Text Classification

Dr. Nguyen Van Vinh

CS Department – UET, Hanoi VNU

Is this spam?

Subject: **Important notice!**
From: Stanford University <newsforum@stanford.edu>
Date: October 28, 2011 12:34:16 PM PDT
To: undisclosed-recipients:;

Greats News!

You can now access the latest news by using the link below to login to Stanford University News Forum.

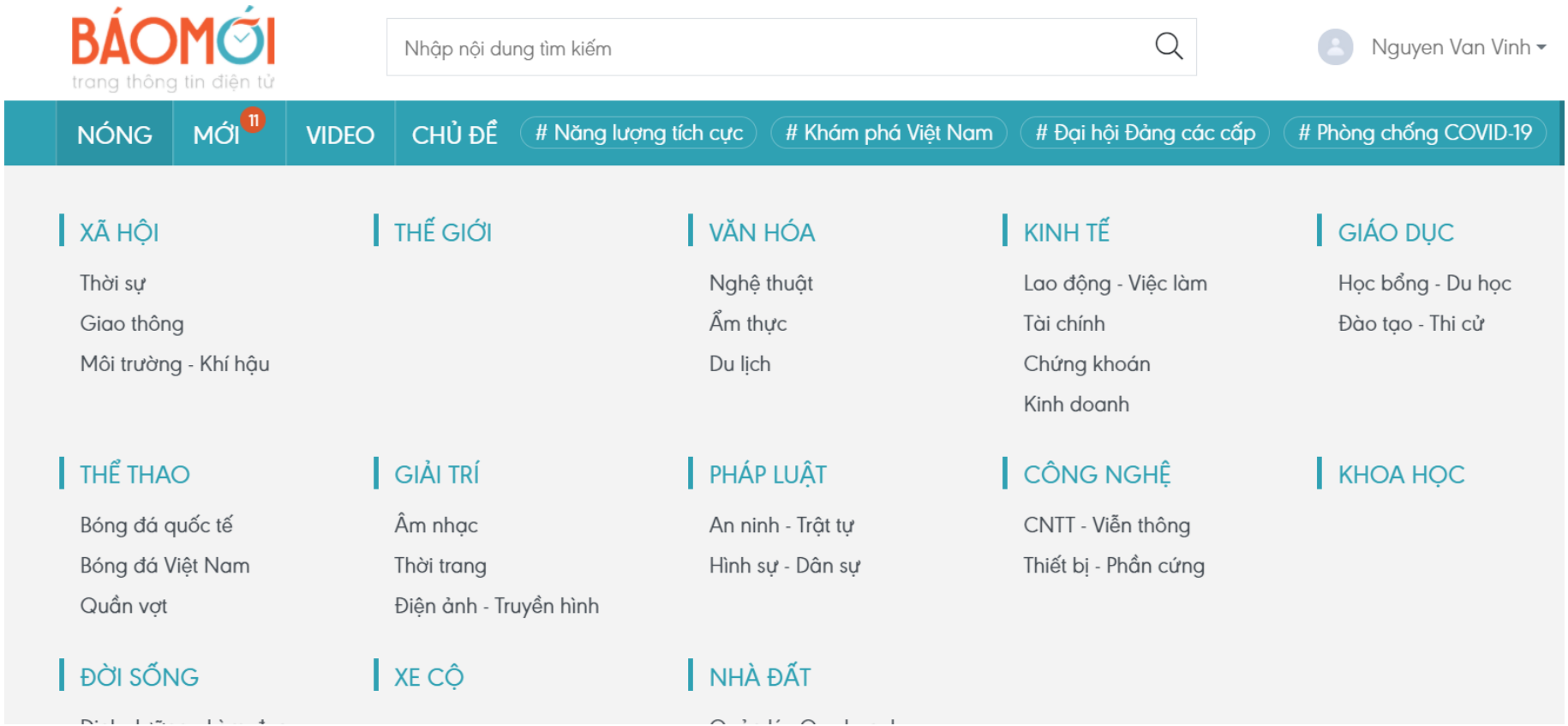
http://www.123contactform.com/contact-form-StanfordNew1-236335.html

Click on the above link to login for more information about this new exciting forum. You can also copy the above link to your browser bar and login for more information
about the new services.

Positive or negative movie review?

- unbelievably disappointing
- Full of zany characters and richly applied satire, and some great plot twists
- this is the greatest screwball comedy ever filmed
- It was pathetic. The worst part about it was the boxing scenes.

Why text classification?



Content

- Text classification problem
- Feature extraction from text
- Naïve Bayes method
- Logistic Regression method
- Sentiment Analysis Casestudy

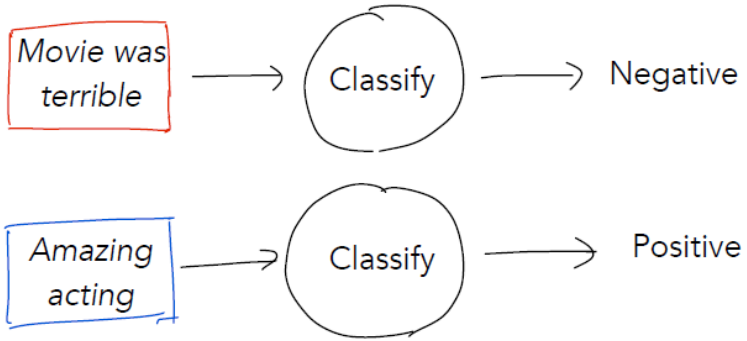
Text Classification: definition

Inputs:

- A document d
- A set of classes C (m classes)

Output:

- Predicted class $c \in C$ for document d



Classification Methods: Hand-coded rules

- Choose features
- Rules based on combinations of words or other features
  - spam: black-list-address OR (“dollars” AND“have been selected”)
- Accuracy can be high
  - If rules carefully refined by expert
- But building and maintaining these rules is expensive

Classification Methods: Supervised Machine Learning

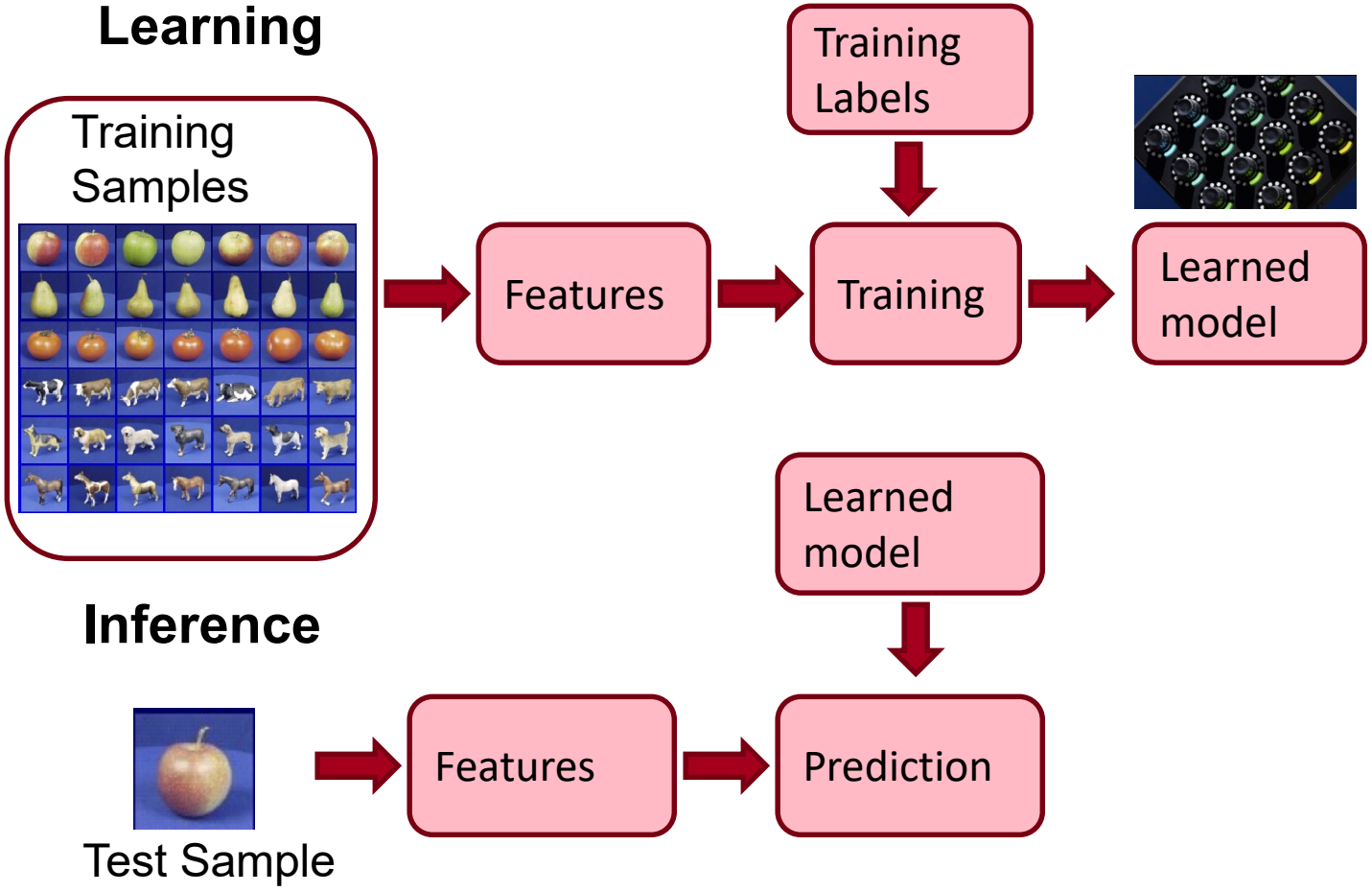
- *Input:*
  - a document *d*
  - a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$
  - A training set of *m* hand-labeled documents $(d_1,c_1),\ldots,(d_m,c_m)$

  $di \subset D$
- *Output:*
  - a learned classifier $F:D \rightarrow C$

Key questions:

a) What is the form of F?
b) b) How do we learn F?

Machine Learning Artchitecture



Classification Methods: Supervised Machine Learning

- **Any kind of classifier**
  - Naïve Bayes
  - Logistic regression
  - Support-vector machines
  - Random Forest/XGBoost/CatBoost/LightGBM
  - Neural Network
  - …

Precision and recall

- **Precision**: % of selected items that are correct
  Recall: % of correct items that are selected

	correct	not correct
selected	tp	fp
not selected	fn	tn

A combined measure: F

- A combined measure that assesses the P/R tradeoff is F measure (weighted harmonic mean):

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

- The harmonic mean is a very conservative average; see *IIR* § 8.3
- People usually use balanced F1 measure
  - i.e., with $\beta = 1$ (that is, $\alpha = ½$): $F = 2PR/(P+R)$

Features

- A measurable variable that is (rather, should be) distinctive of something we want to model.
- We usually choose features that are useful to **identify** something, i.e., to do **classification**
  - **Ex:** Cô gái đó **rất đẹp** trong bữa tiệc hôm đó.
- We often need **several** features to adequately model something – *but not too many*!

Feature vectors

- **Values** for **several** features of an **observation** can be put into a single **vector**

# proper nouns	# 1st person pronouns	# commas
2	0	0
5	0	0
0	1	1

Feature vectors

- Features should be useful in **discriminating** between categories.

Table 3: Features to be computed for each text

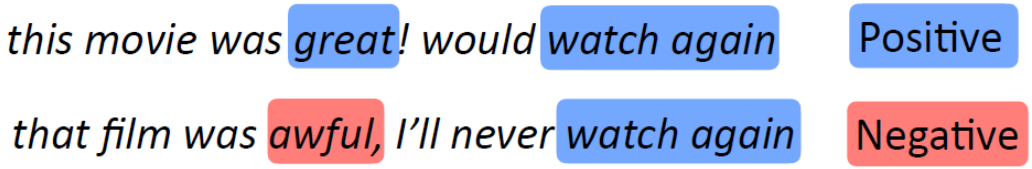
- Counts:
  - First person pronouns
  - Second person pronouns
  - Third person pronouns
  - Coordinating conjunctions
  - Past-tense verbs
  - Future-tense verbs
  - Commas
  - Colons and semi-colons
  - Dashes
  - Parentheses
  - Ellipses
  - Common nouns
  - Proper nouns
  - Adverbs
  - *wh*-words
  - Modern slang acronyms
  - Words all in upper case (at least 2 letters long)
- Average length of sentences (in tokens)
- Average length of tokens, excluding punctuation tokens (in characters)
- Number of sentences

Higher values → this person is referring to themselves (to their opinion, too?)

Higher values → looking forward to (or dreading) some future event?

Lower values → this tweet is more formal. Perhaps not overly sentimental?

Classification: Sentiment Analysis



- Surface cues can basically tell you what's going on here: presence or absence of certain words (*great*, *awful*)
- Steps to classification:
  - Turn examples like this into feature vectors
  - Pick a model / learning algorithm
  - Train weights on data to get our classifier

Feature Representation

this movie was great! would watch again Positive

- Convert this example to a vector using *bag-of-words features*

[contains *the*]	[contains *a*]	[contains *was*]	[contains *movie*]	[contains *film*]	…
position 0	position 1	position 2	position 3	position 4	
$f(x)$ = [0	0	1	1	0	…

- Very large vector space (size of vocabulary), sparse features
- Requires *indexing* the features (mapping them to axes)
- More sophisticated feature mappings possible (m-idf), as well as lots of other features: character n-grams, parts of speech, …

Feature Engineering for Text

- **Feature Engineering** is the process of creating new features or transforming existing features to improve the performance of a machine-learning model

- Tokenizing
- Bag of Words
- N-grams
- TF-IDF
- Embeddings
- NLP

Feature Selection

- Which features to use?
- How many features to use?

Approaches:

- Frequency
- Mutual Information
- Accuracy

Naive Bayes: Exercise!

- Model

$$P(x, y) = P(y) \prod_{i=1}^{n} P(x_i|y)$$

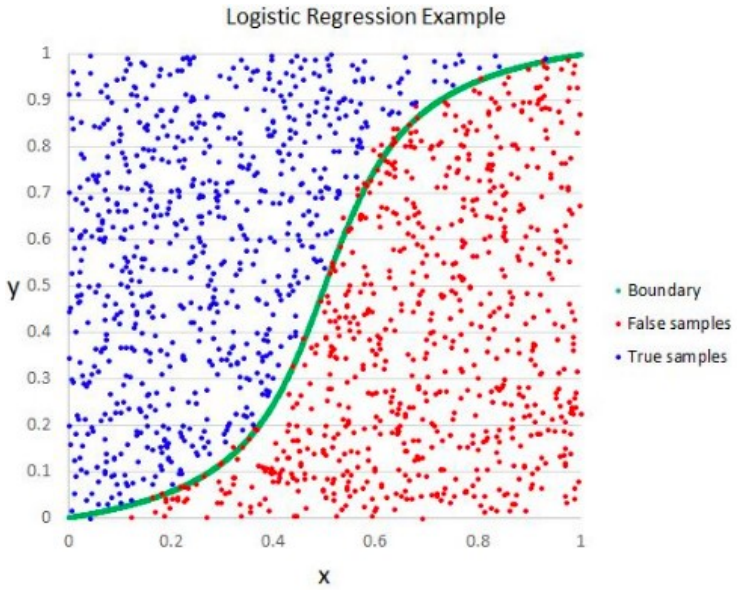
- Inference

$$\text{argmax}_y \log P(y|x) = \text{argmax}_y \left[\log P(y) + \sum_{i=1}^{n} \log P(x_i|y) \right]$$

- Learning: maximize ***P(x,y)*** by reading counts off the data

LOGISTIC REGRESSION MODEL

Logistic Regression



- Powerful supervised model
- Baseline approach for many NLP tasks
- Is also the foundation of neural networks
- Binary (two classes) or multinomial (>2 classes)

Generative and Discriminative Classifiers

- Naive Bayes is a **generative** classifier
- by contrast:
- Logistic regression is a **discriminative** classifier

Generative and Discriminative Classifiers

Suppose we're distinguishing cat from dog images

imagenet

imagenet

Generative Classifier:

- Build a model of what's in a cat image
  - Knows about whiskers, ears, eyes
  - Assigns a probability to any image:
    - how cat-y is this image?

Also build a model for dog images

Now given a new image:

Run both models and see which one fits better

Discriminative Classifier

Just try to distinguish dogs from cats

Oh look, dogs have collars!

Let's ignore everything else

Finding the correct class c from a document d in Generative vs Discriminative Classifiers

- Naive Bayes

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \overbrace{P(d|c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}}$$

- Logistic Regression

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \overset{\text{posterior}}{P(c|d)}$$

Overall process: Discriminative classifiers

Input: a set of labeled documents $\{(d_i, y_i)\}_{i=1}^{n}$

- Components:
  1. Convert d_i into a **feature representation** x_i
  2. **Classification function** to compute $\hat{y}$ using $P(\hat{y} \mid x)$
  3. **Loss function** for learning
  4. Optimization **algorithm**

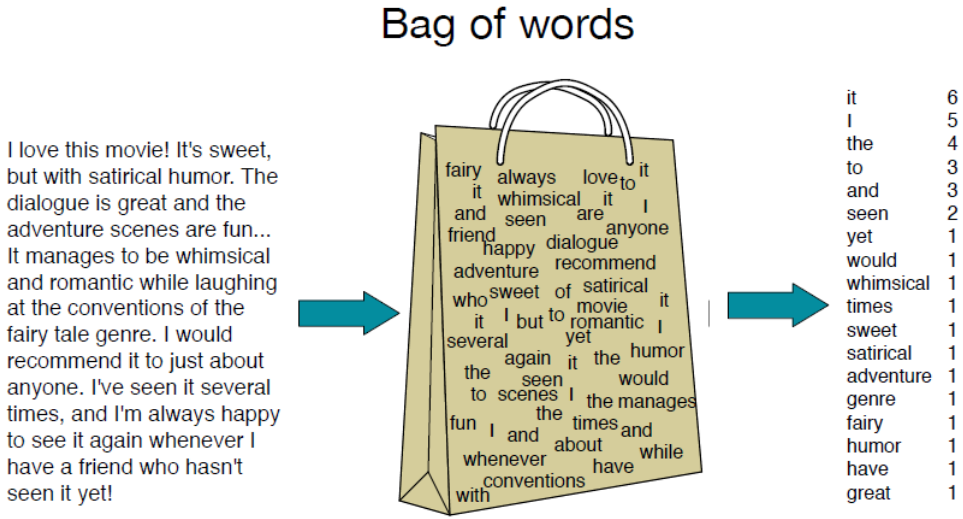
$y_i = 0$ or 1 (binary)

$y_i = 1, \ldots, m$ (multinomial)

Using either sigmoid or softmax!

- Train phase: Learn the **parameters** of the model to minimize **loss function** on the training set
- Test phase: Apply **parameters** to predict class given a new input x (feature representation of testing document d)

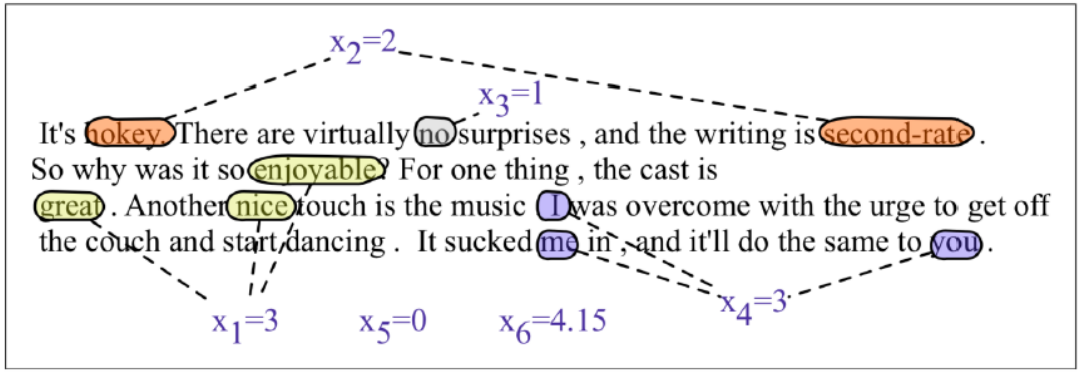
Feature representation



$$\mathbf{x} = [x_1, x_2, \ldots, x_k]$$

In BoW representations, $k = |V|$ and the vector could be very sparse

Example: Sentiment classification



Var	Definition	Value in Fig. 5.2
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)	$\ln(64) = 4.15$

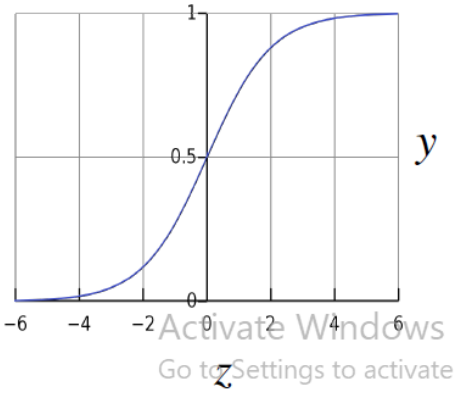
Remember that the values make up the feature vector!

Classification function

- *Given*: Input feature vector $\mathbf{x} = [x_1, x_2, \ldots, x_k]$
- *Output*: $P(y = 1 \mid \mathbf{x})$ and $P(y = 0 \mid \mathbf{x})$ *(binary classification)*

Weight vector $\mathbf{w} = [w_1, w_2, \ldots, w_k]$ bias

- Given input features $\mathbf{x}$: $z = \mathbf{w} \cdot \mathbf{x} + b$
- Therefore, $\hat{y} = P(y = 1 \mid \mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x} + b) = \dfrac{1}{1 + e^{-(\mathbf{w} \cdot \mathbf{x} + b)}}$
- Decision boundary: $= \begin{cases} 1 & \text{if } \hat{y} > 0.5 \\ 0 & \text{otherwise} \end{cases}$



Example: Sentiment classification

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)	$\ln(64) = 4.15$

- Assume weights $\mathbf{w} = [2.5, -5.0, -1.2, 0.5, 2.0, 0.7]$ and bias $b = 0.1$

$$
\begin{aligned}
p(+|x) = P(Y=1|x) &= \sigma(w \cdot x + b) \\
&= \sigma([2.5, -5.0, -1.2, 0.5, 2.0, 0.7] \cdot [3, 2, 1, 3, 0, 4.15] + 0.1) \\
&= \sigma(.805) \\
&= 0.69 \\
p(-|x) = P(Y=0|x) &= 1 - \sigma(w \cdot x + b) \\
&= 0.31
\end{aligned}
$$

Idea of logistic regression

- We'll compute $w \cdot x + b$
- And then we'll pass it through the sigmoid function:
- $$\sigma(w \cdot x + b)$$
- And we'll just treat it as a probability

Sigmoid Function

- Among the functions with the above 2 features, the function sigmoid: $f(s) = \frac{1}{1+e^{-s}}$ widely used.
- f’(s) = f(s) (1-f(s)).

Loss function

- For n data points (x_i, y_i), $\hat{y}_i = P(y_i = 1 \mid x_i)$
- Classifier probability: $\Pi_{i=1}^{n} P(y_i \mid x_i) = \Pi_{i=1}^{n} \hat{y}_i^{y_i}(1 - \hat{y}_i)^{1-y_i}$
- Loss: $-\log \prod_{i=1}^{n} P(y_i \mid x_i) = -\sum_{i=1}^{n} \log P(y_i \mid x_i)$

$$L_{CE} = -\sum_{i=1}^{n} [y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)]$$

Properties of CE loss

- $$L_{CE} = -\sum_{i=1}^{n} [y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)]$$

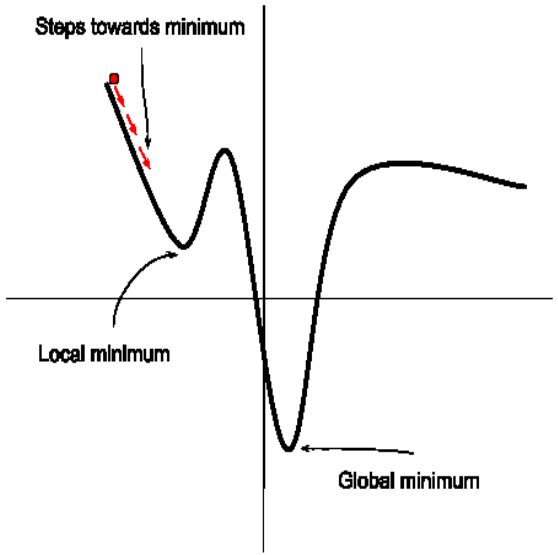
- What values can this loss take?

A) 0 to ∞ B) $-\infty$ to ∞ C) $-\infty$ to 0 D) 1 to ∞

Optimization of loss based on gradient decent

In more detail about Gradient decent: https://towardsdatascience.com/implement-gradient-descent-in-python-9b93ed7108d1

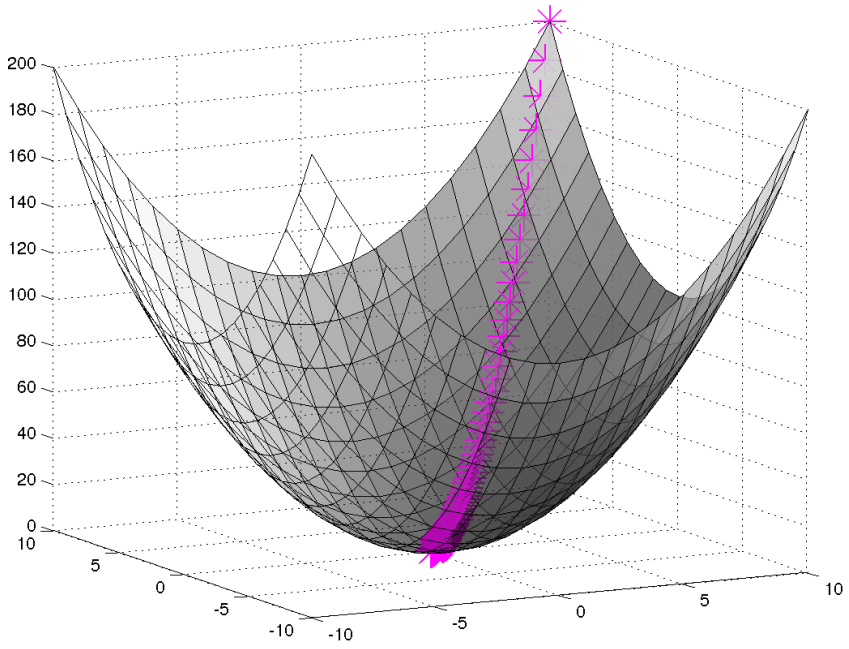
- It is an optimization algorithm to find the **minimum of a function**. We start with a random point on the function and move in the **negative direction** of **the gradient** of the function to reach the **local/global minima**.



Gradient descent

- If the derivative of the function x at x_t: $f'(x_t) > 0$ then x_t is to the right of x^* (and vice versa). In order for the next point x_{t+1} to be closer to x^*, we need to move x_t to the left, .i.e negative side. In other words, we need to move **the opposite sign with the derivative** : $x_{t+1} = x_t + \Delta$
- $x_{t+1} = x_t - \mu f'(x_t)$

Gradient descent



Gradient Descent Algorithm

Algorithm 1 Gradient Descent

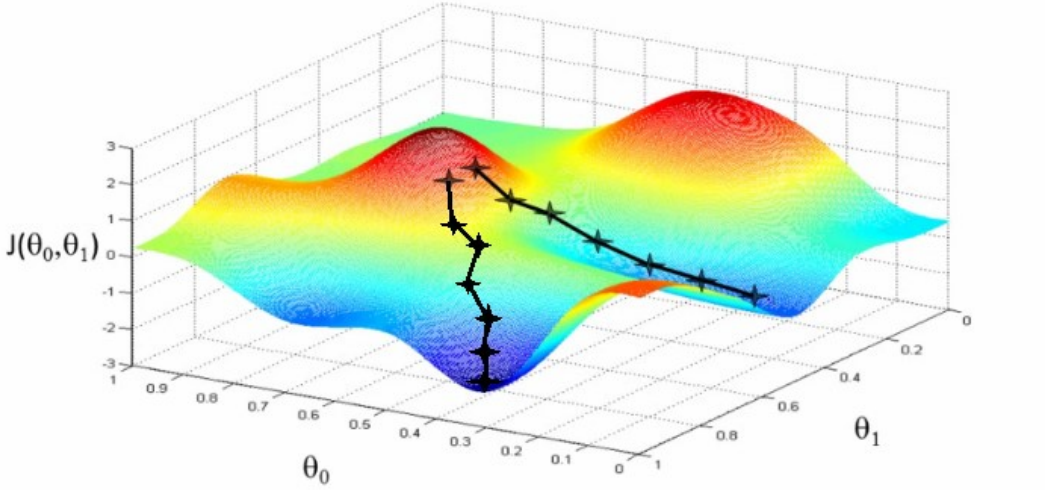
1: Initialize $\theta_0 = k \in \mathbb{R}^d$
2: Initialize $t \leftarrow 0$
3: **repeat**
4: $\quad \forall j \in \{1, ..., d\}, \theta_{j,t+1} \leftarrow \theta_{j,t} - \alpha \frac{\partial}{\partial \theta_j} L(\theta) \Big|_{\theta=\theta_t}$
5: $\quad t \leftarrow t + 1$
6: **until** converge

Variants of GD algorithm:

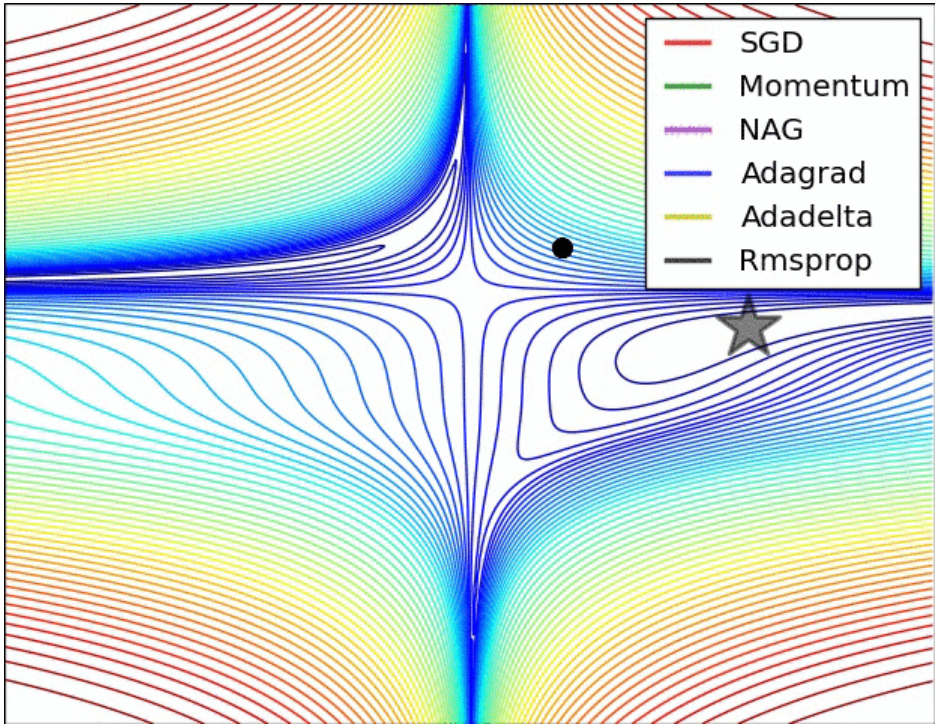
- Batch Gradient Descent
- Stochastic Gradient Descent
- Mini-batch Gradient Descent

Pros and cons of gradient descent

- Simple and often quite effective on ML tasks
- Often very scalable
- Only applies to smooth functions (differentiable)
- Might find a local minimum, rather than a global one



Visualization of algorithms of GD



Gradient for logistic regression

$$\hat{y}_i = \sigma(\mathbf{w} \cdot \mathbf{x}_i + b)$$

$$L_{CE} = -\sum_{i=1}^{n} [y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)]$$

- Gradient, $\frac{dL_{CE}(\mathbf{w}, b)}{dw_j} = \sum_{i=1}^{n} [\hat{y}_i - y_i]x_{i,j}$

  The j-th value of the feature vector $\mathbf{x}_i$

- $\frac{dL_{CE}(\mathbf{w}, b)}{db} = \sum_{i=1}^{n} [\hat{y}_i - y_i]$

Cross entropy loss for logistic regression is **convex** (i.e. has only one global minimum) so gradient descent is guaranteed to find the minimum.

Selecting learning rate

- Use grid-search in log-space over small values on a tuning set:
  - e.g., 0.01, 0.001, …
- Sometimes, decrease after each pass:
  - e.g factor of $1/(1 + dt)$, t=epoch
  - sometimes $1/t^2$
- Fancier techniques:
  - Adaptive gradient: scale gradient differently for each dimension (Adagrad, ADAM, ….)

Logistic Regression: Summary

- Model

$$P(y = +|x) = \frac{\exp(\sum_{i=1}^{n} w_i x_i)}{1 + \exp(\sum_{i=1}^{n} w_i x_i)}$$

- Inference

$\text{argmax}_y P(y|x)$ fundamentally same as Naive Bayes

$P(y = 1|x) \geq 0.5 \Leftrightarrow w^\top x \geq 0$

- Learning: gradient ascent on the (regularized) discriminative log-likelihood

SENTIMENT ANALYSIS

Google Product Search

HP Officejet 6500A Plus e-All-in-One Color Ink-jet - Fax / copier / printer / scanner
$89 online, **$100** nearby ★★★★☆ 377 reviews
September 2010 - Printer - HP - Inkjet - Office - Copier - Color - Scanner - Fax - 250 sh

Reviews

Summary - Based on 377 reviews

1 star	2	3	4 stars	5 stars

What people are saying

ease of use	"This was very easy to setup to four computers."
value	"Appreciate good quality at a fair price."
setup	"Overall pretty easy setup."
customer service	"I DO like honest tech support people."
size	"Pretty Paper weight."
mode	"Photos were fair on the high quality mode."
colors	"Full color prints came out with great quality."

Bing Shopping

HP Officejet 6500A E710N Multifunction Printer

Product summary | Find best price | **Customer reviews** | Specifications | Related items

$121.53 - $242.39 (14 stores)

☐ Compare

Average rating ★★★½☆ (144)

★★★★★	(55)
★★★★☆	(54)
★★★☆☆	(10)
★★☆☆☆	(6)
★☆☆☆☆	(23)
☆☆☆☆☆	(0)

Most mentioned

Performance	(57)
Ease of Use	(43)
Print Speed	(39)
Connectivity	(31)

More ▼

Show reviews by source

- Best Buy (140)
- CNET (5)
- Amazon.com (3)

Sentiment Analysis task

- Input: Given a sentence (paragraph) of text.
- Ouput: Determine whether this sentence is positive (1) or negative (0).
- Movie review dataset: IMDB

Sentiment Analysis

- Dataset: IMDB Movie Reviews
- Data preprocessing
- Features extraction (Vectorization)
- Choosing a machine learning model and training

Data preprocessing

- The smallest unit is the word. String of words:

Ví dụ: **Text**: This is a cat. --> **Word Sequence:** [this, is, a, cat]

- Data crawled from the web is usually very “dirty” such as “Html”, abbreviations, … should be careful with data preprocessing
- Using **Regular Expression** (RE)

RE

- Regular Expression (or **regex**) is a sequence of characters that represent a search pattern
- Each character has a meaning; for example, “`. means any character that isn't the newline character: '\n'`”
- These characters are often combined with quantifiers, such as *, which means zero or more.
- **Regular expressions** are very useful when solving with string problem.

RE

SYMBOL	USAGE
$	Matches the end of the line
\s	Matches whitespace
\S	Matches any non-whitespace character
*	Repeats a character zero or more times
\S	Matches any non-whitespace character
***?**	Repeats a character zero or more times (non-greedy)
+	Repeats a character one or more times
+?	Repeats a character one or more times (non-greedy)
[aeiou]	Matches a single character in the listed set
[^XYZ]	Matches a single character not in the listed set
[a-z0-9]	The set of characters can include a range

Quiz

- Example:

1) Extract **number** from s:

s = 'My 2 favourite numbers are 8 and 25. My mobie is 0912203062‘.

1) Extract **email** from s:

s = ‘Hello from shubhamg199630@gmail.com to priya@yahoo.com about the meeting @2PM'

Features extraction

- Now that we have a way to extract information from text in the form of word sequences, we need a way to transform these word sequences into numerical features: this is **vectorization**.
- The simplest vectorization technique is Bag Of Words (BOW). It starts with a list of words called the vocabulary (this is often all the words that occur in the training data)

Features extraction

- To use BOW vectorization in Python, we can rely on `CountVectorizer` from the `scikit-learn` library
- `scikit-learn` has a built-in list of stop words that can be ignored by passing `stop_words="english"` to the vectorizer
- Moreover, we can pass our custom pre-processing function fromearlier to automatically clean the text before it's vectorized.

Training texts: ["This is a good cat", "This is a bad day"]**=>**
vocabulary: [this, cat, day, is, good, a, bad]
New text: "This day is a good day" --> [1, 0, 2, 1, 1, 1, 0]

Sentiment Analysis CaseStudy

- Dữ liệu văn bản IMDB: The IMDB movie reviews dataset is a set of 50,000 reviews, half of which are positive and the other half negative
- We can download data from:

http://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz

- We have a directory containing data: `aclImdb`
- We can use the following function to load the training/test datasets from IMDB

Sentiment Analysis CaseStudy

- feature vectors that result from BOW are usually very large (80,000-dimensional vectors in this case)
- we need to use simple algorithms that are efficient on a large number of features (e.g., Naive Bayes, linear SVM, or logistic regression)

Improving to the current model

- Features extraction is very important (Features Engineering)
- There are some biases attached with only looking at how many times a word occurs in a text. In particular, the longer the text, the higher its features (word counts) will be
- Dựa vào đặc trưng TF-IDF
- Dựa vào n-gram

Improving model

- TF-IDF feature
- We can train the Linear SVM model with the TF-IDF feature simply by replacing the function `CountVectorizer` by `TfIdfVectorizer`
- `Result improving about 2%`

$$\text{TF(word, text)} = \frac{\text{number of times the word occurs in the text}}{\text{number of words in the text}}$$

$$\text{IDF(word)} = \textbf{log}\left[\frac{\text{number of texts}}{\text{number of texts where the word occurs}}\right]$$

$$\text{TF-IDF(word, text)} = \text{TF(word, text)} \times \text{IDF(word)}$$

Improving model

- Using independent words is not good. Ex:
- if the word `good` occurs in a text, we will naturally tend to say that this text is positive, even if the actual expression that occurs is actually `not good. Phrase is better.`
- `Using n-gram in order to solove this problem`
- An N-gram is a set of N successive words (e.g., `very good` [2-gram] and `not good at all` [4-gram]). Using N-grams, we produce richer word sequences.
- Ví dụ với N =2:

This is a cat. --> [this, is, a, cat, (this, is), (is, a), (a, cat)]

Improving model

- In practice, including N-grams in our `TF-IDF vectorizer` is as simple as providing an additional parameter `ngram_range=(1, N)`.

```
vectorizer = TfidfVectorizer(stop_words="english",
                             preprocessor=clean_text,
                             ngram_range=(1, 2))
```

Homework

- Sentiment Analysis with dataset IMDB (link: http://ai.stanford.edu/~amaas/data/sentiment/)

References

- **Chapter 4,5** - Speech and Language Processing (3rd ed. draft)

(https://web.stanford.edu/~jurafsky/slp3/4.pdf)

(https://web.stanford.edu/~jurafsky/slp3/5.pdf)