Text Classification

CS4742 Natural Language Processing
Lecture 02

Jiho Noh

Department of Computer Science
Kennesaw State University

CS4742 Summer 2025 [1]

KENNESAW STATE UNIVERSITY
COLLEGE OF COMPUTING AND SOFTWARE ENGINEERING

[1] This lecture is based on the slides from Dr. Hafiz Khan at KSU.

1. **Introduction to Classification**

2. Naïve Bayes Classifier

3. Logistic Regression Classifier

Examples of Text Classification

- **Review classification:** Decide if a review is positive or negative
- **Movie classification:** Decide if a movie is a comedy, action, or horror
- **Sentiment Classification:** Decide if a review is positive or negative
- **Spam Detection:** Decide if an email is spam or not
- **Authorship:** Determine the author of a given text based on writing style, linguistic features, etc.
- **Language classification:** Determine the language of a text (e.g., English vs. Portuguese)

Review Classification

Positive vs Negative Review (Binary Classification)

Examples:

- positive *...characters and richly applied satire, and some great plot twists*
- negative *It was pathetic. The worst part about it was the boxing scenes...*
- positive *...awesome caramel sauce and sweet toasty almonds. I love this place!*
- negative *...awful pizza and ridiculously overpriced...*

Movie Classification

- Genre classification
  - Horror Movies
  - Comedy Movies
  - Action Movies
  - Animation Movies
- Datasets and Examples
  - MPST: Movie Plot Synopses with Tags
  - IMBD dataset
  - A multimodal approach for multi-label movie genre classification

Text Classification: Machine Learning

ML classifiers

- A generic (task-independent) learning algorithm to train a classifier from a set of labeled examples.
- From the labeled examples, the classifier learns the patterns and features that distinguish different classes.

Advantages

- ML classifiers can be applied to a wide range of text classification tasks. With a new set of labeled examples, the same classifier can be trained to classify different types of text.
- ML classifiers can handle large amounts of data and can learn complex patterns in the data.

Text Classification

- **Input:**
  - A document d
  - A fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$
  - A training set of m hand-labeled documents $(d_1, c_1), \ldots, (d_m, c_m)$
- **Output:**
  - A predicted class $c \in C$
  - A learned classifier $\gamma : d \to c$

Text Classification – Formal Definition

- Training data representation using symbols

$$D = (X_1, y_1), (X_2, y_2), \ldots, (X_N, y_N)$$

- Learns a model/function f such that it can infer class label y.
- Input is given as text
  - Convert raw text to features $f : X \to y \in Y$
- Probabilistic settings:
  - This function is similar as $P(y|X)$.
  - Label predicted as

$$\hat{y} = \arg\max_{y} P(y|X)$$

Types of Classification

- Number of classes:
  - Binary (e.g., spam or not)
  - Multi-class (e.g., age: child, teen, adult)
- Number of labels:
  - Single-label (e.g., spam or not)
  - Multi-label (e.g., news AND business AND technology)
- Output type:
  - Hard classification (e.g., spam or not, with a binary label)
  - Soft classification (e.g., spam or not, with a probability score)

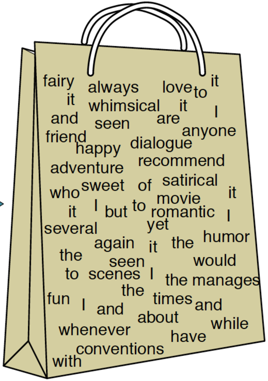
Classifier: Any ML classifier

- **Naïve Bayes**: Based on Bayes theorem and the assumption of independence between features
- **Logistic regression**: Uses a logistic function to model the probability of a binary outcome
- **Support-vector machines**: Find the optimal hyperplane that separates classes in a high-dimensional space
- **k-Nearest Neighbors**: Decides the class of a sample based on the classes of its k nearest neighbors
- **Decision trees**: Builds a tree-like model of decisions based on feature values
- **Random forests**: An ensemble of decision trees, where each tree is trained on a random subset of the data
- **Neural networks**: Deep learning models that can learn complex patterns in data

Naïve Bayes Classifier

- Relies on very simple representation of document
  - **Bag of words**

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
…	…

Bag of Words Example

Example

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.

Classification $\hat{y}$: $y =$ "positive" or "negative"

Bag of Words Example

Considering only a subset of words

Example

x **love** xxxx xxxx xxxxxx **sweet** xxx xxx **satirical** xxxxxx xxx xxxxx xxxxx **great** xxxxxxx xxxxxxxxx xxxxxxxxx **fun** xxxx xxx xxxxx xxx **whimsical** xxxx **romantic** xxxx **laughing** xxxxxx xxxxxx xxxxxxxxxxxxxx xxxxxx xxxxxxxx **recommend** xxxxx xxxx xxxxxxxx xxxxxxxxxxxx xxxxx **several** xx xxxx xxxxxxx x **happy** xxx xxxxx **again** xxxxx xxxxxxxxxx xxxxxx xxxx x xxxx xxxxx xxxx

Classification $\hat{y}$: $y =$ “positive” or “negative”

Estimating $P(c|d)$ or $P(d|c)$?

- Here C presents class and d represents documents.
- We compute conditional probability of the class given the document $P(c|d)$.
- It is difficult to estimate $P(c|d)$ directly. Why?
  - Because we need to estimate the joint probability $P(c, d)$ of all features, which becomes computationally intractable as feature dimensions grow.

Bayes Rules

- We can represent document d with features (or words) $(x_1, x_2 \ldots, x_n)$
- With a length of k document and n possible words (features):
  - Explicitly modeling $P(c|d)$ requires estimating n^k joint probabilities.
  - On the other hand, modeling $P(d|c)$ can be estimated using $\prod_{i=1}^{n} P(w_i|c)$ with the naïve assumption that features are conditionally independent given the class.
- The inversion of conditional probabilities can reduce the complexity of estimation significantly. We can use Bayes rule to compute $P(c|d)$ in terms of $P(d|c)$, $P(c)$, and $P(d)$, which are easier to estimate.

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

Naïve Bayes Classifier

Maximum A-Posteriori (MAP) estimation:

$$
\begin{aligned}
P(c|d) &= \frac{P(d|c)P(c)}{P(d)} \\
&\propto P(d|c)P(c) \\
&\propto P(c)\prod_{i=1}^{n} P(w_i|c)
\end{aligned}
$$

- Simplifying the equation by dropping denominator? Why?
- $\Rightarrow$ Because for all class, the denominator should be same.

For inference, we need to compute the class c that maximizes the posterior probability $P(c|d)$.

$$
\hat{c} = \arg\max_{c} P(c)\prod_{i=1}^{n} P(w_i|c)
$$

Multinomial Naïve Bayes

$$P(c|d) \propto P(c) \prod_{i=1}^{n} P(w_i|c)$$

- **Bag of Words** representations:
  - assume that position (or order) of the features doesn't matter in classification.
- **Naïve Bayes** assumptions:
  - assumes that the features are conditionally independent given the class.

Naïve Bayes: Text Classification

Again, we want to find the class c_j that maximizes the posterior probability $P(c_j|d)$ given a document d.

$$c_{NB} = \arg\max_{c_j \in C} P(c_j) \prod_{i=1}^{n} P(w_i|c_j)$$

To avoid underflow and increase speed, we apply log scale

$$c_{NB} = \arg\max_{c_j \in C} \log(P(c_j)) + \sum_{i=1}^{n} \log(P(w_i|c_j))$$

Estimating Probabilities

- $P(c_j)$? $P(w_i|c_j)$?
- First attempt: **maximum likelihood estimates**
  - Simply use the frequencies in the data

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

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

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

That is, fraction of times word w_i appears among all words in documents of topic c_j

Discussion

Any problems with this estimation?

$$\hat{c} = \arg\max_{c} P(c) \prod_{i=1}^{n} P(w_i|c),$$

where

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

Problems with Estimation

- What if we have seen no training documents with the word *fantastic* and classified in the topic *positive* (thumbs-up)?

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

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

$$\prod_{i=1}^{n} P(w_i|c_j) = 0$$

- The product of conditional probabilities is always 0, if one of the probabilities is 0.

A Solution to Zero Probabilities

Laplace (add-1) smoothing

$$P(w_j|c) = \frac{\text{count}(w_j, c) + 1}{\sum \text{count}(w, c) + 1} = \frac{\text{count}(w_j, c) + 1}{N(c) + |V|}$$

where $N(c)$ is the number of words in class c and $|V|$ is the size of the vocabulary.

Another Issue:

- Unknown words in the test data.
  - Vocabulary did not occur in the train data in any class.
  - Solution: Remove them or ignore them while computing probability.
- More frequent (and non-informative) words – especially stop words
  - Best practice to remove them

Naïve Bayes: Learning

1. From training corpus, build a vocabulary.
2. Calculate $\hat{P}(c_j)$ terms:
   - For each c_j in C Do
     - $D_j \leftarrow$ all docs with class c_j
     - $\hat{P}(c_j) \leftarrow |D_j|/N$
3. Calculate $\hat{P}(w_i|c_j)$ terms:
   - $T_j \leftarrow$ single doc containing all docs in D_j
   - For each word w_i in the vocabulary,
     - $n_i \leftarrow$ # of occurrences of w_i in T_j
     - $\hat{P}(w_i|c_j) \leftarrow (N_i + \alpha)/(n + \alpha|V|)$

$$\hat{P}(c) = \frac{N_c}{N}$$

$$\hat{P}(w \mid c) = \frac{count(w,c)+1}{count(c)+|V|}$$

	Doc	Words	Class
Training	1	Chinese Beijing Chinese	c
	2	Chinese Chinese Shanghai	c
	3	Chinese Macao	c
	4	Tokyo Japan Chinese	j
Test	5	Chinese Chinese Chinese Tokyo Japan	?

Priors:

$P(c) = \frac{3}{4}$

$P(j) = \frac{1}{4}$

Choosing a class:

P(c|d5) $\propto$ 3/4 * (3/7)^3 * 1/14 * 1/14

$\approx$ 0.0003

Conditional Probabilities:

P(Chinese|*c*) = (5+1) / (8+6) = 6/14 = 3/7

P(Tokyo|*c*) = (0+1) / (8+6) = 1/14

P(Japan|*c*) = (0+1) / (8+6) = 1/14

P(Chinese|*j*) = (1+1) / (3+6) = 2/9

P(Tokyo|*j*) = (1+1) / (3+6) = 2/9

P(Japan|*j*) = (1+1) / (3+6) = 2/9

P(j|d5) $\propto$ 1/4 * (2/9)^3 * 2/9 * 2/9

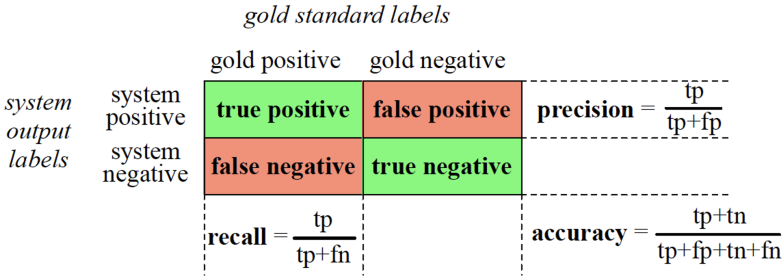
$\approx$ 0.0001

Naïve Bayes Summary

- Very Fast, low storage requirements
- Robust to Irrelevant Features
  - Irrelevant Features cancel each other without affecting results
- Very good in domains with many equally important features
- Optimal if the independence assumptions hold: If assumed independence is correct, then it is the Bayes Optimal Classifier for problem
- A good **dependable baseline** for text classification

Evaluation Metrics and Confusion Matrix

- **Performance metrics** - Precision, Recall, F measure
- **Precision** - the fraction of relevant instances among the retrieved instances
- **Recall** - fraction of relevant instances that were retrieved.



F-1 Score

- Precision: $P = \frac{TP}{TP+FP}$, Recall: $R = \frac{TP}{TP+FN}$

$$F_\beta = \frac{(1+\beta^2) \cdot P \cdot R}{\beta^2 \cdot P + R}$$

- The β parameter differentially weights the importance of recall and precision:
  - Values of $\beta > 1$ favor recall
  - Values of $\beta < 1$ favor precision
  - When $\beta = 1$, precision and recall are equally balanced, where F-1 score becomes the harmonic mean of precision and recall.

$$F_1 = \frac{2 \cdot P \cdot R}{P + R} = \frac{2}{\left(\frac{1}{P} + \frac{1}{R}\right)}$$

Confusion Matrix with multiple classes

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

gold labels

system output		urgent	normal	spam	
	urgent	8	10	1	**precision**u$=\frac{8}{8+10+1}$
	normal	5	60	50	**precision**n$=\frac{60}{5+60+50}$
	spam	3	30	200	**precision**s$=\frac{200}{3+30+200}$
		recallu $=\frac{8}{8+5+3}$	**recall**n $=\frac{60}{10+60+30}$	**recall**s $=\frac{200}{1+50+200}$	

Micro- vs. Macro-Averaging: Example

- **Macro averaging:** Compute performance for each class, then average.
- **Micro averaging:** Collect decisions for all classes, compute contingency table, evaluate.

Class 1: Urgent

	true urgent	true not
system urgent	8	11
system not	8	340

$$\text{precision} = \frac{8}{8+11} = .42$$

Class 2: Normal

	true normal	true not
system normal	60	55
system not	40	212

$$\text{precision} = \frac{60}{60+55} = .52$$

Class 3: Spam

	true spam	true not
system spam	200	33
system not	51	83

$$\text{precision} = \frac{200}{200+33} = .86$$

Pooled

	true yes	true no
system yes	268	99
system no	99	635

$$\text{microaverage precision} = \frac{268}{268+99} = .73$$

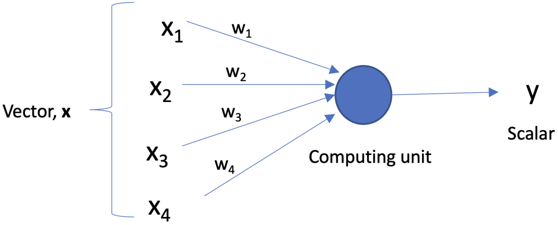
$$\text{macroaverage precision} = \frac{.42+.52+.86}{3} = .60$$

Regression, Linear Regression, Logistic Regression

- **Regression** is a statistical method for estimating the relationships among variables. The output is a continuous value, not a class label.
- **Linear regression** is a type of regression that uses a linear function to model the relationship between the input features and the output variable.
- **Logistic regression** is a type of regression that uses a logistic function with a linear regression to model the probability of a binary outcome. It is used for classification tasks, where the output is a class label.

Linear Regression

- During training:
  - Assigns weights (w) to features and also adds a bias term (intercept).
  - Updates weights using the gradient descent algorithm.
- Decision during test:
  - Learned weight multiplied with the feature to compute score.



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

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

dot product form.

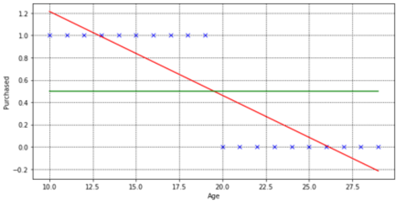
Linear regression is not for classification

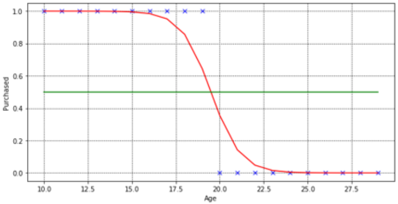
Linear regression for binary classification

- Encode class labels as $y = \{0, 1\}$ or $\{-1, 1\}$
- Apply Linear Regression
- check which class the prediction is closer to. If class 1 is encoded to 1 and class 2 is -1.
  - class 1 if $f(x) \geq 0$
  - class 2 if $f(x) < 0$
- Linear models are NOT optimized for classification

Linear regression is not for classification

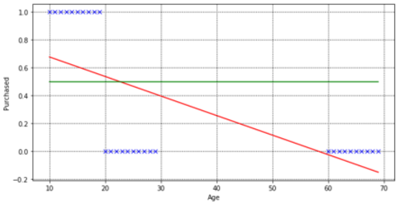
- The predicted value by a linear regression classifier is continuous, not probabilistic

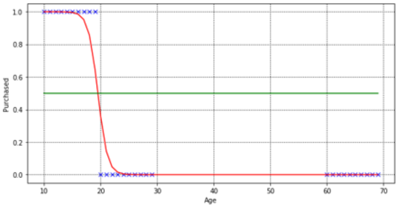




Linear regression is not for classification

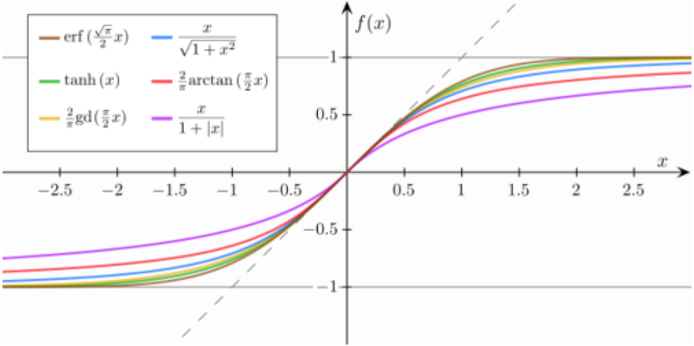
- Sensitive to imbalanced data





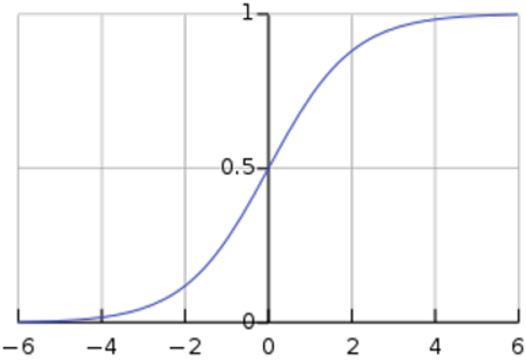
Probabilistic Approach

- Learn $P(Y|X)$ directly
- Cumulative probability distribution
- Using a sigmoid function
- It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits



Logistic Function

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}},$$

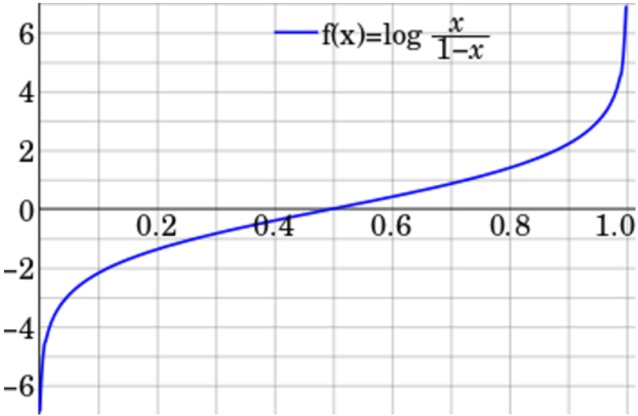


- x_0, the x value of the function's midpoint
- L, the curve's maximum value
- k, the logistic growth rate or steepness of the curve
- e, natural logarithm base

Standard logistic function where $L = 1, k = 1, x_0 = 0$

Logit function

- The inverse of the logit function is the logistic function.



Representation used for Logistic Regression

- Logistic regression uses an equation as the representation, very much like linear regression.
- Below is an example **logistic regression equation**:

$$\forall z \in \mathbb{R}, \quad g(z) = \frac{1}{1 + e^{-z}} \in [0, 1]$$

- where $g(z)$ is the predicted output in terms of probability, z **is a linear regression output**.
- $g(\cdot)$ is a non-linear transformation (in this case, using logistic function).

Making Predictions using Logistic Regression

- For example, Given a height of 150cm, is the person male or female?
- Assume that we have learned the coefficients of $w_0 = -100$ and $w_1 = 0.6$. Using the equation above we can calculate the probability of male given a height of 150cm or more formally $P(male|height = 150)$.

$$y = 1/(1 + e^{-(w_0 + w_1 X)})$$

$$y = 1/(1 + exp(-(-100 + 0.6 * X)))$$

$$y = 0.0000453978687$$

- Or a probability of near zero that the person is a male.

Making Predictions using Logistic Regression

- In practice we can use the probabilities directly. We can convert the probabilities into binary class values, for instance
  - 0 if $p(male) < 0.5$
  - 1 if $p(male) \geq 0.5$
- Now that we know how to make predictions using logistic regression.

Softmax regression

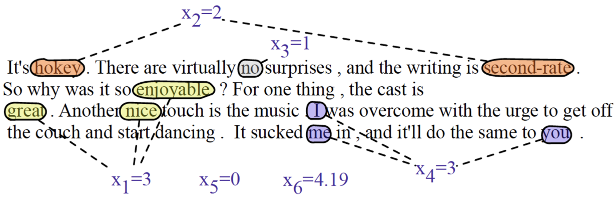
- A softmax regression, also called a multiclass logistic regression, is used to generalize logistic regression where there are more than 2 outcome classes.
- By convention, we set $\theta_K = 0$ the regression based on the Bernoulli distribution.

$$\phi_i = \frac{exp(\theta_i^T x)}{\sum_{j=1}^{K} exp(\theta_j^T x)}$$

Example - Sentiment Analysis

- Binary classification (sentiment) on moview review text
- We'll represent each input observation by the 6 features ($x_1 \ldots x_6$) of the input shown in the following table:

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)



Example: Compute Probability

Given real valued weight $W = [2.5, 5.0, 1.2, 0.5, 2.0, 0.7]$, and bias, $b = 0.1$.

$$
\begin{aligned}
p(+ \mid x) = P(Y = 1 \mid x) &= \sigma(w \cdot x + b) \\
&= \sigma([2.5, -5.0, -1.2, 0.5, 2.0, 0.7] \cdot [3, 2, 1, 3, 0, 4.19] + 0.1) \\
&= \sigma(.833) \\
&= 0.70 \\
p(- \mid x) = P(Y = 0 \mid x) &= 1 - \sigma(w \cdot x + b) = 0.30
\end{aligned}
$$

Assign the class which has the highest probability.

Naïve Bayes vs. Logistic Regression

- Naïve Bayes (NB) assumes conditional independences
  - Consider two features – strongly correlated
  - NB treats separately and multiplies them – overestimate the evidence.
- **Logistic regression more robust**
  - Assigns part of the weight to each feature.
  - Logistic regression works better on larger documents or datasets
  - **No conditional assumption!**

Cost Function for Classifier Models

- We need to measure how close the classifier output is to the correct output
- $L(y', y) =$ how much y' (prediction) differs from the true y
- As output follows Bernoulli distribution

$$p(y \mid x) = \hat{y}^{y}(1-\hat{y})^{1-y}$$

- We want to maximize the probability of the correct class by minimizing the negative log likelihood. Converting in log scale:

$$\log p(y \mid x) = \log\left[\hat{y}^{y}(1-\hat{y})^{1-y}\right] = y\log\hat{y} + (1-y)\log(1-\hat{y})$$

$$L_{\mathrm{CE}}(\hat{y}, y) = -\log p(y \mid x) = -[y\log\hat{y} + (1-y)\log(1-\hat{y})]$$
$$L_{\mathrm{CE}}(\hat{y}, y) = -[y\log\sigma(w\cdot x+b) + (1-y)\log(1-\sigma(w\cdot x+b))]$$

Demo: News Article Classification Using NB Classifier and Logistic Regression

- Naive Bayes, Logistic Regression, SVM using Scikit-Learn in Python