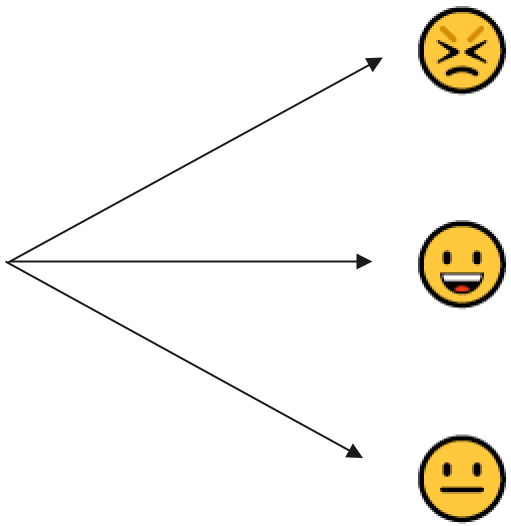
Logistic Regression (Maximum Entropy Model)

NLP II 2025
Attapol Thamrongrattanarit
รศ. ดร.อรรถพล ธำรงรัตนฤทธิ์

Text Classification

- Text classification is the task of assigning predefined categories or labels (output) to a given piece of text (input), which can be a sentence, a document, or a set of documents.

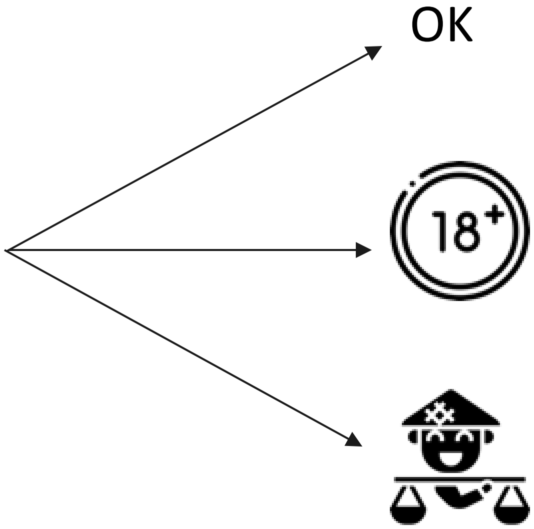
Text input
"อาหารเหมาะสำหรับสัตว์เลี้ยงเท่านั้น"
"ไกลแค่ไหนก็ต้องมาทาน"
...
"พนักงานบริการเต็มใจมาก"



Predefined categories or labels

Sentiment Analysis

Text input
"ชื่นชอบผลงานมานานแล้วค่ะ"
"เงินกู้ด่วน ไม่ต้องค้ำ ดอกเบี้ยต่ำ คลิกเลย"
"ถ้าจะเต้นแค่นี้ กลับบ้านเหอะ เซ็ง"
...
"เพลงน่าจะชัดกว่านี้ค่ะ แต่เต้นเป๊ะมาก"



Predefined categories or labels

Spam classification

4 Steps for Supervised Learning

Input	Output
"อาหารเหมาะสำหรับสัตว์เลี้ยงเท่านั้น"	
"ไกลแค่ไหนก็ต้องมาทาน"	
...	...
"พนักงานบริการเต็มใจมาก"	

Data preparation

f1	f2	f3	label
0	1	3	บวก
-1.0	0	4	ลบ
1	0	3	กลาง
1	1	4	บวก
...	...	...	...

Feature engineering

Model training

Evaluation

How make a good classifier

- Domain: The data must match the real scenario.
- Data quality and quantity: The annotation must be consistent, and the size must be large.
- Feature engineering: What does the model need to pay attention in order to make a good decision?
- Model: Some models are better than others.

Logistic Regression

Logistic Regression

- Logistic regression (or Maximum Entropy Model) is a statistical model that computes P(Y|X) from a linear combination of input features. It models a link between each label and each feature (in favor or against).
- If Y is multiclass (e.g. positive, neutral, and negative sentiment), logistic regression should be called 'multinomial logistic regression. In NLP, we call it logistic regression or MaxEnt.

Model training

- We train the model on the training set. Each model has its own formula for training the model parameters.
- We evaluate the model on the dev set. Each model has its own way of using the trained parameters. This process is called 'inference' (the model infers the labels from the text)

Components of ML classifier

1. Representation - How do we convert from text to a feature vector?
2. Inference/prediction - How do we compute P(Y|X)?
3. Training
   a. Objective function
   b. Optimization algorithm for training

Representation

A feature vector represents text. A vector is a list of numbers that can be compared with another vector to measure similarity.

- Unigram count feature = bag-of-word features we saw last week
- Unigram binary feature
- Unigram TF-IDF feature
- Bigram count feature

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-word features

Taken from Martin and Jurafsky 3rd ed.

feature vector to represent the text 'predictable and boring'

Text	**Label (Y)**	predictable	and	boring	very	few	laughs	short	but	powerful	fun	good
predictable and boring	negative	**1**	**1**	**1**								
very few laughs	negative				**1**	**1**	**1**					
short but boring	negative			**1**				**1**	**1**			
very very powerful	positive				**2**					**1**		
fun and good good laughs	positive		**1**				**1**				**1**	**2**

Bag-of-word features or bag of unigram

feature vector to represent the text 'predictable and boring'

Text	**Label (Y)**	predictable	and	boring	very	few	laughs	short	but	powerful	fun	good
predictable and boring	negative	**1**	**1**	**1**								
very few laughs	negative				**1**	**1**	**1**					
short but boring	negative			**1**				**1**	**1**			
very very powerful	positive				**1**					**1**		
fun and good good laughs	positive		**1**				**1**				**1**	**1**

Unigram binary

feature vector to represent the text 'predictable and boring'

Text	**Label (Y)**	predictable	and	boring	very	few	laughs	short	but	powerful	fun	good
predictable and boring	negative	**1**	**1/2**	**1/2**								
very few laughs	negative				**1/2**	**1**	**1/2**					
short but boring	negative			**1/2**				**1**	**1**			
very very powerful	positive				**2/2**					**1**		
fun and good good laughs	positive		**1/2**				**1/2**				**1**	**2**

Unigram TFIDF feature

feature vector to represent the text 'predictable and boring'

Text	**Label (Y)**	predictable-and	and-boring	very-few	few-laughs	short-but	…
predictable and boring	negative	**1**	**1**				**…**
very few laughs	negative			**1**	**1**		**…**
short but boring	negative					**1**	**…**
very very powerful	positive						**…**
fun and good good laughs	positive						**…**

Bag-of-bigram feature

Text Representation (Features)

- Unigram count feature assumes each label has a list of associated keywords. This is a very good baseline feature.
- Unigram binary feature is like unigram count feature but ignores the effects of duplicate words.
- Unigram TF-IDF feature is like unigram count feature but downweights some of the words that appear in too many documents.
- Bigram count feature assumes that we must consider at least two adjacent words to be able to predict the label. This feature is always too sparse i.e. too many zeros in the feature vector.
- These features do work well if we have a good amount of data, but we will see more advanced representation called 'word embeddings.' Stay tuned.

Components of ML classifier

1. Representation - How do we convert from text to a feature vector?
2. **Inference/prediction - How do we compute P(Y|X)?**
3. Training
   a. Objective function
   b. Optimization algorithm for training

Logistic Regression - Inference

- Naive Bayes computes P(Y|X) by multiplying up P(x|Y=y) (product of all features x for each label y)
- Logistic regression compute P(Y|X) by using sigmoid function (if 2 classes) or softmax function (if > 2 classes)

Consider this model

- Text: tweet
- Feature: bag-of-word ('against', 'love')+ text length
- Label: {positive (1), negative (0)}

Multiply features with parameters and sum up

Compute the unnormalized score (z) by summing up (linear combination) the product between feature and parameter

text		'against'	'love'	text length
The protester is against the ...		1	0	100

Weight	bias	'against'	'love'	text length
positive	0.05	-1	2	-0.0004

score (z)		'against'	'love'	text length
positive	0.05	1 x - 1	0 x -2	100 x - 0.0004
=-0.99				

Bias + dot product between w x

$$z = \mathbf{w} \cdot \mathbf{x} + b$$

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

	Feature vector	x_1	x_2	x_3
	x =	1	0	100

	Weight	w_1	w_2	w_3
	w =	-1	2	-0.0004

score (z)	bias+	dot product = **w** · **x**
z =	0.05 +	(1 x - 1) + (0 x -2) + (100 x -0.0004)

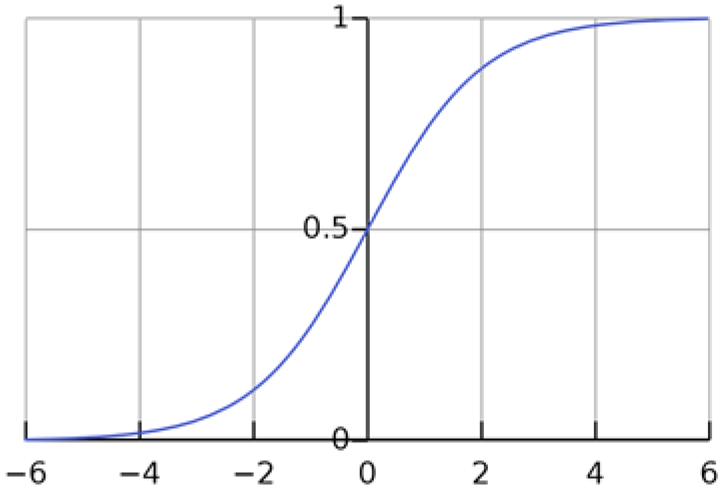
$b + w_1x_1 + w_2x_2 + w_3x_3$

Sigmoid

We convert z into probability P(Y=1|X) by passing z into a sigmoid function (also called logistic function).

$e \approx 2.71828$

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



Compute P(Y|X)

Using complementation, if Y is not 1, then Y must be 0.

P(Y=0) = 1 - P(Y=1)

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

$$
P(y=0) = 1-\sigma(\mathbf{w}\cdot\mathbf{x}+b)
$$

(Binary) Logistic Regression

- One instance of the text is represented by a feature vector $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]$
- The logistic regression model has a bias term b (also called intercept term) and weight vector $\boldsymbol{w} = [w_1, w_2, \ldots, w_n]$. b and $\boldsymbol{w}$ are the model parameters that need to be learned/trained from the training set.
- Bias b represents the score in favor of the positive label regardless of the feature vector.
- Each weight w_i, represents the score in favor of the positive label associated with each feature x_i.
- z = unnormalized score computed by bias + the dot product of $\boldsymbol{w}$ and $\boldsymbol{x}$
- The probability of the positive label P(Y=1|X) is computed by passing z into a sigmoid function.

Multiclass Logistic Regression

Multiclass Logistic Regression

- Multiclass logistic regression is just like binary logistic regression, but it supports the scenario where classes are > 2.
- The idea is the same but we have weight vector for each class. If there are 3 classes, we have three weight vectors. We concatenate these vectors and call them weight matrix.

Multiply features with parameters and sum up

Compute the unnormalized score z_j for each class j

$$z_j = \sum_{i=1}^{n} w_{ji} x_i + b_j$$

text		'against'	'love'	text length
The protester is against the ...	**x =**	1	0	100

Weight matrix (W)	bias	'against'	'love'	text length
positive	0.15	-2	-1	0.0004
negative	-0.2	2	-0.2	0.005
neutral	1	-1	0.4	-0.00001

score (z)	bias	'against'	'love'	text length
positive	0.15	1 x - 2	0 x -1	100 x 0.0004
negative	-0.2	1 x 2	0 x -0.2	100 x 0.005
neutral	1	1 x -1	0 x 0.4	100 x - 0.00001

Softmax Function

	bias	'against'	'love'	text length	**score (z)**
positive	0.15	1 x - 2	0 x -1	100 x 0.0004	**-1.81**
negative	-0.2	1 x 2	0 x -0.2	100 x 0.005	**2.3**
neutral	1	1 x -1	0 x 0.4	100 x -0.00001	**-0.001**

- Convert (normalize) **z** into a probability vector by passing it to softmax function.

	z	exp(**z**)	P(Y)
positive	-1.81	0.1636541368	0.0147
negative	2.3	9.974182455	0.8956
neutral	-0.001	0.9990004998	0.0897

$$\text{softmax}(\mathbf{z}) = \left[\frac{\exp(\mathbf{z}_1)}{\sum_{i=1}^{K} \exp(\mathbf{z}_i)}, \frac{\exp(\mathbf{z}_2)}{\sum_{i=1}^{K} \exp(\mathbf{z}_i)}, \ldots, \frac{\exp(\mathbf{z}_K)}{\sum_{i=1}^{K} \exp(\mathbf{z}_i)} \right]$$

Matrix and Vector

Notes here

https://drive.google.com/file/d/1KjgDhdY9z5THuPfZEYBXjwoEIchYyi0a/view?usp=drive_link

Matrix Multiplication

- Compute dot product for each row

$$
\mathbf{z} = \underbrace{\begin{array}{ccc} \text{'against'} & \text{'love'} & \text{text length} \\ \hline -2 & -1 & 0.0004 \\ 2 & -0.2 & 0.005 \\ -1 & 0.4 & -0.00001 \end{array}}_{\mathbf{W}} \cdot \underbrace{\begin{array}{cl} 1 & \text{'against'} \\ 0 & \text{'love'} \\ 100 & \text{'text length} \end{array}}_{\mathbf{x}} + \underbrace{\begin{array}{c} \text{bias} \\ \hline 0.15 \\ -0.2 \\ 1 \end{array}}_{\mathbf{b}}
$$

$$
\mathbf{z} = \begin{bmatrix} -1.96 \\ 2.5 \\ -1.001 \end{bmatrix} + \underbrace{\begin{bmatrix} 0.15 \\ -0.2 \\ 1 \end{bmatrix}}_{\mathbf{b}}
$$

Matrix Multiplication

- Compute dot product for each row

$$
z = \underbrace{\begin{array}{ccc} \text{'against'} & \text{'love'} & \text{text length} \\ \left[\begin{array}{c} -2 \\ 2 \\ -1 \end{array}\right. & \begin{array}{c} -1 \\ -0.2 \\ 0.4 \end{array} & \left.\begin{array}{c} 0.0004 \\ 0.005 \\ -0.00001 \end{array}\right] \end{array}}_{\mathbf{W}} \cdot \underbrace{\begin{bmatrix} 1 \\ 0 \\ 100 \end{bmatrix}}_{\mathbf{x}} \begin{array}{l} \text{'against'} \\ \text{'love'} \\ \text{'text length} \end{array} + \underbrace{\begin{array}{c} \text{bias} \\ \begin{bmatrix} 0.15 \\ -0.2 \\ 1 \end{bmatrix} \end{array}}_{\mathbf{b}}
$$

$$
z = \begin{bmatrix} -1.96 \\ 2.5 \\ -1.001 \end{bmatrix} + \underbrace{\begin{bmatrix} 0.15 \\ -0.2 \\ 1 \end{bmatrix}}_{\mathbf{b}}
$$

Matrix Multiplication

- Compute dot product for each row

$$
z = \underbrace{\begin{array}{c} \text{'against'} \quad \text{'love'} \quad \text{text length} \\ \begin{bmatrix} -2 & -1 & 0.0004 \\ 2 & -0.2 & 0.005 \\ -1 & 0.4 & -0.00001 \end{bmatrix} \end{array}}_{\mathbf{W}} \cdot \underbrace{\begin{array}{cl} \begin{bmatrix} 1 \\ 0 \\ 100 \end{bmatrix} & \begin{array}{l} \text{'against'} \\ \text{'love'} \\ \text{'text length} \end{array} \end{array}}_{\mathbf{x}} + \underbrace{\begin{array}{c} \text{bias} \\ \begin{bmatrix} 0.15 \\ -0.2 \\ 1 \end{bmatrix} \end{array}}_{\mathbf{b}}
$$

$$
z = \begin{bmatrix} -1.96 \\ 2.5 \\ -1.001 \end{bmatrix} + \underbrace{\begin{bmatrix} 0.15 \\ -0.2 \\ 1 \end{bmatrix}}_{\mathbf{b}}
$$

Matrix Multiplication

- Compute dot product for each row

$$\mathbf{z} = \underset{\mathbf{W}}{\begin{array}{c} \begin{array}{ccc}\text{'against'} & \text{'love'} & \text{text length}\end{array} \\ \begin{bmatrix} -2 & -1 & 0.0004 \\ 2 & -0.2 & 0.005 \\ -1 & 0.4 & -0.00001 \end{bmatrix}\end{array}} \; . \; \underset{\mathbf{x}}{\begin{bmatrix}1\\0\\100\end{bmatrix}} \begin{array}{l}\text{'against'}\\ \text{'love'}\\ \text{'text length}\end{array} + \underset{\mathbf{b}}{\begin{array}{c}\text{bias}\\ \begin{bmatrix}0.15\\-0.2\\1\end{bmatrix}\end{array}}$$

$$\mathbf{z} = \begin{bmatrix}-1.96\\2.5\\-1.001\end{bmatrix} + \underset{\mathbf{b}}{\begin{bmatrix}0.15\\-0.2\\1\end{bmatrix}}$$

Matrix Multiplication

- Compute dot product for each row
- Add the result with the bias vector

$$
\mathbf{z} = \underbrace{\begin{array}{ccc} \text{'against'} & \text{'love'} & \text{text length} \\ \left[\begin{array}{r} -2 \\ 2 \\ -1 \end{array}\right. & \begin{array}{r} -1 \\ -0.2 \\ 0.4 \end{array} & \left.\begin{array}{r} 0.0004 \\ 0.005 \\ -0.00001 \end{array}\right] \end{array}}_{\mathbf{W}} \cdot \underbrace{\begin{bmatrix} 1 \\ 0 \\ 100 \end{bmatrix}}_{\mathbf{x}} \begin{array}{l} \text{'against'} \\ \text{'love'} \\ \text{'text length} \end{array} + \underbrace{\begin{array}{c} \text{bias} \\ \begin{bmatrix} 0.15 \\ -0.2 \\ 1 \end{bmatrix} \end{array}}_{\mathbf{b}}
$$

$$
\mathbf{z} = \begin{bmatrix} -1.96 \\ 2.5 \\ -1.001 \end{bmatrix} + \underbrace{\begin{bmatrix} 0.15 \\ -0.2 \\ 1 \end{bmatrix}}_{\mathbf{b}}
$$

$$
\mathbf{z} = \begin{bmatrix} -1.81 \\ 2.3 \\ -0.001 \end{bmatrix}
$$

Matrix Multiplication

- Compute dot product for each row
- Add the result with the bias vector
- Then softmax vector **z**

$$
\mathbf{z} = \underbrace{\begin{array}{ccc} \text{'against'} & \text{'love'} & \text{text length} \\ \left[\begin{matrix} -2 \\ 2 \\ -1 \end{matrix}\right. & \begin{matrix} -1 \\ -0.2 \\ 0.4 \end{matrix} & \left.\begin{matrix} 0.0004 \\ 0.005 \\ -0.00001 \end{matrix}\right] \end{array}}_{\mathbf{W}} \cdot \underbrace{\begin{bmatrix} 1 \\ 0 \\ 100 \end{bmatrix}}_{\mathbf{x}} \begin{array}{l} \text{'against'} \\ \text{'love'} \\ \text{'text length} \end{array} + \underbrace{\begin{array}{c} \text{bias} \\ \begin{bmatrix} 0.15 \\ -0.2 \\ 1 \end{bmatrix} \end{array}}_{\mathbf{b}}
$$

$$
\mathbf{z} = \begin{bmatrix} -1.96 \\ 2.5 \\ -1.001 \end{bmatrix} + \underbrace{\begin{bmatrix} 0.15 \\ -0.2 \\ 1 \end{bmatrix}}_{\mathbf{b}}
$$

$$
\mathbf{z} = \begin{bmatrix} -1.81 \\ 2.3 \\ -0.001 \end{bmatrix}
$$

$$
\text{softmax}(\mathbf{z}) = \left[\frac{\exp(\mathbf{z}_1)}{\sum_{i=1}^{K}\exp(\mathbf{z}_i)}, \frac{\exp(\mathbf{z}_2)}{\sum_{i=1}^{K}\exp(\mathbf{z}_i)}, \ldots, \frac{\exp(\mathbf{z}_K)}{\sum_{i=1}^{K}\exp(\mathbf{z}_i)}\right]
$$

Inference for multinomial logistic regression

- Same as binary logistic regression. One instance of the text is represented by a feature vector $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]$ if we use n features.
- The logistic regression model has a bias vector $\boldsymbol{b}$ and weight matrix $\mathbf{W}$ (of size k x n if we have k classes.,. $\boldsymbol{b}$ and $\mathbf{W}$ are the model parameters that need to be learned/trained from the training set.
- b_i is the bias for the class i. If $b_i > 0$, then class i is more probable.
- w_{ij} is the weight for class *i* feature j. (row i column j in $\mathbf{W}$)
  If $w_{ij} > 0$ and $x_j > 0$_, then class i is more probable.
- The probability $P(Y|\mathbf{x})$ is computed as follows: $\mathbf{y} = \mathrm{softmax}(\mathbf{Wx} + \mathbf{b})$

Components of ML classifier

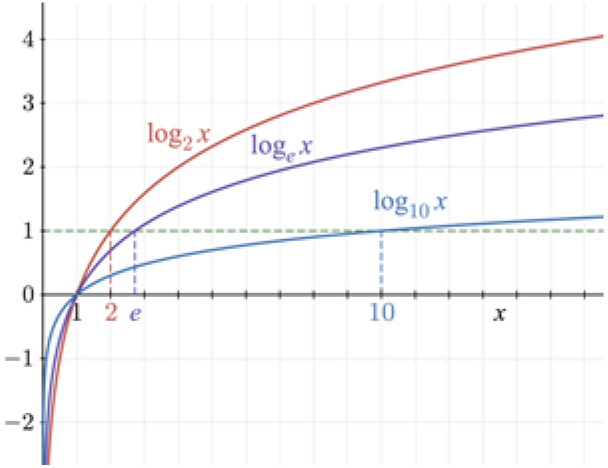
1. Representation - How do we convert from text to a feature vector?
2. Inference/prediction - How do we compute P(Y|X)?
3. **Training**
    a. **Objective function**
    b. Optimization algorithm for training

Where do **W** and ***b*** come from?

- We want W and b that produce P(Y|x) to be the most similar to the actual label in the training set. So we need to measure the similarity, which is called loss function or cost function.
  - If loss is high, the model is bad.
  - If loss is low, the model is good.
- An optimization algorithm is an algorithm that helps minimize loss function. (optimize = หาค่าที่เหมาะสมที่สุด ถึงแม้ว่าสถานการณ์ต้องมีการได้อย่างเสียอย่างเกิดขึ้น)

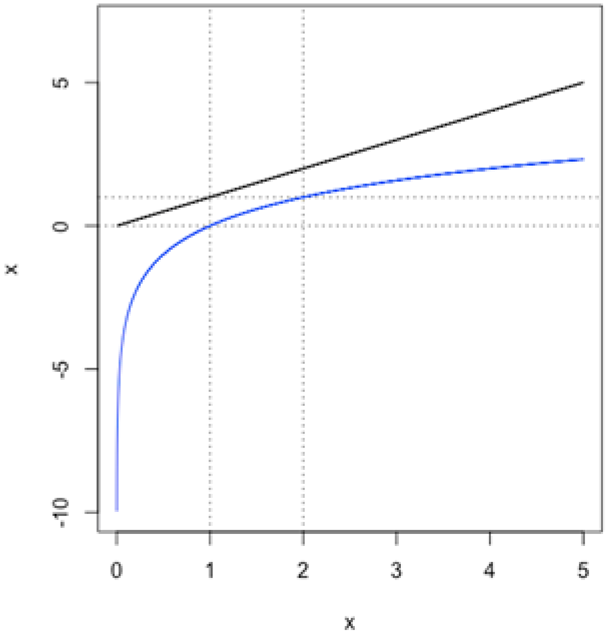
Aside: Logarithm

- A logarithm log(x) is a mathematical function that
  - 'compresses' the range of x if x > 1
  - 'expands' the range of x if x < 1.
- A logarithm of different base has different shapes. ln (natural logarithm) is a logarithm with base e ≈ 2.71828
- In machine learning, log is assumed to have base e.



Aside: Logarithm probability

- $0 < P(x) < 1$ so the value can be hard to read and calculate.
- Log probability: $-\text{Inf} < \log(P(x)) < 0$ since log compresses the range. A small change in $P(x)$ results in large change in $\log(P(x))$
  - $\log(0.001) = -6.907\ldots$
  - $\log(0.002) = -6.214\ldots$
  - $\log(0.020) = -3.912\ldots$



Measuring the deviation from gold standard

- Intuitively, we want the model parameters **W** and ***b*** such that the accuracy is the highest possible. But the accuracy is not a smooth function, so it is not suitable.
- Instead, we use If y is the correct label for text x, then we want **W** and ***b*** such that P(Y=y|x, W, b) to be very close to 1. That means log(P(Y=y|x, W, b) should be close to log(1) = 0 if the model is good.
    - Remember 0 < P(Y|X) < 1 and -Inf < log P(Y|X) < 0

Computing cross-entropy loss

- Example https://docs.google.com/spreadsheets/d/1W2ss3QwECw1g8W087vZ6Pqnjcn7iZ9jPZxALMkHXmv8/edit?usp=sharing

Cross-entropy Loss

$$L_{CE}(W,b) = -\frac{1}{N}\sum_{i=1}^{N} \log P(Y = y^i|x^i, W, b)$$

W, b are the model parameters for a logistic regression model.
y^i is the the label of row i
x^i the feature vector representing row i
$P(Y = y^i|x^i, W, b)$ is the likelihood of the gold standard label from row i

Components of ML classifier

1. Representation - How do we convert from text to a feature vector?
2. Inference/prediction - How do we compute P(Y|X)?
3. Training
    a. Objective function
    b. **Optimization algorithm for training**

Optimization algorithm

- Our goal is to find the optimal W and b which maximizes the probability (likelihood) of the label in the training set. = we want to minimize the loss function (e.g. cross-entropy loss)
- Gradient descent is an optimization algorithm that is the base for many modern optimization algorithms that we will see later in this class.

Calculus Review

Notes here

Gradient is the direction where the slope is steepest

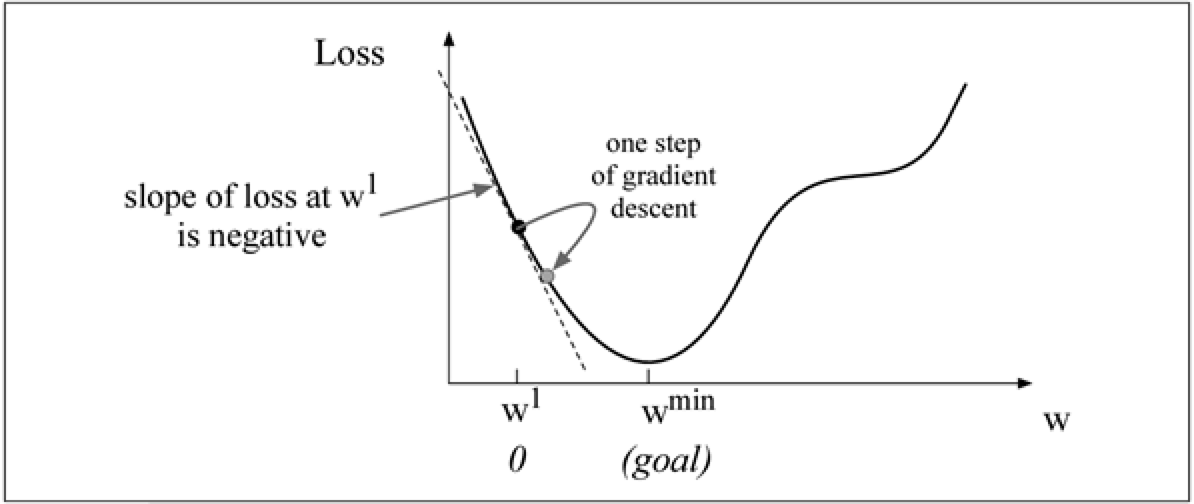
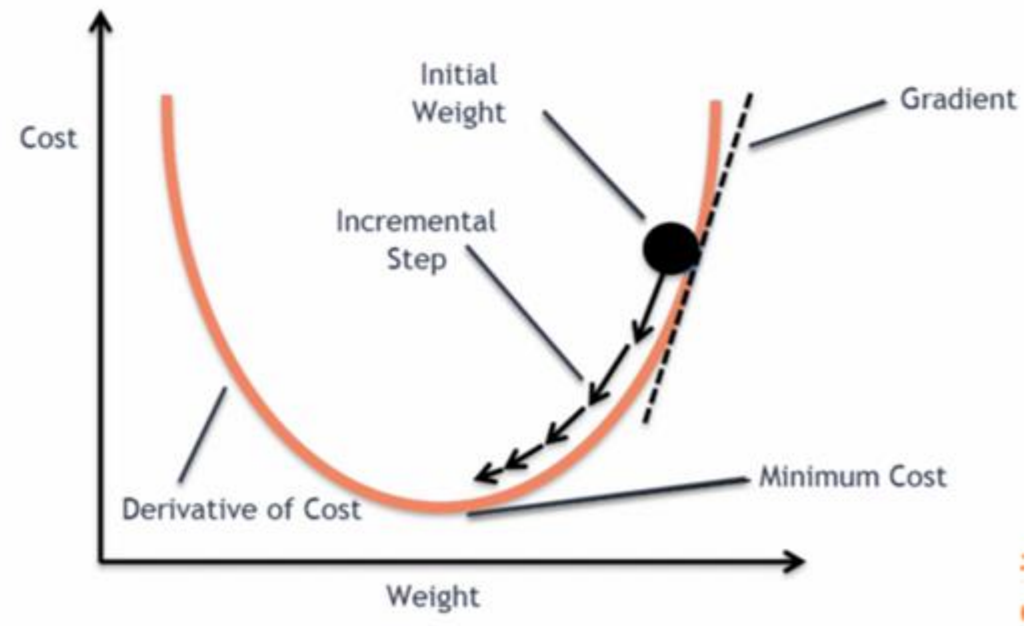


Figure 5.4 The first step in iteratively finding the minimum of this loss function, by moving w in the reverse direction from the slope of the function. Since the slope is negative, we need to move w in a positive direction, to the right. Here superscripts are used for learning steps, so w^1 means the initial value of w (which is 0), w^2 the value at the second step, and so on.

Gradient Descent

การใช้ gradient (ซึ่งมาจากการ diff objective/loss function) นำมาปรับ parameter เพื่อให้ loss function ต่ำลง



https://blog.clairvoyantsoft.com/the-ascent-of-gradient-descent-23356390836f

We compute gradient for each w_{ij}

$$\frac{\partial}{\partial w_{ij}} L_{CE}(W,b) = g_{ij} = \begin{cases} (P(Y=i|X,W,b)-1)x_j & \text{true label = i} \\ (P(Y=i|X,W,b)-0)x_j & \text{otherwise} \end{cases}$$

Update equation for w_{ij} for stochastic gradient

$$\frac{\partial}{\partial w_{ij}} L_{CE}(W,b) = g_{ij} = \begin{cases} (P(Y=i|X,W,b)-1)x_j & \text{true label} = \text{i} \\ (P(Y=i|X,W,b)-0)x_j & \text{otherwise} \end{cases}$$

new w_{ij} = old w_{ij} - η g_{ij}

- If the probability for the true label is 1, then do nothing. (it's perfect)
- If the probability for the true label is too low, then increase the weight.
- If the probability for the false label is 0, then do nothing. (it's perfect)
- If the probability for the false label is too high, then decrease the weight.

Compute the prediction error and then correct it

$$\frac{\partial}{\partial w_{ij}} L_{CE}(W,b) = g_{ij} = \begin{cases} (P(Y=i|X,W,b)-1)x_j & \text{true label} = \text{i} \\ (P(Y=i|X,W,b)-0)x_j & \text{otherwise} \end{cases}$$

$$\frac{\partial}{\partial w_{ij}} L_{CE}(W,b) = g_{ij} = \begin{cases} -(1-P(Y=i|X,W,b))x_j & \text{true label} = \text{i} \\ -(0-P(Y=i|X,W,b))x_j & \text{otherwise} \end{cases}$$

Stochastic Gradient Algorithm

```
Repeat until OK
       for x, y in training_set:
              compute P(Y|x)
              update wᵢⱼ = old wᵢⱼ - η gᵢⱼ
```

Batch Stochastic Gradient Descent Algorithm

Repeat until OK

    $total\_g_{ij} = 0$

    for x, y in training_set:

        compute $P(Y|x)$ and g_{ij}

        $total\_g_{ij} \mathrel{+}= g_{ij}$

    update $w_{ij} =$ old $w_{ij} - \eta\ (total\_g_{ij} / N)$

Batch vs Stochastic Gradient

- Batch gradient descent uses the entire dataset to calculate the gradients at each step. The gradient is more accurate because we average across many training samples.
  - + The process is stable because of averaging across the training set.
  - - This can be computationally expensive when the dataset is large.
- Stochastic gradient descent uses a single data point to calculate the gradients at each step.
  - + Computing gradient is efficient based on one single point at a time.
  - - The process is less stable because some data points are bad and give us a wrong direction to update.

Interpreting Logistic Regression

Interpreting models

"Often we want to know more than just the correct classification of an observation. We want to know why the classifier made the decision it did. That is, we want our decision to be interpretable. Interpretability can be hard to define strictly, but the core idea is that as humans we should know why our algorithms reach the conclusions they do. Because the features to logistic regression are often human-designed, one way to understand a classifier's decision is to understand the role each feature plays in the decision." (Martin and Jurafsky, 3rd edition)

Example: Examine heavy weights of each class

	b	predictable	boring	very	few	laughs	short	powerful	fun	good
negative	**0.20**	**0.14**	**0.29**	0.10	0.20	-0.34	-0.10	-0.10	-0.40	-0.20
positive	-0.4	-0.20	-1.00	0.01	**0.20**	**0.20**	-0.40	**0.20**	**1.10**	**0.40**
neutral	1	-0.50	-2.30	**0.50**	0.10	0.01	**0.30**	-0.40	0.10	**0.20**

How do we develop a text classifier on a real dataset?

Scikit-learn (sklearn)

- Scikit-learn (also known as sklearn) is a free and open-source Python library for machine learning. It provides a range of supervised and unsupervised learning algorithms in Python. It also provides preprocessing tools, feature extraction tools, and evaluation tools.
- The library is very well-documented and actively maintained by a strong open-source community.

Demo

- Naive Bayes and logistic regression on sklearn.
- For the most part, you can ask ChatGPT for the full code.
- Let's go!