifier

$$\hat{y} = \begin{cases} 1 & \text{if } P(y=1|x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

- 0.5 here is called the **decision boundary**

The probabilistic classifier

$$P(y=1) = \sigma(w \cdot x + b) = \frac{1}{1+\exp(-(w \cdot x + b))}$$



Turning a probability into a classifier

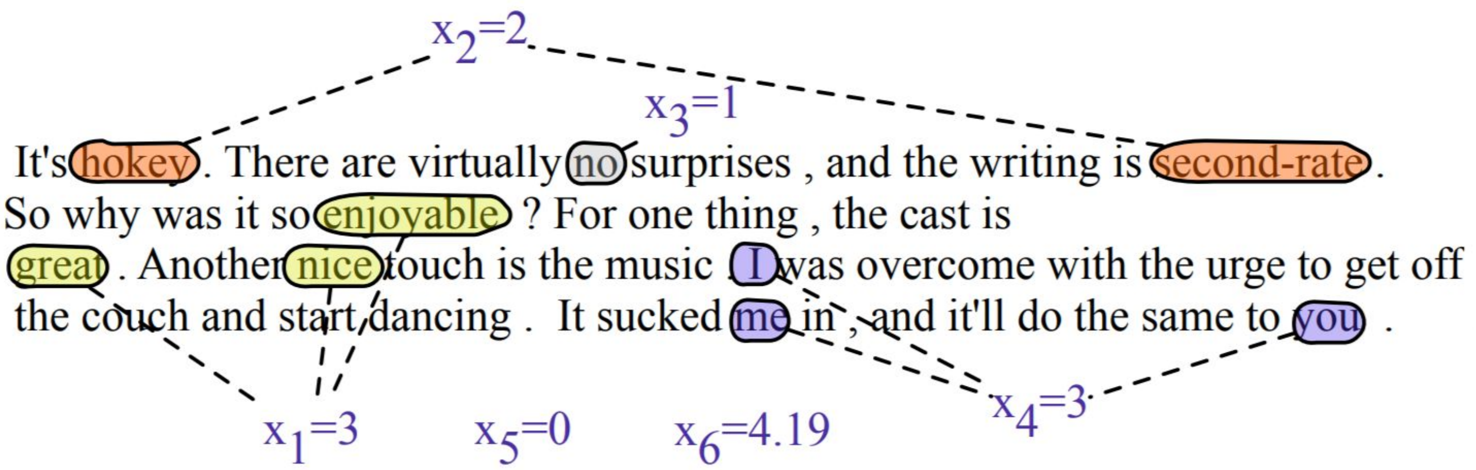
$$\hat{y} = \begin{cases} 1 & \text{if } P(y=1|x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

if $w \cdot x + b > 0$

if $w \cdot x + b \leq 0$

Sentiment example: does y=1 or y=0?

It's hokey . There are virtually no surprises , and the writing is second-rate . So why was it so enjoyable ? For one thing , the cast is great . Another nice touch is the music . I was overcome with the urge to get off the couch and start dancing . It sucked me in , and it'll do the same to you .



Var	Definition	Value
x_1	count(positive lexicon) $\in$ doc)	3
x_2	count(negative lexicon) $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	log(word count of doc)	$\ln(66) = 4.19$

Classifying sentiment for input x

Var	Definition	Value
x_1	count(positive lexicon) $\in$ doc)	3
x_2	count(negative lexicon) $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	log(word count of doc)	$\ln(66) = 4.19$

Suppose w = [2.5, -5.0, -1.2, 0.5, 2.0, 0.7]

b = 0.1

Classifying sentiment for input x

$$
\begin{aligned}
p(+|x) = P(Y=1|x) &= \sigma(w \cdot x + b) \\
&= \sigma([2.5, -5.0, -1.2, 0.5, 2.0, 0.7] \cdot [3, 2, 1, 3, 0, 4.19] + 0.1) \\
&= \sigma(.833) \\
&= 0.70
\end{aligned}
$$

$$
\begin{aligned}
p(-|x) = P(Y=0|x) &= 1 - \sigma(w \cdot x + b) \\
&= 0.30
\end{aligned}
$$

Scaling input features

- z-score

$$\mu_i = \frac{1}{m}\sum_{j=1}^{m} x_i^{(j)} \qquad \sigma_i = \sqrt{\frac{1}{m}\sum_{j=1}^{m}\left(\mathbf{x}_i^{(j)} - \mu_i\right)}$$

$$\mathbf{x}_i' = \frac{\mathbf{x}_i - \mu_i}{\sigma_i}$$

- normalize

$$\mathbf{x}_i' = \frac{\mathbf{x}_i - \min(\mathbf{x}_i)}{\max(\mathbf{x}_i) - \min(\mathbf{x}_i)}$$

Wait, where did the W's come from?

- Supervised classification:
  - At training time we know the correct label y (either 0 or 1) for each x.
  - But what the system produces at inference time is an estimate $\hat{y}$

Wait, where did the W's come from?

- Supervised classification:
  - A training time we know the correct label y (either 0 or 1) for each x.
  - But what the system produces at inference time is an estimate $\hat{y}$

- We want to set w and b to minimize the **distance** between our estimate $\hat{y}^{(i)}$ and the true $y^{(i)}$
  - We need a distance estimator: a **loss function** or a cost function
  - We need an **optimization algorithm** to update w and b to minimize the loss

Learning components in LR

A loss function:

- **cross-entropy loss**

An optimization algorithm:

- **stochastic gradient descent**

Loss function: the distance between ŷ and y

We want to know how far is the classifier output $\hat{y} = \sigma(w \cdot x + b)$

from the true output: y [= either 0 or 1]

We'll call this difference: $L(\hat{y}, y)$ = how much ŷ differs from the true y

Intuition of negative log likelihood loss (NLL) = cross-entropy loss

A case of conditional maximum likelihood estimation

We choose the parameters w,b that maximize

- the log probability
- of the true y labels in the training data
- given the observations x

Next class:

- Deriving cross-entropy loss (please review Bernoulli distribution before class)
- Stochastic gradient descent
- Softmax