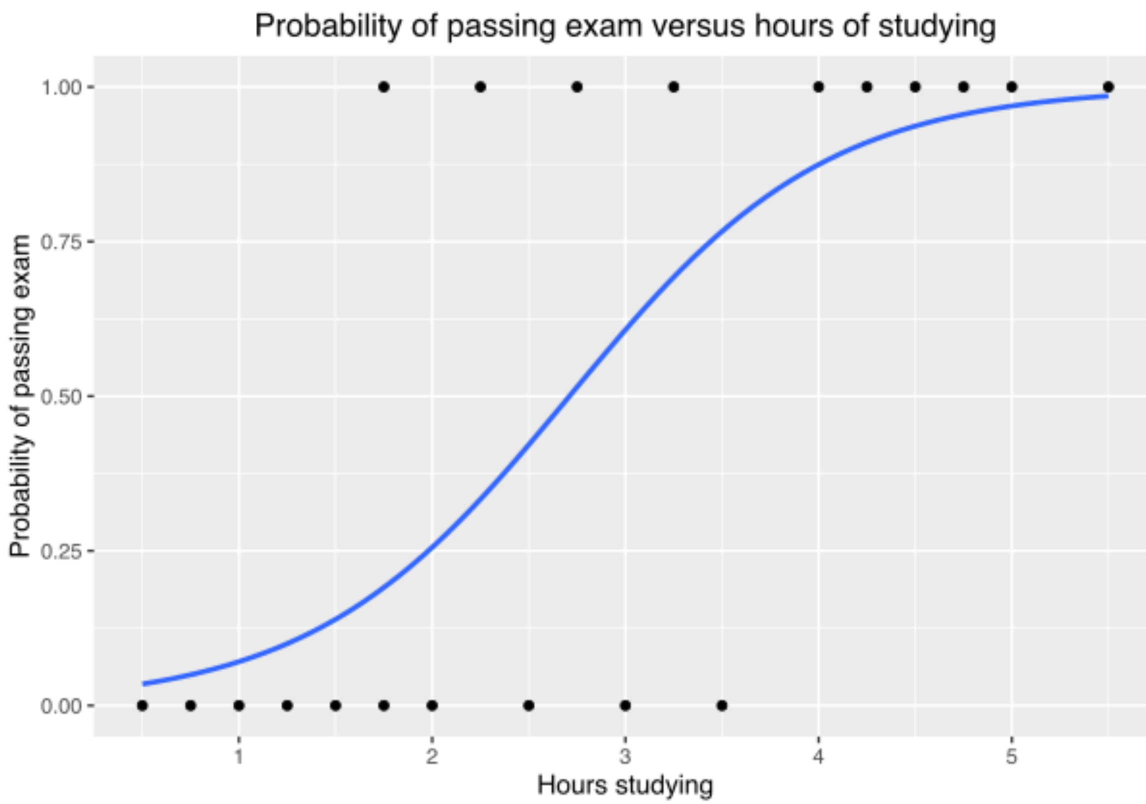
CS440/ECE448 Lecture 10: Logistic Regression & word2vec

Mark Hasegawa-Johnson

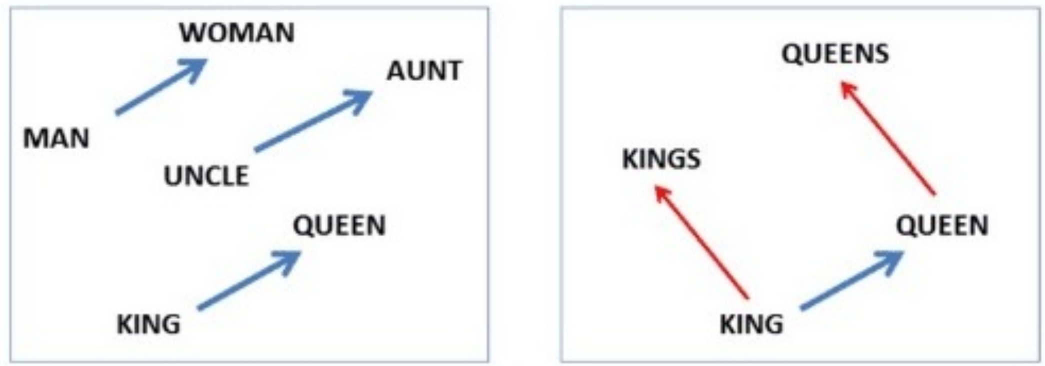


Outline

- “You shall know a word by the company it keeps”
- Linear Regression review
- Dot-product similarity
- Calculating probabilities using the logistic sigmoid
- Derivative of the log sigmoid
- Noise contrastive estimation (NCE)

vec("woman") - vec("man") + vec("king") = vec("queen")

Today's Goal: Lexical Semantics



Christian S. Perone, "Voynich Manuscript: word vectors and t-SNE visualization of some patterns," in *Terra Incognita*, 16/01/2016, http://blog.christianperone.com/2016/01/voynich-manuscript-word-vectors-and-t-sne-visualization-of-some-patterns/.

Suppose we want each word to be an m-vector

We want similar words to be represented by similar vectors

We want meaning changes to be simple movements in vector space

How can we do that?

Skip-gram: “You shall know a word by the company it keeps”

- The key idea of skip-gram is that “You shall know a word by the company it keeps” (J.R. Firth, 1957)
- The words “vanish” and “disappear” should be considered similar if they can occur in the same contexts:

“The ship will vanish into the mists.”

”The ship will disappear into the mists.”

Review: Naïve Bayes: the “Bag-of-words” model

We can estimate the likelihood of an e-mail by pretending that the e-mail is just a bag of words (order doesn’t matter).

With only a few thousand spam e-mails, we can get a pretty good estimate of these things:

- $P(W = \text{“hi”} | Y = \text{spam})$, $P(W = \text{“hi”} | Y = \text{ham})$
- $P(W = \text{“vitality”} | Y = \text{spam})$, $P(W = \text{“vitality”} | Y = \text{ham})$
- $P(W = \text{“production”} | Y = \text{spam})$, $P(W = \text{“production”} | Y = \text{ham})$

Then we can approximate $P(X|Y)$ by assuming that the words, W, are ***conditionally independent of one another given the category label***:

$$P(X = x | Y = y) \approx \prod_{i=1}^{n} P(W = w_i | Y = y)$$

Similarity: The Internet is the database

Similarity = words can be used interchangeably in most contexts

How do we measure that in practice?

Answer: extract examples of word w_1, +/- C words (C=2 or 3):

…hot, although iced **coffee** is a popular…

…indicate that moderate **coffee** consumption is benign…

…and of w_2:

…consumed as iced **tea**. Sweet tea is…

…national average of **tea** consumption in Ireland…

The words "iced" and "consumption" appear in both contexts, so we can conclude that $s(\text{coffee}, \text{tea}) > 0$. No other words are shared, so we can conclude $s(\text{coffee}, \text{tea}) < 1$.

Turning the internet into a list of word pairs

…hot, although iced **coffee** is a popular…

…indicate that moderate **coffee** consumption is benign…

…consumed as iced **tea**. Sweet tea is…

…national average of **tea** consumption in Ireland…

List each word in the internet, v_t, and all of its context words, $u_{t,1}, \ldots, u_{t,6}$:

v_t	$u_{t,1}$	$u_{t,2}$	$u_{t,3}$	$u_{t,4}$	$u_{t,5}$	$u_{t,6}$
coffee	hot	although	iced	is	a	popular
coffee	indicate	that	moderate	consumption	is	benign
tea	consumed	as	iced	sweet	tea	is
tea	national	average	of	consumption	in	Ireland
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Continuous Bag of Words (CBOW)

“Context bag of words” (CBOW) approximates each word’s probability, in context, by the product of single-word context probabilities:

$$p\left(v_t \middle| u_{t,1}, \dots, u_{t,6}\right) \approx \prod_{j=1}^{6} p(v_t | u_{t,j})$$

Using this model, two words are considered similar if they have similar CBOW probabilities:

$$p(\text{vanish}|\text{the ship will } - \text{ into the mists}) \approx$$
$$p(\text{disappear}|\text{the ship will } - \text{ into the mists})$$

Skip-gram

"Skip-gram" approximates the probability of the context given the center word:

$$p\left(u_{t,1}, \ldots, u_{t,6} \middle| v_t\right) \approx \prod_{j=1}^{6} p(u_{t,j}|v_t)$$

Using this model, two words are considered similar if they have similar skip-gram probabilities:

$$p(\text{the ship will } - \text{into the mists}|\text{vanish}) \approx$$
$$p(\text{the ship will } - \text{into the mists}|\text{disappear})$$

Maximize probability of the training dataset?

- Let's write the training dataset as a whole bunch of word pairs that should be similar: $\mathcal{D} = \left\{\left(v_1, u_{1,1}\right), \ldots, \left(v_T, u_{T,2C}\right)\right\}$
- Let's suppose all those examples are chosen independently. Then

$$P(\mathcal{D}) = \prod_{t=1}^{T} \prod_{j=1}^{2C} P\left(u_{t,j} | v_j\right)$$

- Gradient method: choose some parameters, $\boldsymbol{w}$, to maximize $P(\mathcal{D})$
- What is $\frac{\partial}{\partial \boldsymbol{w}} \prod_{t=1}^{T} \prod_{j=1}^{2C} P\left(u_{t,j} | v_j\right)$? Answer: UGLY!!!!!

No! Maximize its log probability!

- Notice that whatever parameters maximizes this:

$$P(\mathcal{D}) = \prod_{t=1}^{T} \prod_{j=1}^{2C} P\left(u_{t,j} | v_j\right)$$

… will also minimize this:

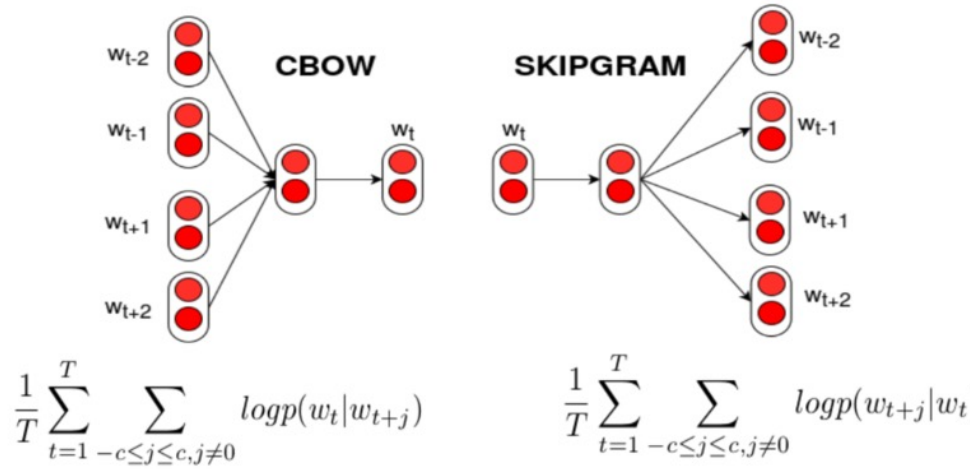
$$-\log P(\mathcal{D}) = -\sum_{t=1}^{T} \sum_{j=1}^{2C} \log P\left(u_{t,j} | v_j\right)$$

…and…

$$-\frac{\partial}{\partial \boldsymbol{w}} \log P(\mathcal{D}) = -\sum_{t=1}^{T} \sum_{j=1}^{2C} \frac{\partial}{\partial \boldsymbol{w}} \log P\left(u_{t,j} | v_j\right)$$

- Using log probability, instead of probability, allows us to differentiate each training token separately --- simplifies computation A LOT!

Neural net loss = negative log probability



Log probability is easier to differentiate than probability, so:

CBOW:

$$\mathcal{L} = -\frac{1}{T}\sum_{t=0}^{T-1}\sum_{j=1}^{2C}\ln P\left(v_t|u_{t,j}\right)$$

Skip-gram:

$$\mathcal{L} = -\frac{1}{T}\sum_{t=0}^{T-1}\sum_{j=1}^{2C}\ln P\left(u_{t,j}|v_t\right)$$

Outline

- "You shall know a word by the company it keeps"
- Linear Regression review
- Dot-product similarity
- Calculating probabilities using the logistic sigmoid
- Derivative of the log sigmoid
- Noise contrastive estimation (NCE)

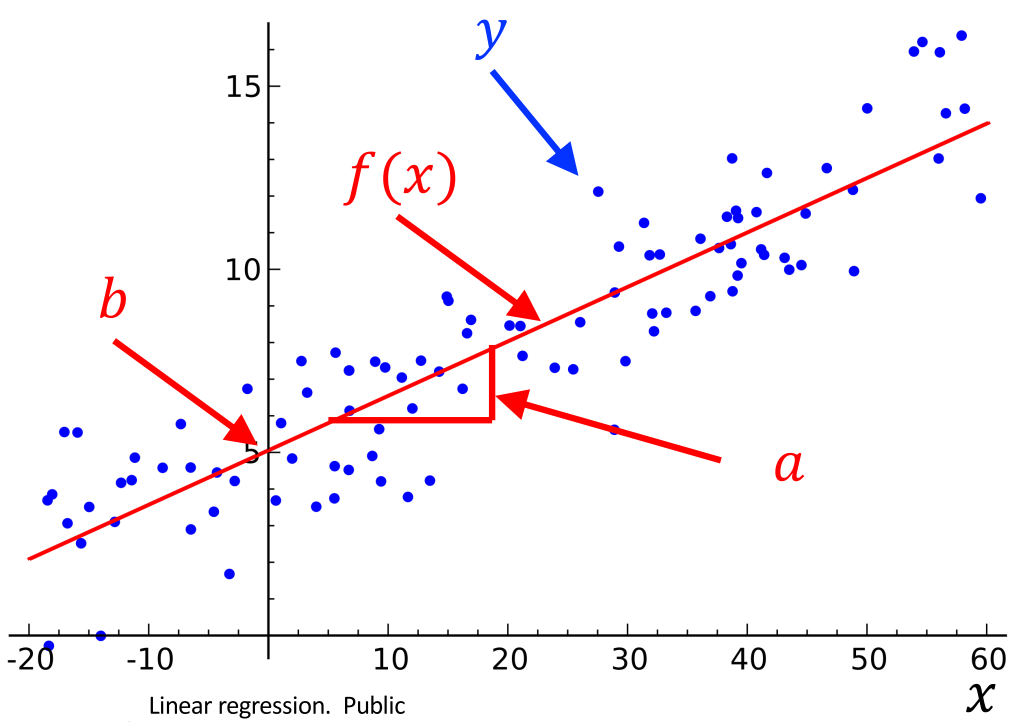
Linear regression

Linear regression is used to estimate a real-valued target variable, y, using a linear function of another variable, x:

$$f(x) = ax + b$$

… so that …

$$f(x) \approx y$$



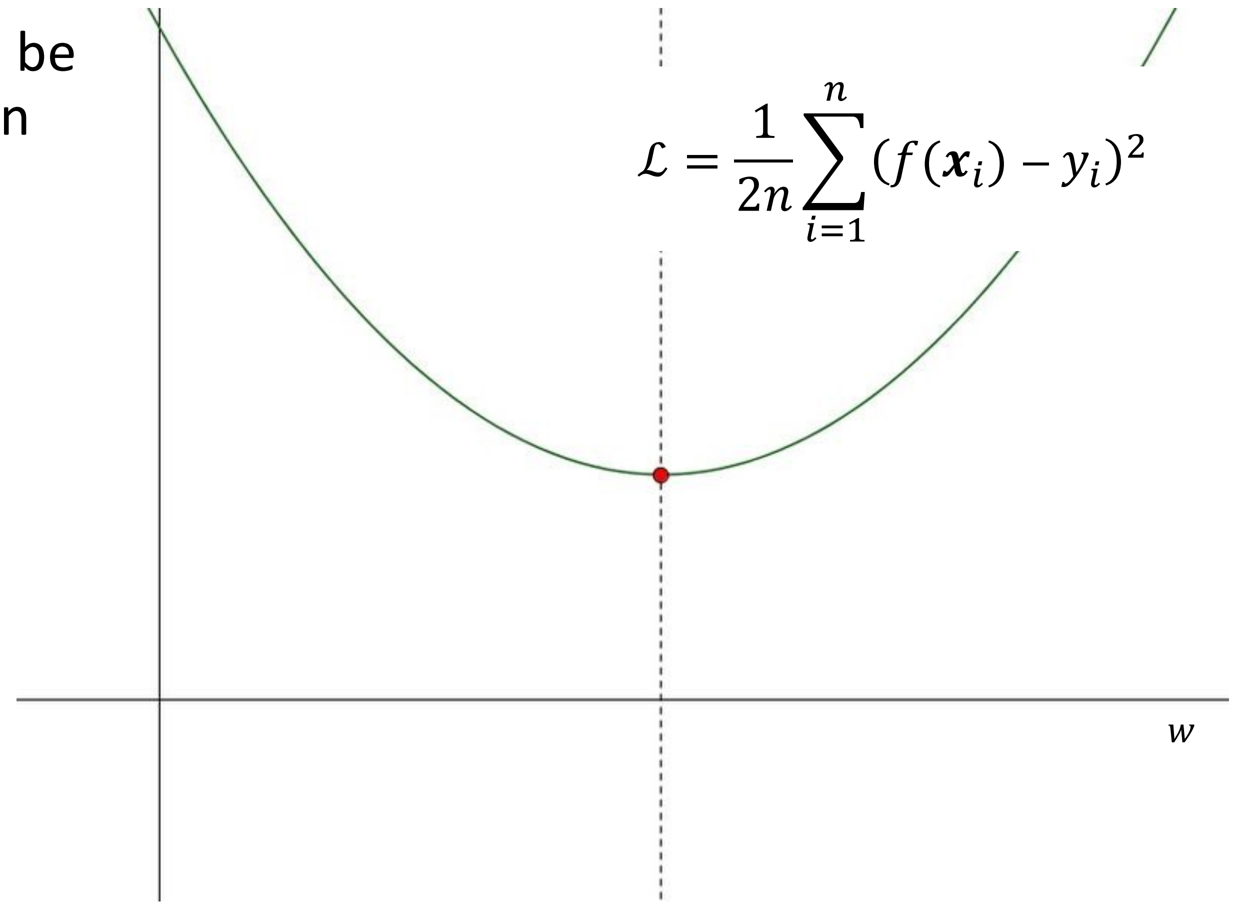
Linear regression. Public domain image, Seewaqu, 2010

MSE = Parabola

A good closed-form solution can be achieved by minimizing the mean squared error,

$$\mathcal{L} = \frac{1}{2n}\sum_{i=1}^{n}(f(\boldsymbol{x}_i) - y_i)^2$$

…where $f(\boldsymbol{x}_i) = \boldsymbol{w}^T \boldsymbol{x}_i$



Gradient descent and SGD

Often, closed-form solution is too computationally expensive. In those situations, we choose a random initial guess for the value of $\boldsymbol{w}$, and then improve it using either gradient descent:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{w}}, \qquad \frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = \frac{1}{n} \sum_{i=1}^{n} \epsilon_i \boldsymbol{x}_i$$

Or stochastic gradient descent:

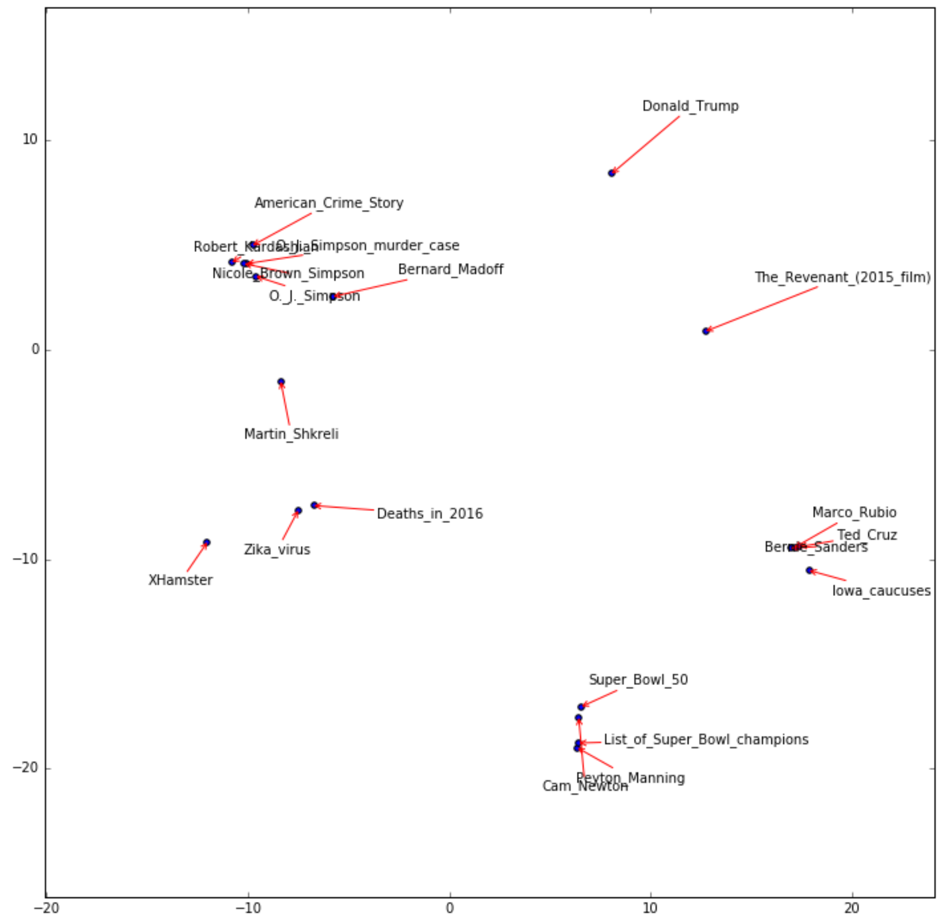
$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \frac{\partial \mathcal{L}_i}{\partial \boldsymbol{w}}, \qquad \frac{\partial \mathcal{L}_i}{\partial \boldsymbol{w}} = \epsilon_i \boldsymbol{x}_i$$

Outline

- “You shall know a word by the company it keeps”
- Linear Regression review
- Dot-product similarity
- Calculating probabilities using the logistic sigmoid
- Derivative of the log sigmoid
- Noise contrastive estimation (NCE)

Words as vectors

- We want a vector space where similar words are located near each other
- Here is a visualization of Word2vec embedding, ewulczyn 2016, https://commons.wikimedia.org/wiki/File:2016_02_mini_embedding.png
- How can we put similar words in the same direction?



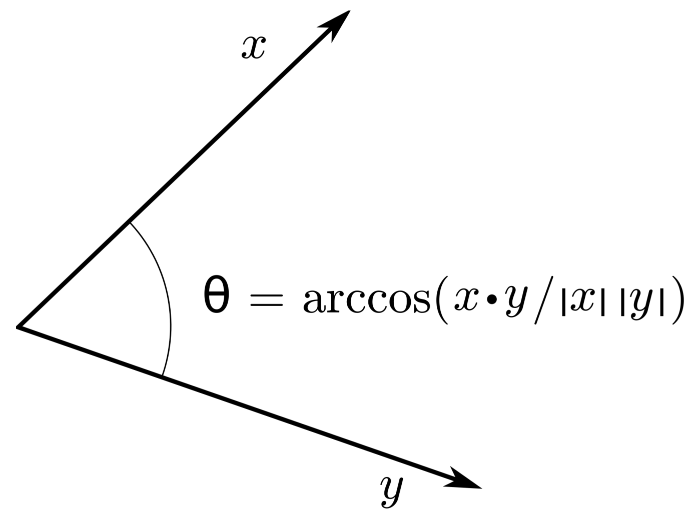
Dot product between vectors

The dot product between two vectors is:

$$\boldsymbol{v}_1^T \boldsymbol{v}_2 = |\boldsymbol{v}_1||\boldsymbol{v}_2| \cos\theta$$

The dot product is a pretty good measure of the similarity between the vectors.

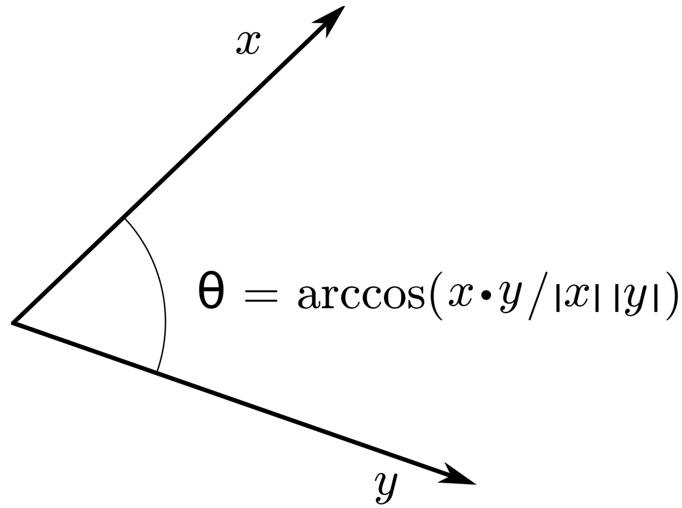
- If $\theta < 90$ degrees, then $\boldsymbol{v}_1^T \boldsymbol{v}_2 > 0$
- If $\theta = 90$ degrees, then $\boldsymbol{v}_1^T \boldsymbol{v}_2 = 0$
- If $\theta > 90$ degrees, then $\boldsymbol{v}_1^T \boldsymbol{v}_2 < 0$



Words as vectors

- Suppose that, for every word in the dictionary w_1, we train a vector $\boldsymbol{v}_1$
- The similarity between two words should be measured by the dot product between their vectors:

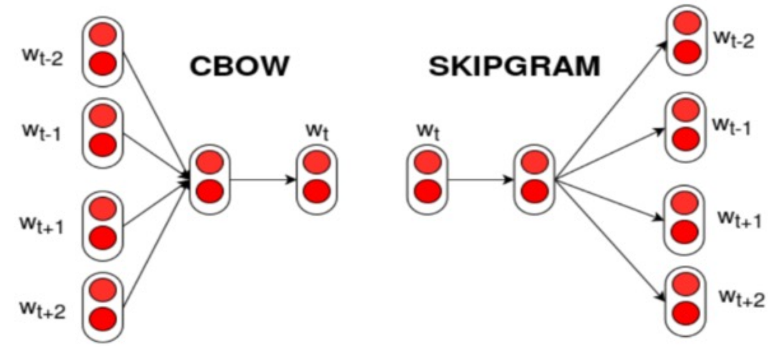
$$s(w_1, w_2) = \boldsymbol{v}_1^T \boldsymbol{v}_2$$



Outline

- "You shall know a word by the company it keeps"
- Linear Regression review
- Dot-product similarity
- Calculating probabilities using the logistic sigmoid
- Derivative of the log sigmoid
- Noise contrastive estimation (NCE)

From similarity to probability



$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} logp(w_t|w_{t+j})$$

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} logp(w_{t+j}|w_t)$$

Suppose

$$s(v_t, u_{t,j}) = \boldsymbol{v}_t^T \boldsymbol{u}_{t,j}$$

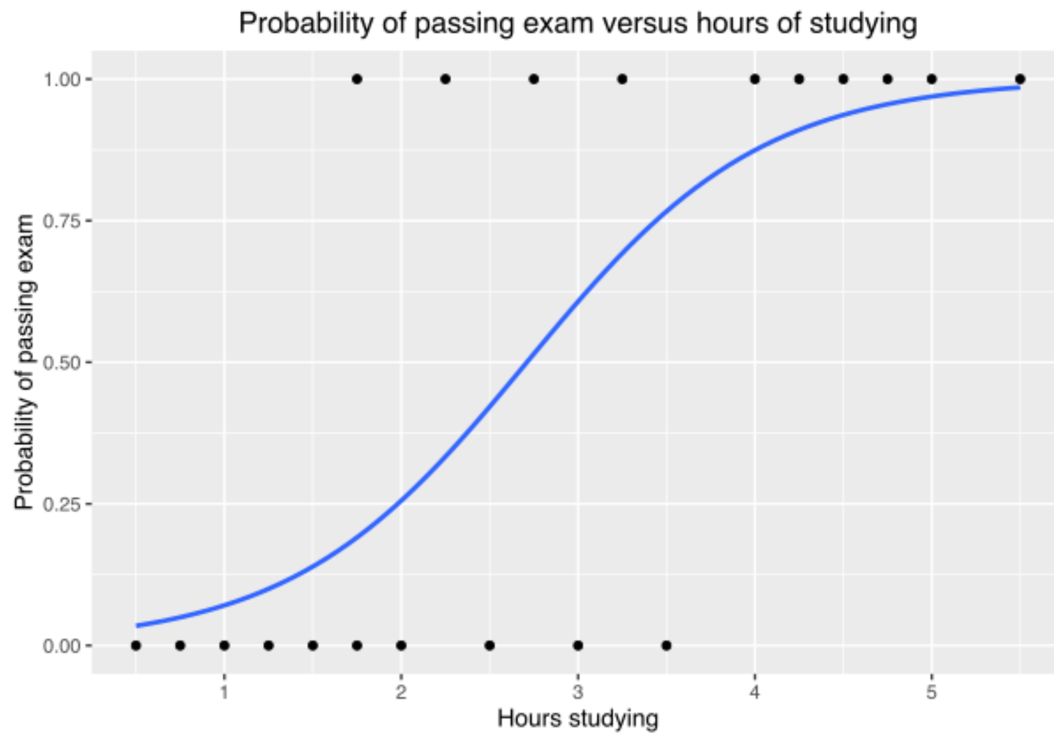
What we want is the probability:

$$\mathcal{L} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{2C} \ln P(u_{t,j}|v_t)$$

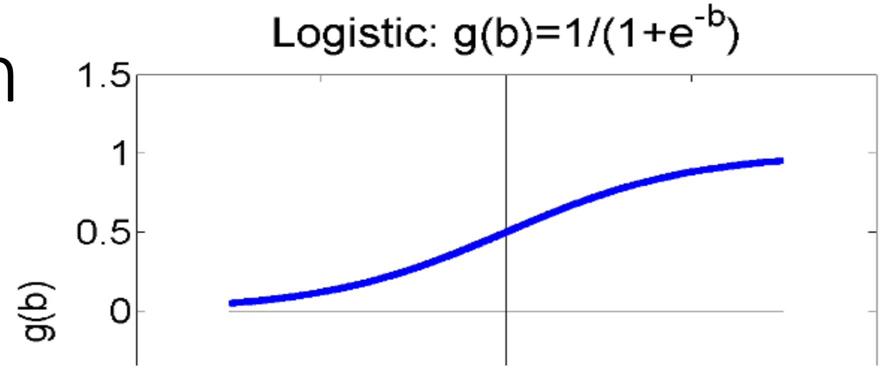
How do we get from similarity to probability?

Logistic regression

- Logistic regression was invented by psychologists in the early 20th century
- They wanted to model binary outcomes, like "Did student i pass or fail the test?" In other words, every output is either $y_i = 1$ or $y_i = 0$
- Instead of modeling $y_i = \boldsymbol{x}_i^T \boldsymbol{w}$ as a real number, it makes more sense to try to model $P(y_i = 1|\boldsymbol{x}_i)$ as some kind of function of $\boldsymbol{x}_i$.



The logistic sigmoid function



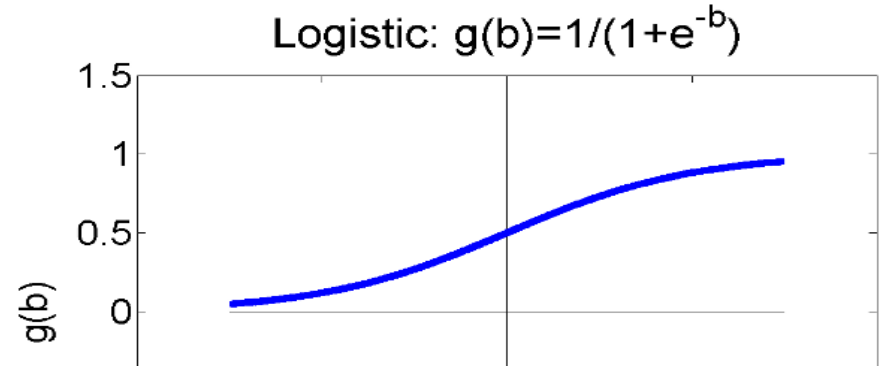
- To model $P(y_i = 1|\boldsymbol{x}_i)$ as some kind of function of $\boldsymbol{x}_i$, we need some kind of nonlinear function that squashes the linear output $\boldsymbol{x}_i^T\boldsymbol{w}$ down to the range $0 \leq f(\boldsymbol{x}_i) \leq 1$.
- Psychologists studied many possibilities, but the one most often used today is the logistic sigmoid function:

$$f(\boldsymbol{x}_i) = \sigma(\boldsymbol{x}_i^T\boldsymbol{w}) = \frac{1}{1 + e^{-\boldsymbol{x}_i^T\boldsymbol{w}}}$$

This function is called sigmoid because it is S-shaped.

$$\sigma(z) = \begin{cases} 1 & z \to \infty \\ 0.5 & z = 0 \\ 0 & z \to -\infty \end{cases}$$

Interpretation as a probability



- Since $0 < f(\boldsymbol{x}) < 1$, we can interpret $f(\boldsymbol{x})$ as a probability
- Specifically, we interpret it as $f(\boldsymbol{x}) = P(Y = 1 | X = \boldsymbol{x})$.

$$f(\boldsymbol{x}) = \sigma(\boldsymbol{x}^T \boldsymbol{w}) = \frac{1}{1 + e^{-\boldsymbol{x}^T \boldsymbol{w}}}$$

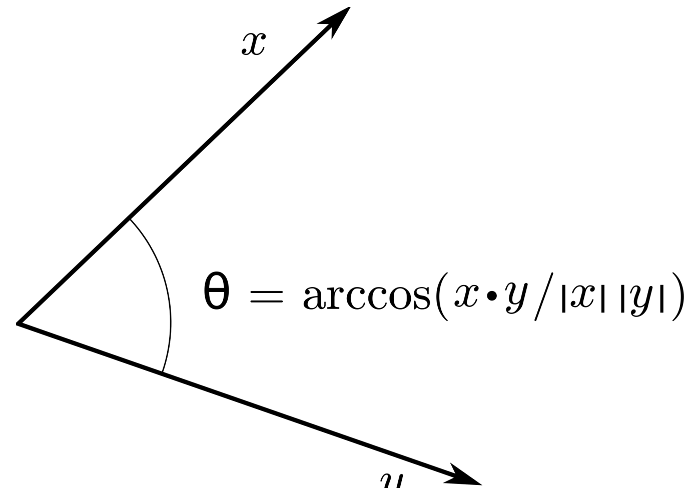
- The argument of the sigmoid, $\boldsymbol{x}^T \boldsymbol{w}$, is called the "logit." Notice that there is a straightforward relationship between the logit and the probability:

$$\sigma(z) = \begin{cases} 1 & z \to \infty \\ 1/2 & z = 0 \\ 0 & z \to -\infty \end{cases}$$

Words as vectors

- Suppose that, for every word in the dictionary w_1, we train a vector $\boldsymbol{v}_1$
- The similarity between two words should be measured by the dot product between their vectors:

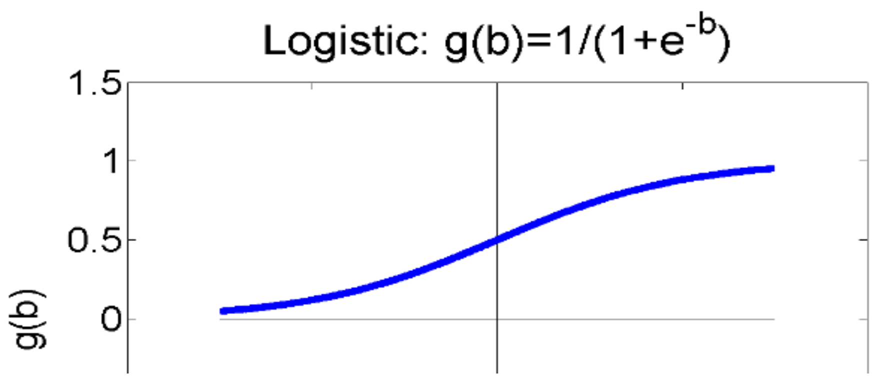
$$s(v_t, u_{t,j}) = \boldsymbol{v}_t^T \boldsymbol{u}_{t,j}$$



By BenFrantzDale at the English Wikipedia, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=49972362

The probability of one word, given the other, is the sigmoid of their dot product:

$$P(u_{t,j}|v_t) = \frac{1}{1 + e^{-\boldsymbol{v}_t^T \boldsymbol{u}_{t,j}}}$$

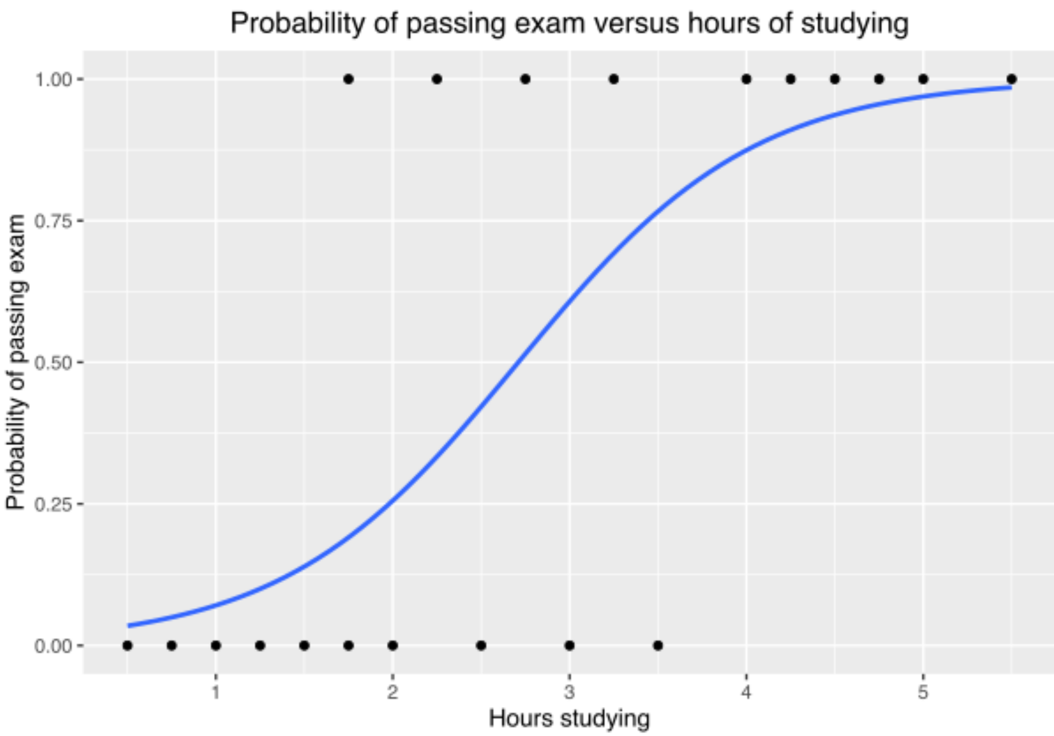


Example

For example, the blue line here shows

$$P(Y=1|x) = \sigma(3x - 2.75)$$

$$= \begin{cases} 1 & x \to \infty \\ 1/2 & x = 2.75 \\ 0 & x \to -\infty \end{cases}$$



Outline

- “You shall know a word by the company it keeps”
- Linear Regression review
- Dot-product similarity
- Calculating probabilities using the logistic sigmoid
- Derivative of the log sigmoid
- Noise contrastive estimation (NCE)

Maximize probability of the training dataset

- We want to minimize the negative log probability of the training data:

$$\mathcal{L} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{2C}\ln\sigma(\boldsymbol{v}_t^T\boldsymbol{u}_{t,j})$$

1. Initialize: choose the vectors $\boldsymbol{u}_{t,j}$ and $\boldsymbol{v}_t$ to be small random vectors
2. Gradient descent: update them according to

$$\boldsymbol{v}_t \leftarrow \boldsymbol{v}_t - \eta\frac{\partial\mathcal{L}}{\partial\boldsymbol{v}_t}, \qquad \boldsymbol{u}_{t,j} \leftarrow \boldsymbol{u}_{t,j} - \eta\frac{\partial\mathcal{L}}{\partial\boldsymbol{u}_{t,j}}$$

3. Repeat step 2 until convergence!

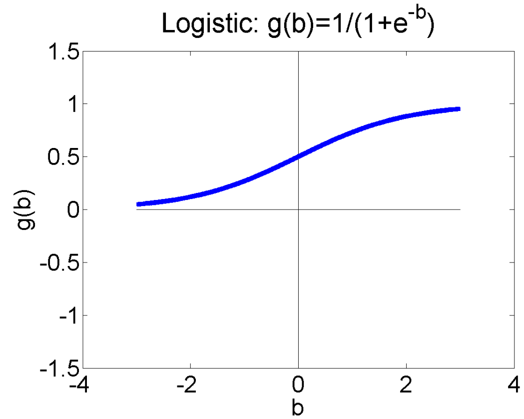
What's the gradient?

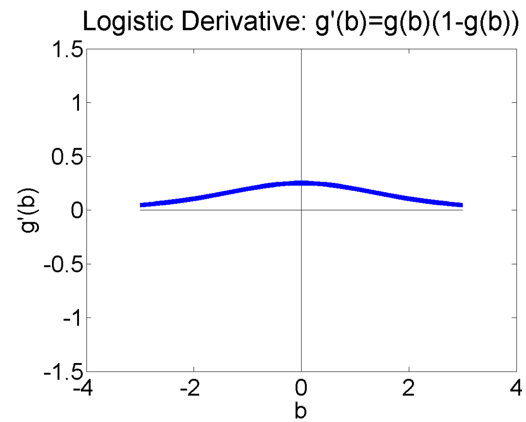
$$\mathcal{L} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{2C} \ln \sigma(\boldsymbol{v}_t^T \boldsymbol{u}_{t,j})$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{v}_t} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{2C} \frac{\partial}{\partial \boldsymbol{v}_t} \ln \sigma(\boldsymbol{v}_t^T \boldsymbol{u}_{t,j})$$

$$= -\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{2C} \frac{1}{\sigma\left(\boldsymbol{v}_t^T \boldsymbol{u}_{t,j}\right)} \frac{\partial}{\partial \boldsymbol{v}_t} \sigma\left(\boldsymbol{v}_t^T \boldsymbol{u}_{t,j}\right) = ?$$

OK, what's $\frac{\partial \sigma(\boldsymbol{v}_t^T \boldsymbol{u}_{t,j})}{\partial \boldsymbol{v}_t}$?





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

$$\frac{\partial \sigma}{\partial z} = \left(-\frac{1}{(1 + e^{-z})^2}\right)(-e^{-z}) = \left(\frac{1}{1 + e^{-z}}\right)\left(\frac{e^{-z}}{1 + e^{-z}}\right)$$

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

What's the gradient?

$$\mathcal{L} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{2C} \ln \sigma(\boldsymbol{v}_t^T \boldsymbol{u}_{t,j})$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{v}_t} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{2C} \frac{1}{\sigma\left(\boldsymbol{v}_t^T \boldsymbol{u}_{t,j}\right)} \times \sigma\left(\boldsymbol{v}_t^T \boldsymbol{u}_{t,j}\right)\left(1-\sigma\left(\boldsymbol{v}_t^T \boldsymbol{u}_{t,j}\right)\right) \times \frac{\partial \boldsymbol{v}_t^T \boldsymbol{u}_{t,j}}{\partial \boldsymbol{v}_t}$$

$$= -\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{2C}\left(1-\sigma\left(\boldsymbol{v}_t^T \boldsymbol{u}_{t,j}\right)\right)\boldsymbol{u}_{t,j}$$

Training the word vectors

So, if we want to maximize the probability of the training data, the word vector $\boldsymbol{v}_t$ gets updated as:

$$\boldsymbol{v}_t \leftarrow \boldsymbol{v}_t - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{v}_t} = \boldsymbol{v}_t + \frac{\eta}{T} \sum_{w_t = w} \sum_{j=1}^{2C} \left(1 - \sigma(\boldsymbol{u}_{t,j}^T \boldsymbol{v}_t)\right) \boldsymbol{u}_{t,j}$$

- In other words, $\boldsymbol{v}_t$ becomes a weighted average of the vectors $\boldsymbol{u}_{t,j}$ for words that occur in the context of $\boldsymbol{v}_t$.
- The vectors start out random, then vectors that frequently co-occur get pushed together, to become more similar

Cross-entropy

This loss function:

$$\mathcal{L}_i = -\ln \Pr(Y = y_i | \boldsymbol{x}_i)$$

is called cross-entropy. The term comes from physics, where "entropy" is our degree of uncertainty about whether or not something will happen.

Outline

- “You shall know a word by the company it keeps”
- Linear Regression review
- Dot-product similarity
- Calculating probabilities using the logistic sigmoid
- Derivative of the log sigmoid
- Noise contrastive estimation (NCE)

Training the neural net

- This formula has a problem:

$$\boldsymbol{v} \leftarrow \boldsymbol{v} + \frac{\eta}{T} \sum_{w_t = w} \sum_{j=1}^{2c} \left(1 - \sigma(\boldsymbol{u}_{t,j}^T \boldsymbol{v})\right) \boldsymbol{u}_{t,j}$$

- The problem is that $\left(1 - \sigma(\boldsymbol{u}_{t,j}^T \boldsymbol{v})\right)$ is always positive
- Therefore, $\boldsymbol{v}$ gets gradually bigger and bigger – it never stops growing!!

Fixing the problem

To keep all the vectors from growing forever, we need to subtract something from the right-hand side:

$$\boldsymbol{v} \leftarrow \boldsymbol{v} + \frac{\eta}{T} \sum_{w_t = w} \sum_{j=1}^{2C} \left(1 - \sigma(\boldsymbol{u}_{t,j}^T \boldsymbol{v})\right) \boldsymbol{u}_{t,j} - SOMETHING$$

…such that, on average, the update will be zero (keeping all vectors small).

- If we still want to say that

$$\boldsymbol{v} \leftarrow \boldsymbol{v} - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{v}}$$

- …then we need to add something to $\mathcal{L}$ whose derivative will be "SOMETHING"

Noise contrastive estimation

So far, we've said that the probability of $\boldsymbol{u}_{t,j}$ occurring in the context of $\boldsymbol{v}_t$ is:

$$P(\boldsymbol{u}_{t,j}|\boldsymbol{v}_t) = \sigma(\boldsymbol{u}_{t,j}^T \boldsymbol{v}_t)$$

For some randomly chosen "noise" sample, $\boldsymbol{n}_i$, the probability it DOESN'T occur in the context of $\boldsymbol{v}_t$ is:

$$1 - P(\boldsymbol{n}_i|\boldsymbol{v}_t) = 1 - \sigma(\boldsymbol{n}_i^T \boldsymbol{v}_t)$$

We can create a well-normalized loss function by adding these two log probabilities:

$$\mathcal{L} = -\ln \sigma\left(\boldsymbol{u}_{t,j}^T \boldsymbol{v}_t\right) - \ln\left(1 - \sigma(\boldsymbol{n}_i^T \boldsymbol{v}_t)\right)$$

Skipgram NCE

Skigram NCE uses the following update step:

1. Choose a pair $\boldsymbol{v}_t$ and $\boldsymbol{u}_{t,j}$ from the training data
2. Choose k different "noise" examples, $\boldsymbol{n}_i$, uniformly at random from the vocabulary
3. Update $\boldsymbol{v}_t$ according to

$$\mathcal{L} = -\ln \sigma\left(\boldsymbol{u}_{t,j}^T \boldsymbol{v}_t\right) - \frac{1}{k} \sum_{i=1}^{k} \ln\left(1 - \sigma(\boldsymbol{n}_i^T \boldsymbol{v}_t)\right)$$

$$\boldsymbol{v}_t \leftarrow \boldsymbol{v}_t - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{v}_t}$$

Try the quiz!

- Go to PrairieLearn, try the quiz!

Conclusions

- Logistic regression

$$f(\boldsymbol{x}_i) = \sigma(\boldsymbol{x}_i^T \boldsymbol{w}) = \frac{1}{1 + e^{-\boldsymbol{x}_i^T \boldsymbol{w}}}$$

- Derivative of the sigmoid

$$\frac{\partial \sigma(z)}{\partial z} = \sigma(z)(1 - \sigma(z))$$

- Dot-product similarity:

$$\mathcal{L} = -\frac{1}{T} \sum_{t=1}^{T} \sum_{j=1}^{2C} \ln \sigma(\boldsymbol{u}_{t,j}^T \boldsymbol{v}_t)$$

- Skip-gram noise contrastive estimation:

$$\mathcal{L} = -\ln \sigma\left(\boldsymbol{u}_{t,j}^T \boldsymbol{v}_t\right) - \frac{1}{k} \sum_{i=1}^{k} \ln\left(1 - \sigma(\boldsymbol{n}_i^T \boldsymbol{v}_t)\right)$$