# Deep Learning 101

## CSCI 1460: Computational Linguistics
## Lecture 8

**Ellie Pavlick**
**Fall 2023**

# Announcements

- Assignment 2 fixes (tf-idf vs. BOW)
- NLP talks in the department!
    - **Oct 13th, 12pm Panel/Discussion** Strong vs. Weak Compositionality in Humans and Machines!

# Topics

- More Followup on Word Embeddings from SVD
- Logistic Regression and Gradient Descent
- Multilayer Perceptrons
- Word Embeddings from NNs

# Topics

- **More Followup on Word Embeddings from SVD**
- Logistic Regression and Gradient Descent
- Multilayer Perceptrons
- Word Embeddings from NNs

# SVD Revisited

The below figure shows the following: (Part of) a term-document matrix M; A V matrix that results when running LSA on M; An embedding (i.e., row of the U matrix) associated with a document d. Which of the below represents the most likely content of document d? (Note that document d is not supposed to be one of the docs doc1, doc2, doc3, doc4, doc5 along the rows of M. You can assume d is a different document that also occurred in M but is not shown in the below figure.) *

| | red | green | apple | kiwi |
|---|---|---|---|---|
| doc1 | 1 | 1 | 1 | 1 |
| doc2 | 1 | 0 | 1 | 0 |
| doc3 | 1 | 1 | 0 | 1 |
| doc4 | 1 | 0 | 1 | 1 |
| doc5 | 0 | 0 | 1 | 1 |

M

| | | |
|---|---|---|
| 10 | 1 | 0 |
| 8 | 0 | 8 |
| 1 | 6 | 1 |
| -1 | 7 | 11 |

V

| | | | |
|---|---|---|---|
| d | 1 | 2 | 12 |

# SVD Revisited

The below figure shows the following: (Part of) a term-document matrix M; A V matrix that results when running LSA on M; An embedding (i.e., row of the U matrix) associated with a document d. Which of the below represents the most likely content of document d? (Note that document d is not supposed to be one of the docs doc1, doc2, doc3, doc4, doc5 along the rows of M. You can assume d is a different document that also occurred in M but is not shown in the below figure.) *

| | red | green | apple | kiwi |
|---|---|---|---|---|
| doc1 | 1 | 1 | 1 | 1 |
| doc2 | 1 | 0 | 1 | 0 |
| doc3 | 1 | 1 | 0 | 1 |
| doc4 | 1 | 0 | 1 | 1 |
| doc5 | 0 | 0 | 1 | 1 |

M

| | | |
|---|---|---|
| 10 | 1 | 0 |
| 8 | 0 | 8 |
| 1 | 6 | 1 |
| -1 | 7 | 11 |

V

| | | | |
|---|---|---|---|
| d | 1 | 2 | 12 |

# SVD Revisited

The below figure shows the following: (Part of) a term-document matrix M; A V matrix that results when running LSA on M; An embedding (i.e., row of the U matrix) associated with a document d. Which of the below represents the most likely content of document d? (Note that document d is not supposed to be one of the docs doc1, doc2, doc3, doc4, doc5 along the rows of M. You can assume d is a different document that also occurred in M but is not shown in the below figure.) *

| | red | green | apple | kiwi |
|---|---|---|---|---|
| doc1 | 1 | 1 | 1 | 1 |
| doc2 | 1 | 0 | 1 | 0 |
| doc3 | 1 | 1 | 0 | 1 |
| doc4 | 1 | 0 | 1 | 1 |
| doc5 | 0 | 0 | 1 | 1 |

M

| | | |
|---|---|---|
| 10 | 1 | 0 |
| 8 | 0 | 8 |
| 1 | 6 | 1 |
| -1 | 7 | 11 |

V

| | | | |
|---|---|---|---|
| d | 1 | 2 | 12 |

# SVD Revisited

The below figure shows the following: (Part of) a term-document matrix M; A V matrix that results when running LSA on M; An embedding (i.e., row of the U matrix) associated with a document d. Which of the below represents the most likely content of document d? (Note that document d is not supposed to be one of the docs doc1, doc2, doc3, doc4, doc5 along the rows of M. You can assume d is a different document that also occurred in M but is not shown in the below figure.) *

| | red | green | apple | kiwi |
|---|---|---|---|---|
| doc1 | 1 | 1 | 1 | 1 |
| doc2 | 1 | 0 | 1 | 0 |
| doc3 | 1 | 1 | 0 | 1 |
| doc4 | 1 | 0 | 1 | 1 |
| doc5 | 0 | 0 | 1 | 1 |

M

| | | |
|---|---|---|
| 10 | 1 | 0 |
| 8 | 0 | 8 |
| 1 | 6 | 1 |
| -1 | 7 | 11 |

V

| | | | |
|---|---|---|---|
| d | 1 | 2 | 12 |

# SVD Revisited

The below figure shows the following: (Part of) a term-document matrix M; A V matrix that results when running LSA on M; An embedding (i.e., row of the U matrix) associated with a document d. Which of the below represents the most likely content of document d? (Note that document d is not supposed to be one of the docs doc1, doc2, doc3, doc4, doc5 along the rows of M. You can assume d is a different document that also occurred in M but is not shown in the below figure.) *

| | red | green | apple | kiwi |
|---|---|---|---|---|
| doc1 | 1 | 1 | 1 | 1 |
| doc2 | 1 | 0 | 1 | 0 |
| doc3 | 1 | 1 | 0 | 1 |
| doc4 | 1 | 0 | 1 | 1 |
| doc5 | 0 | 0 | 1 | 1 |

M

| | | |
|---|---|---|
| 10 | 1 | 0 |
| 8 | 0 | 8 |
| 1 | 6 | 1 |
| -1 | 7 | 11 |

V

| | | | |
|---|---|---|---|
| d | 1 | 2 | 12 |

# SVD Revisited

The below figure shows the following: (Part of) a term-document matrix M; A V matrix that results when running LSA on M; An embedding (i.e., row of the U matrix) associated with a document d. Which of the below represents the most likely content of document d? (Note that document d is not supposed to be one of the docs doc1, doc2, doc3, doc4, doc5 along the rows of M. You can assume d is a different document that also occurred in M but is not shown in the below figure.) *

| | red | green | apple | kiwi |
|---|---|---|---|---|
| doc1 | 1 | 1 | 1 | 1 |
| doc2 | 1 | 0 | 1 | 0 |
| doc3 | 1 | 1 | 0 | 1 |
| doc4 | 1 | 0 | 1 | 1 |
| doc5 | 0 | 0 | 1 | 1 |

M

| | | |
|---|---|---|
| 10 | 1 | 0 |
| 8 | 0 | 8 |
| 1 | 6 | 1 |
| -1 | 7 | 11 |

V

| | | | |
|---|---|---|---|
| d | 1 | 2 | 12 |

# SVD Revisited

The below figure shows the following: (Part of) a term-document matrix M; A V matrix that results when running LSA on M; An embedding (i.e., row of the U matrix) associated with a document d. Which of the below represents the most likely content of document d? (Note that document d is not supposed to be one of the docs doc1, doc2, doc3, doc4, doc5 along the rows of M. You can assume d is a different document that also occurred in M but is not shown in the below figure.) *

| | red | green | apple | kiwi |
|---|---|---|---|---|
| doc1 | 1 | 1 | 1 | 1 |
| doc2 | 1 | 0 | 1 | 0 |
| doc3 | 1 | 1 | 0 | 1 |
| doc4 | 1 | 0 | 1 | 1 |
| doc5 | 0 | 0 | 1 | 1 |

M

| | | |
|---|---|---|
| 10 | 1 | 0 |
| 8 | 0 | 8 |
| 1 | 6 | 1 |
| -1 | 7 | 11 |

V

| | | | |
|---|---|---|---|
| d | 1 | 2 | 12 |

# SVD Revisited

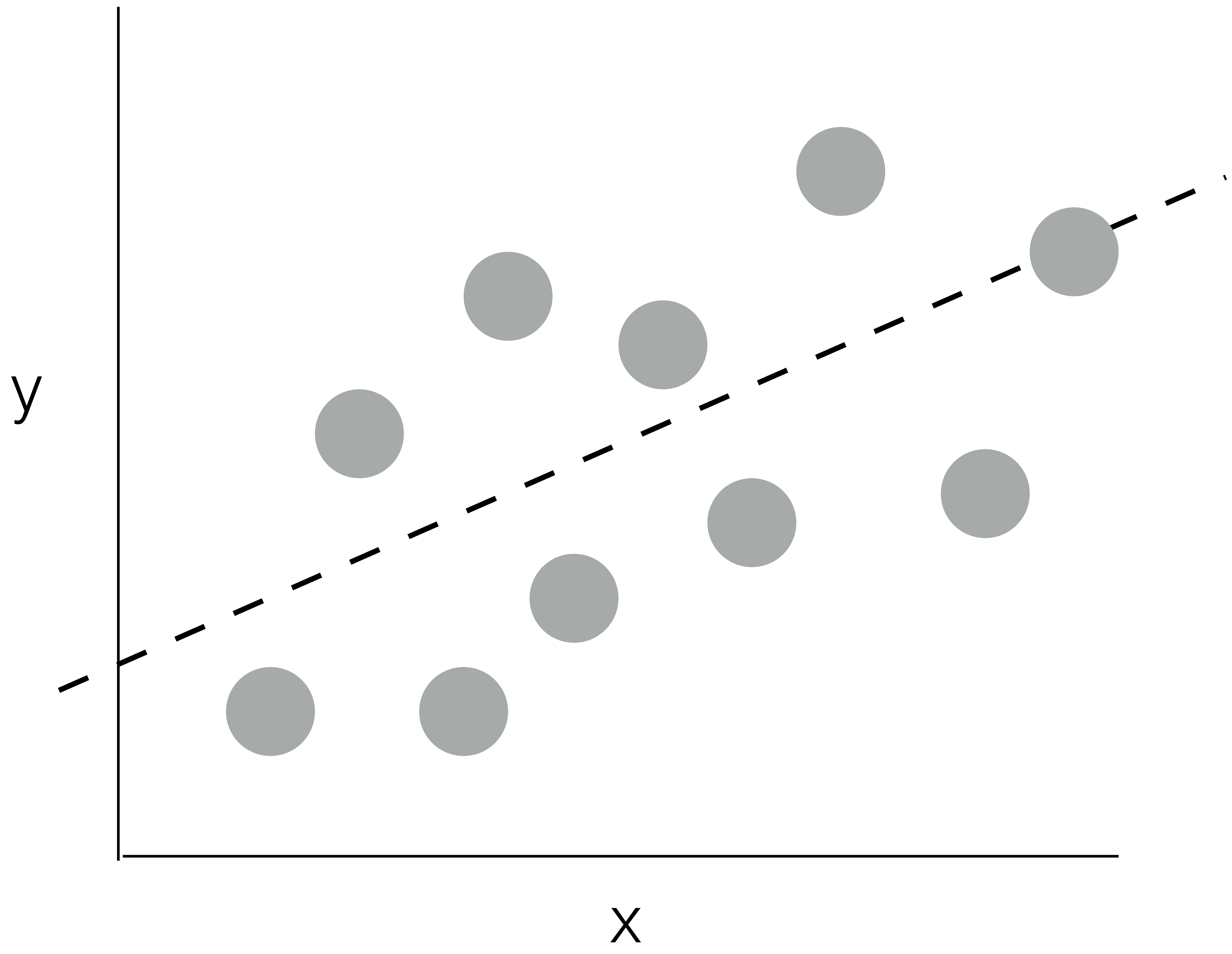
Colab Notebook

# Topics

- More Followup on Word Embeddings from SVD
- **Logistic Regression and Gradient Descent**
- Multilayer Perceptrons
- Word Embeddings from NNs

# Logistic Regression Classifiers

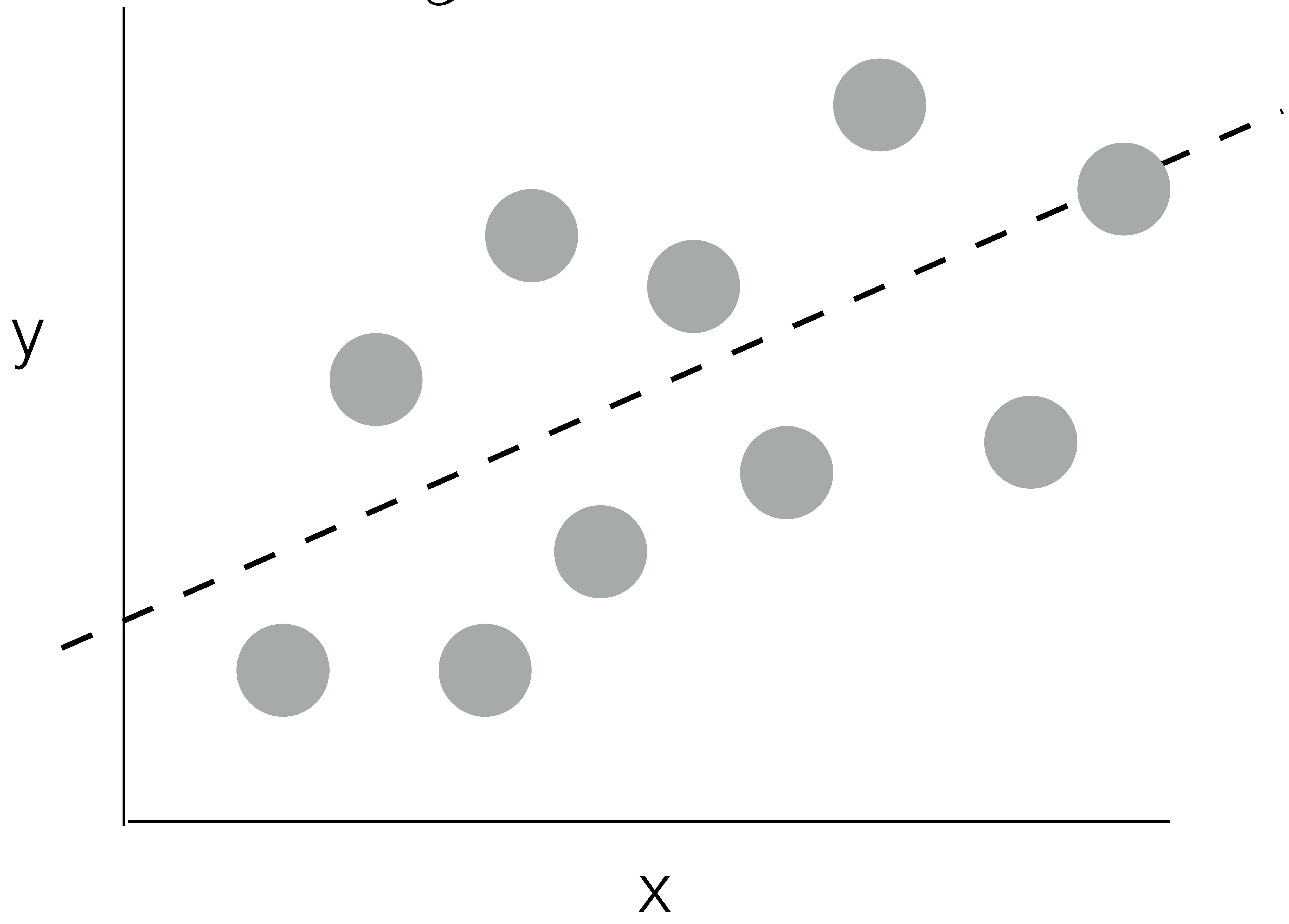
## Making Predictions



# Logistic Regression Classifiers

## Making Predictions

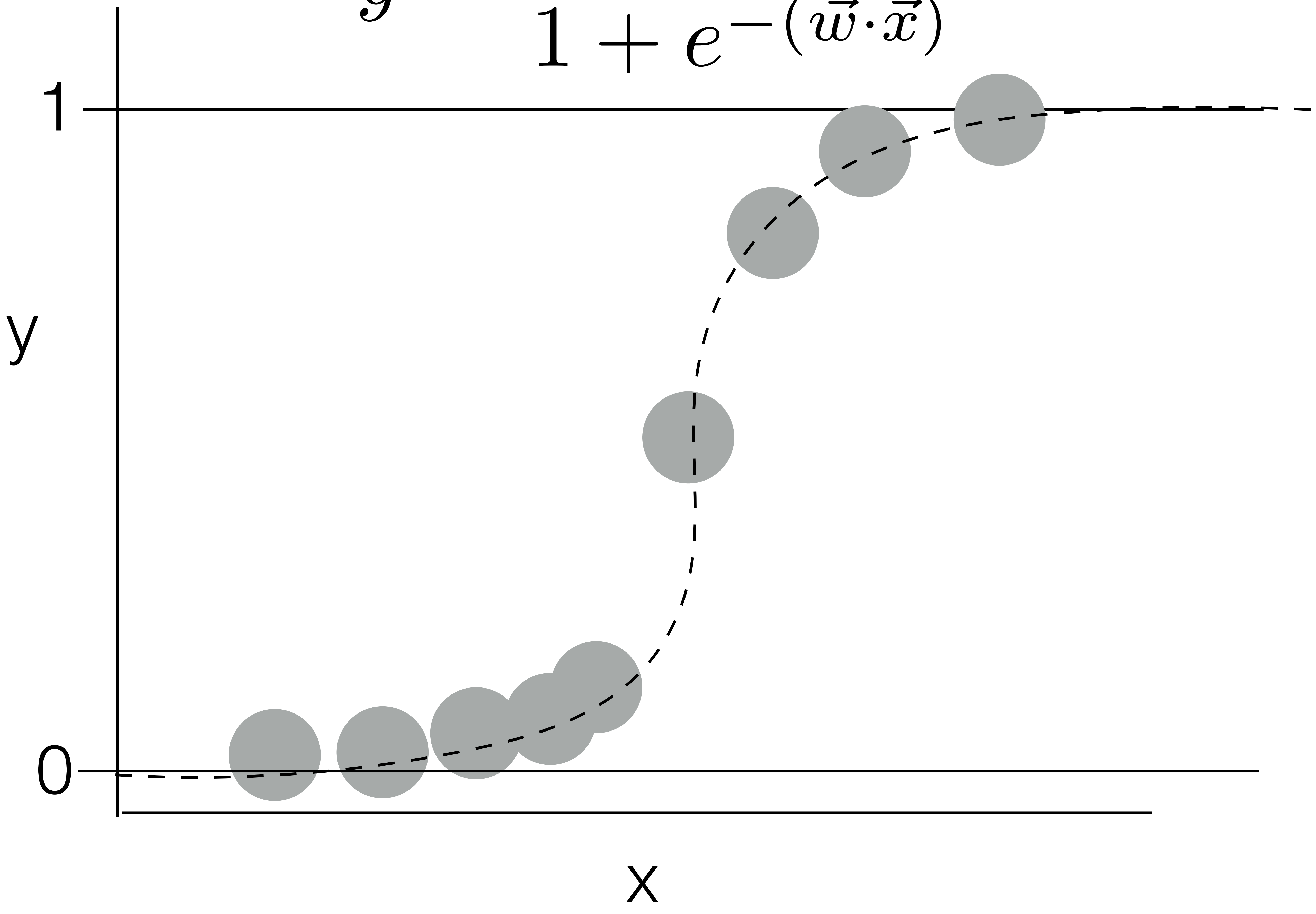
$$y = \vec{w} \cdot \vec{x}$$



# Logistic Regression Classifiers

## Making Predictions

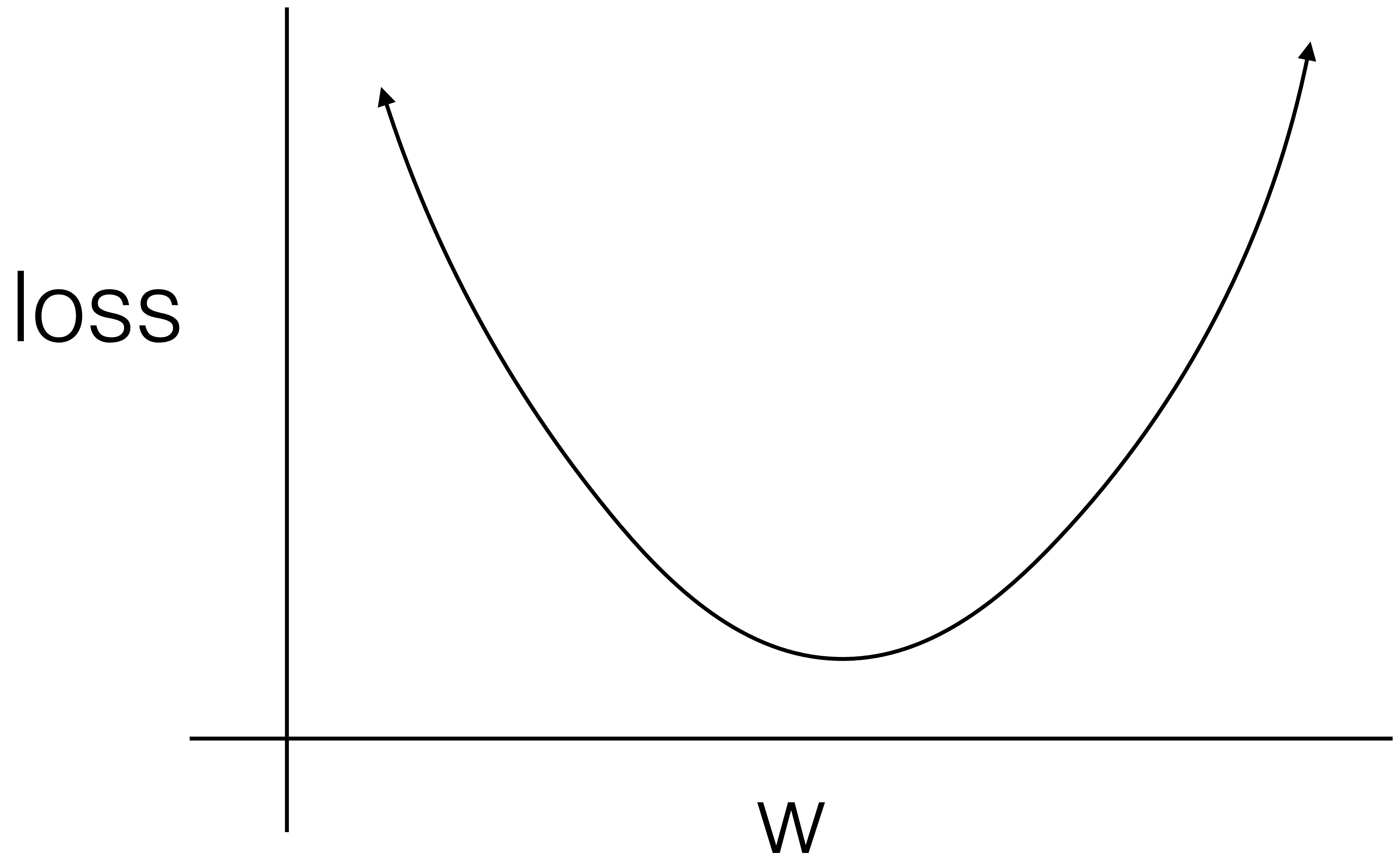
$$y = \frac{1}{1 + e^{-(\vec{w} \cdot \vec{x})}}$$



# Logistic Regression Classifiers

## Training with Gradient Descent

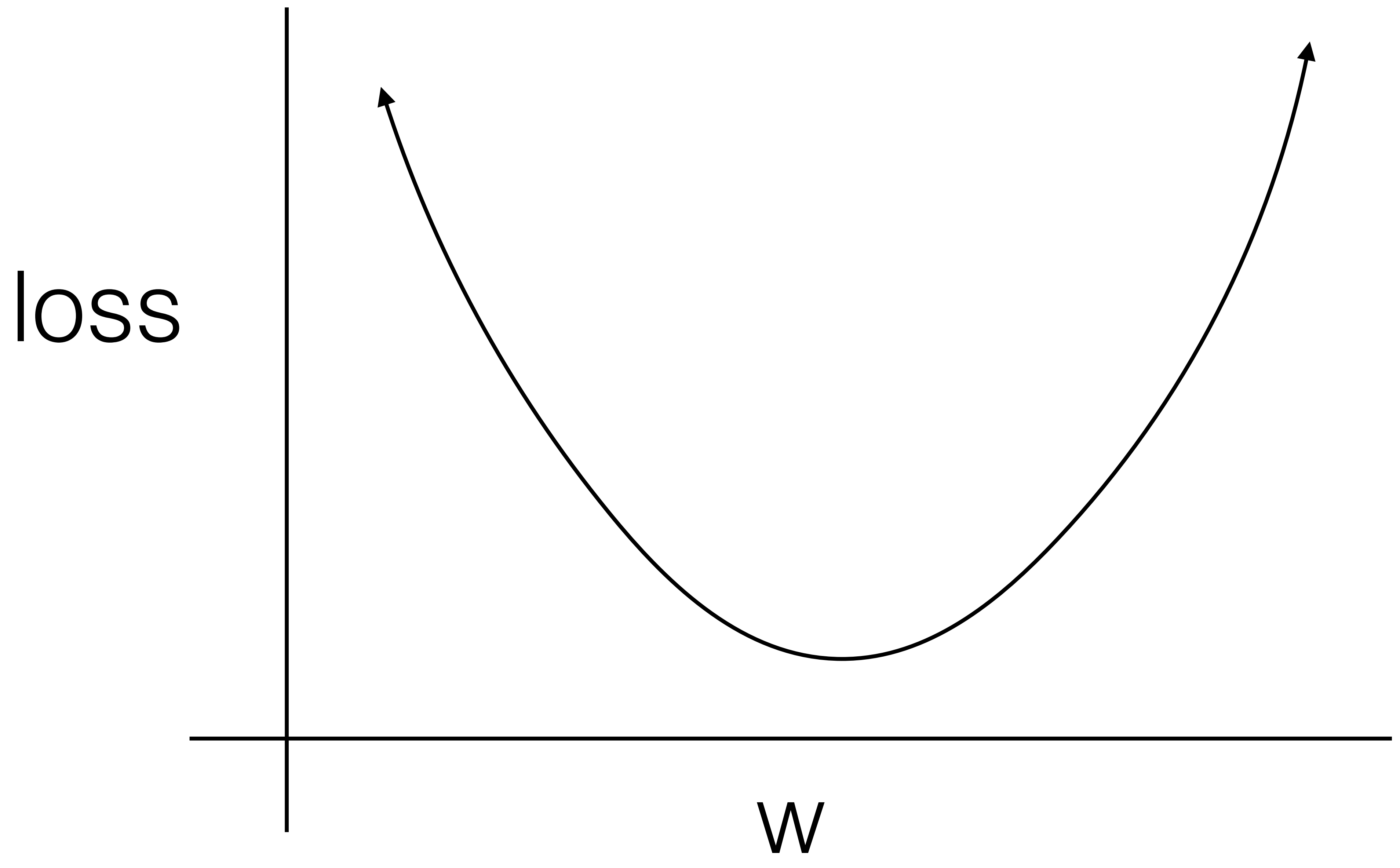
minimize loss(data, w)



# Logistic Regression Classifiers

## Training with Gradient Descent

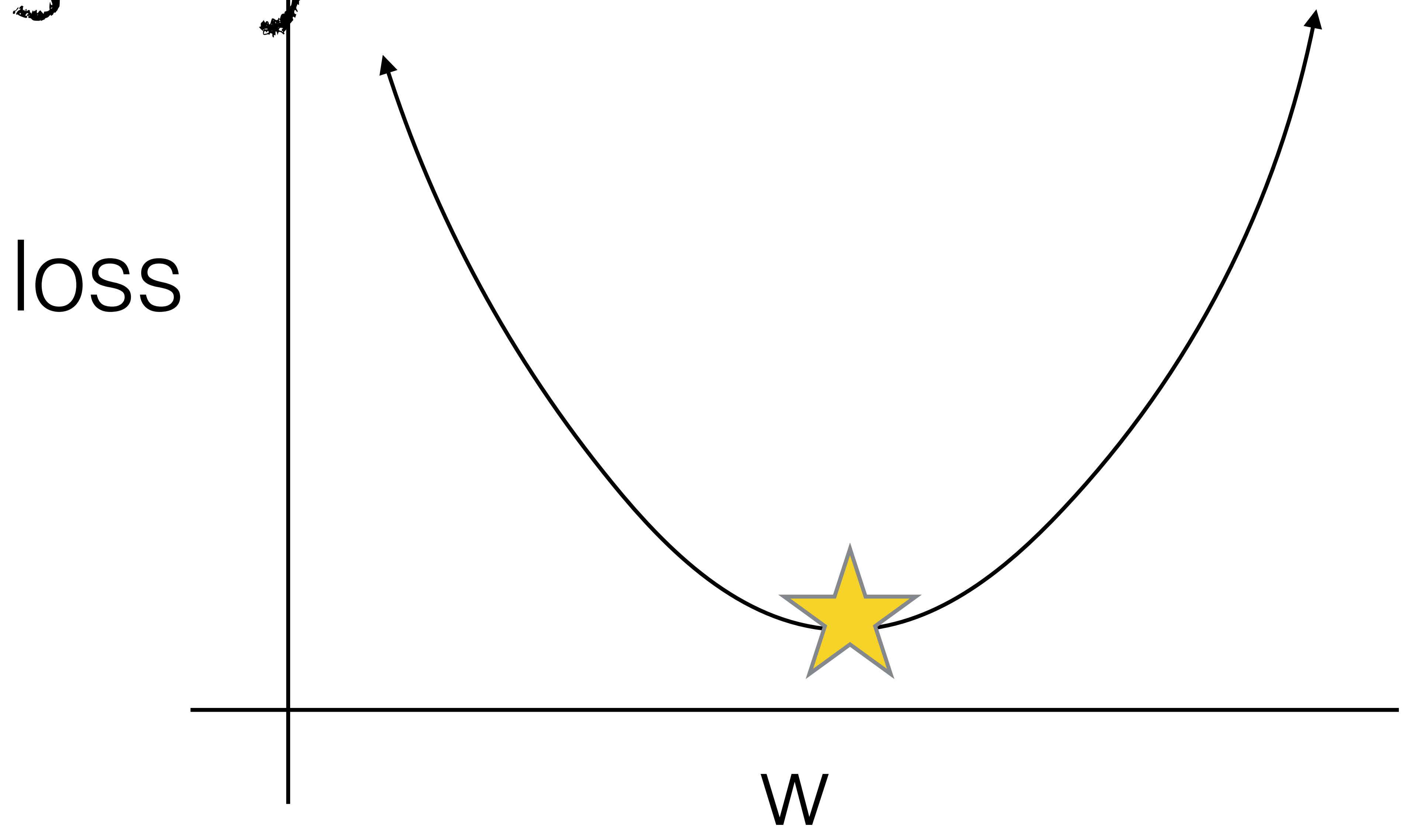
$$-Y log\hat{Y} + (1-Y)log(1-\hat{Y})$$



# Logistic Regression Classifiers

## Training with Gradient Descent

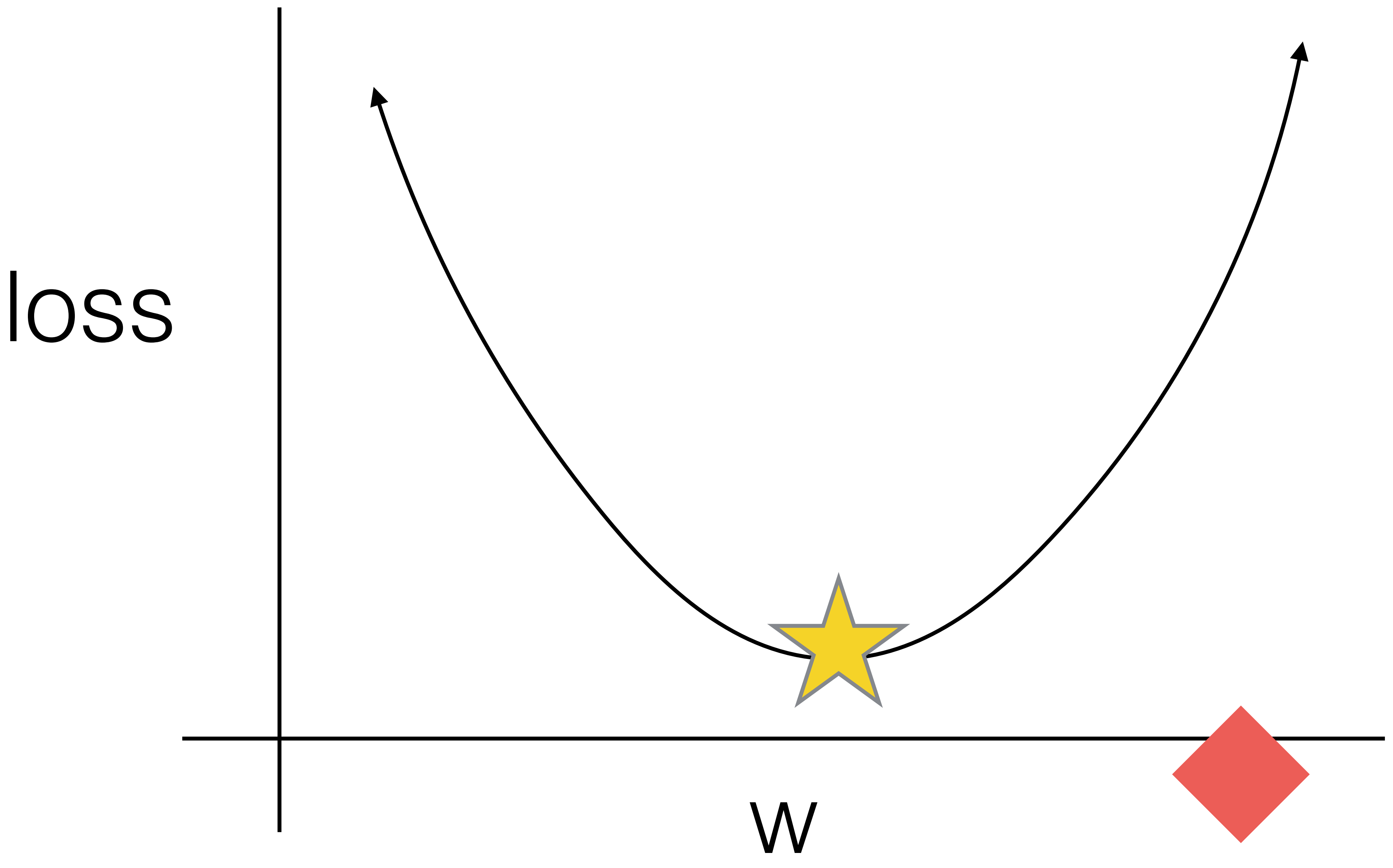
Goal (lowest achievable value for loss given data). You dont know what value of parameters will give you this.



# Logistic Regression Classifiers

## Training with Gradient Descent

$$-Y log\hat{Y} + (1 - Y)log(1 - \hat{Y})$$

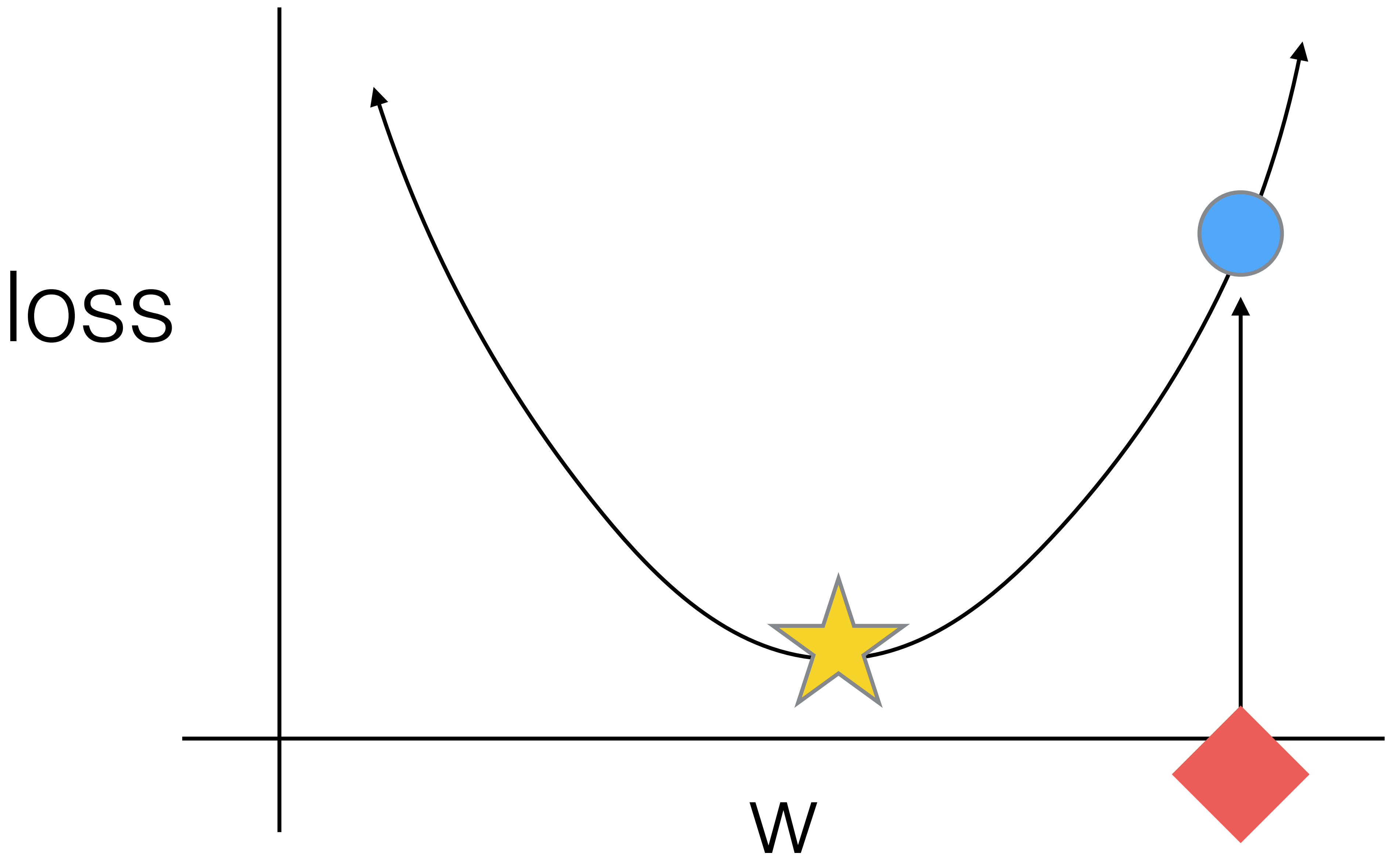



current model parameters (usually start with random parameters)

# Logistic Regression Classifiers

## Training with Gradient Descent

$$-Y log\hat{Y} + (1 - Y)log(1 - \hat{Y})$$

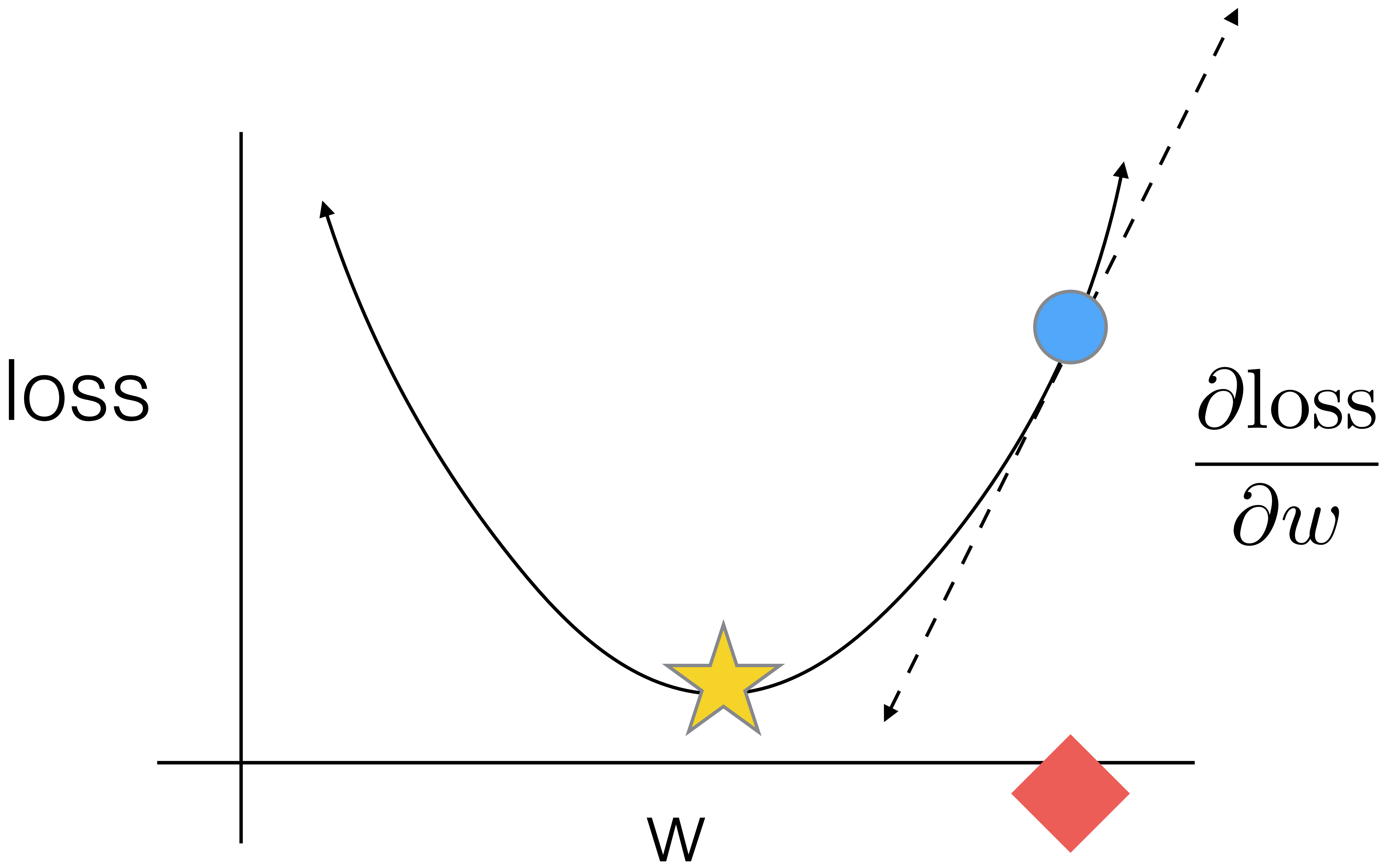



loss given data
and current
model
parameters

# Logistic Regression Classifiers

## Training with Gradient Descent

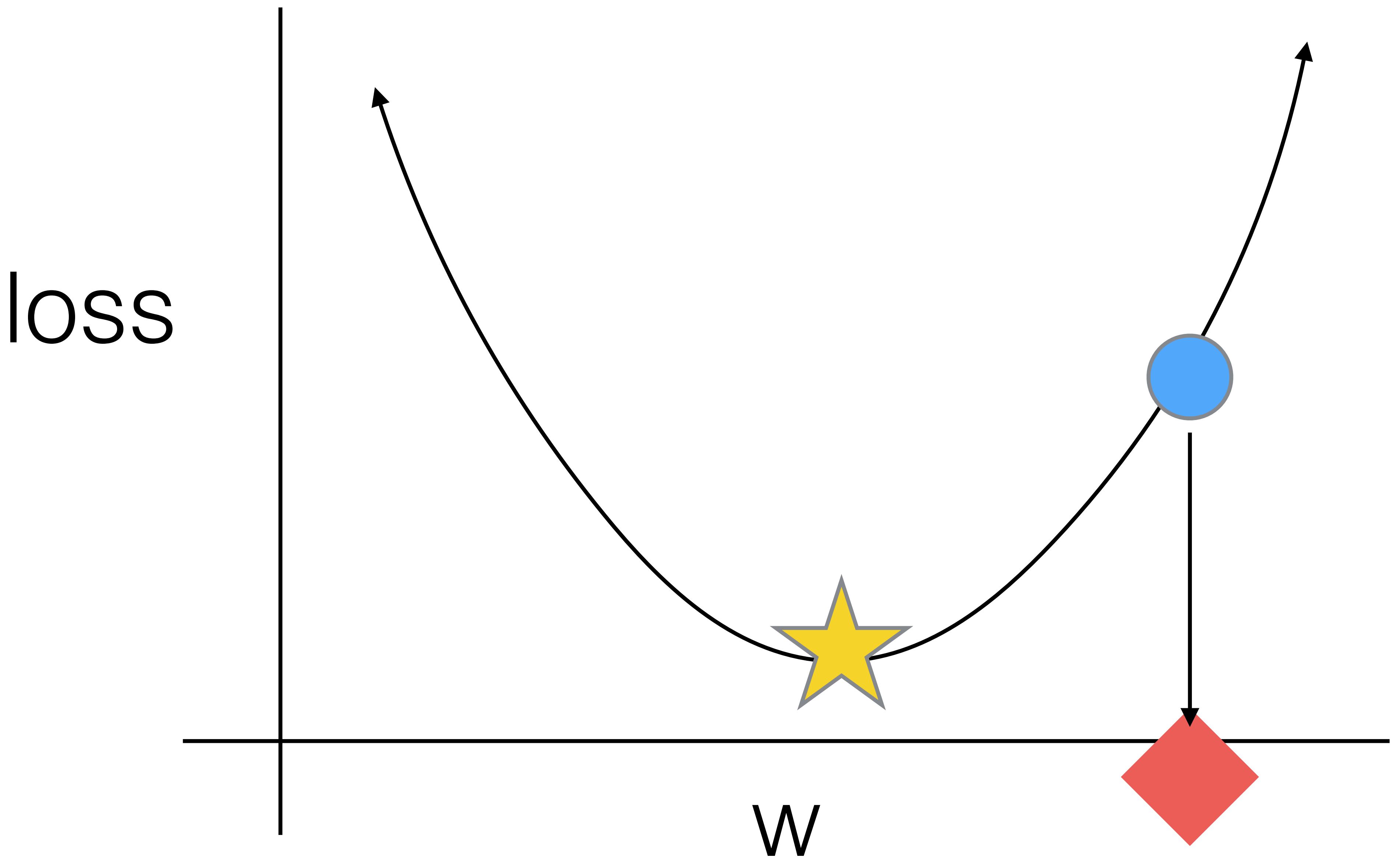
$$-Y log\hat{Y} + (1 - Y)log(1 - \hat{Y})$$



# Logistic Regression Classifiers

## Training with Gradient Descent

$$-Y log\hat{Y} + (1 - Y)log(1 - \hat{Y})$$

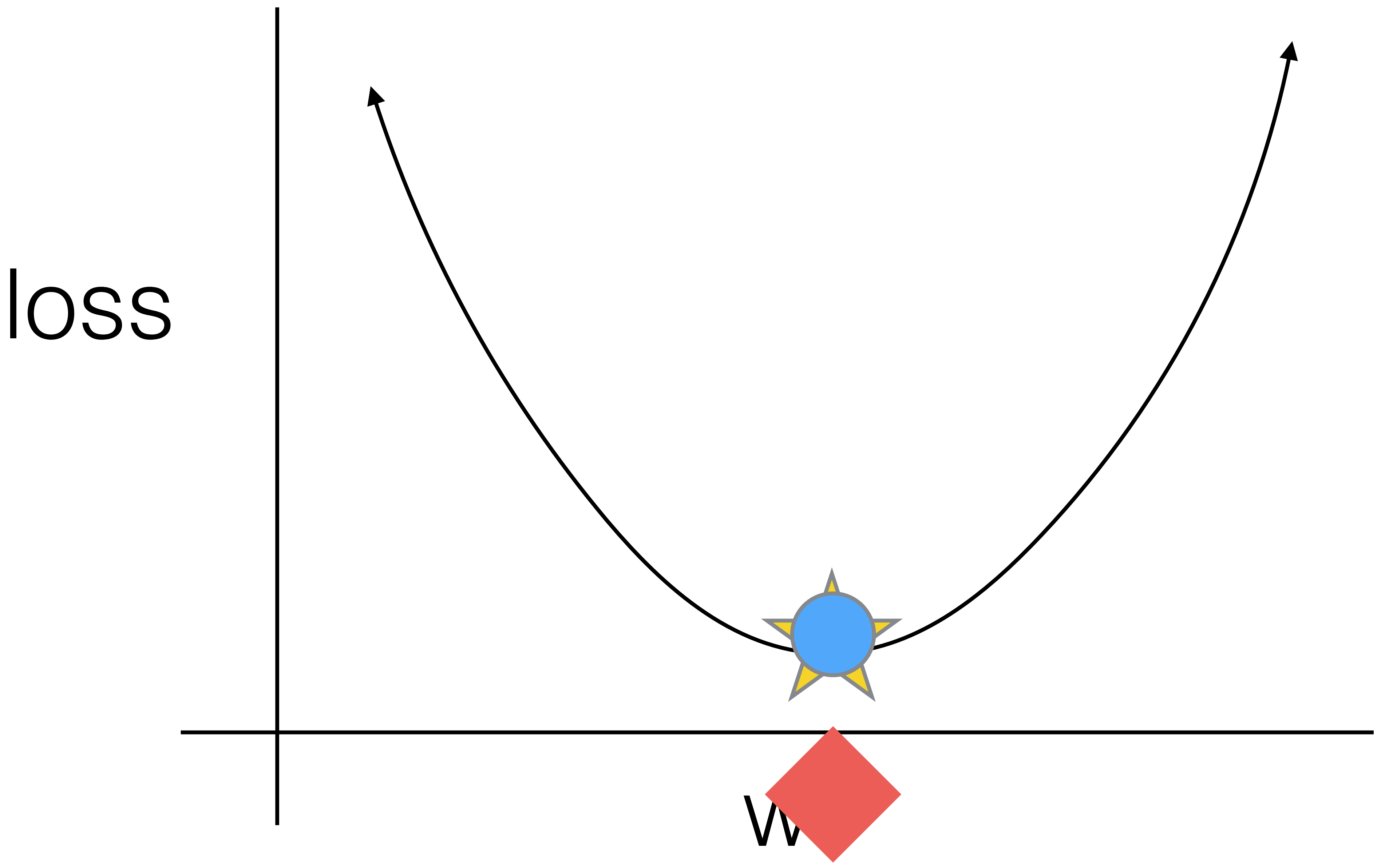



Take small step
to reduce loss,
update
parameters
accordingly.

# Logistic Regression Classifiers

## Training with Gradient Descent

$$-Y log\hat{Y} + (1 - Y)log(1 - \hat{Y})$$

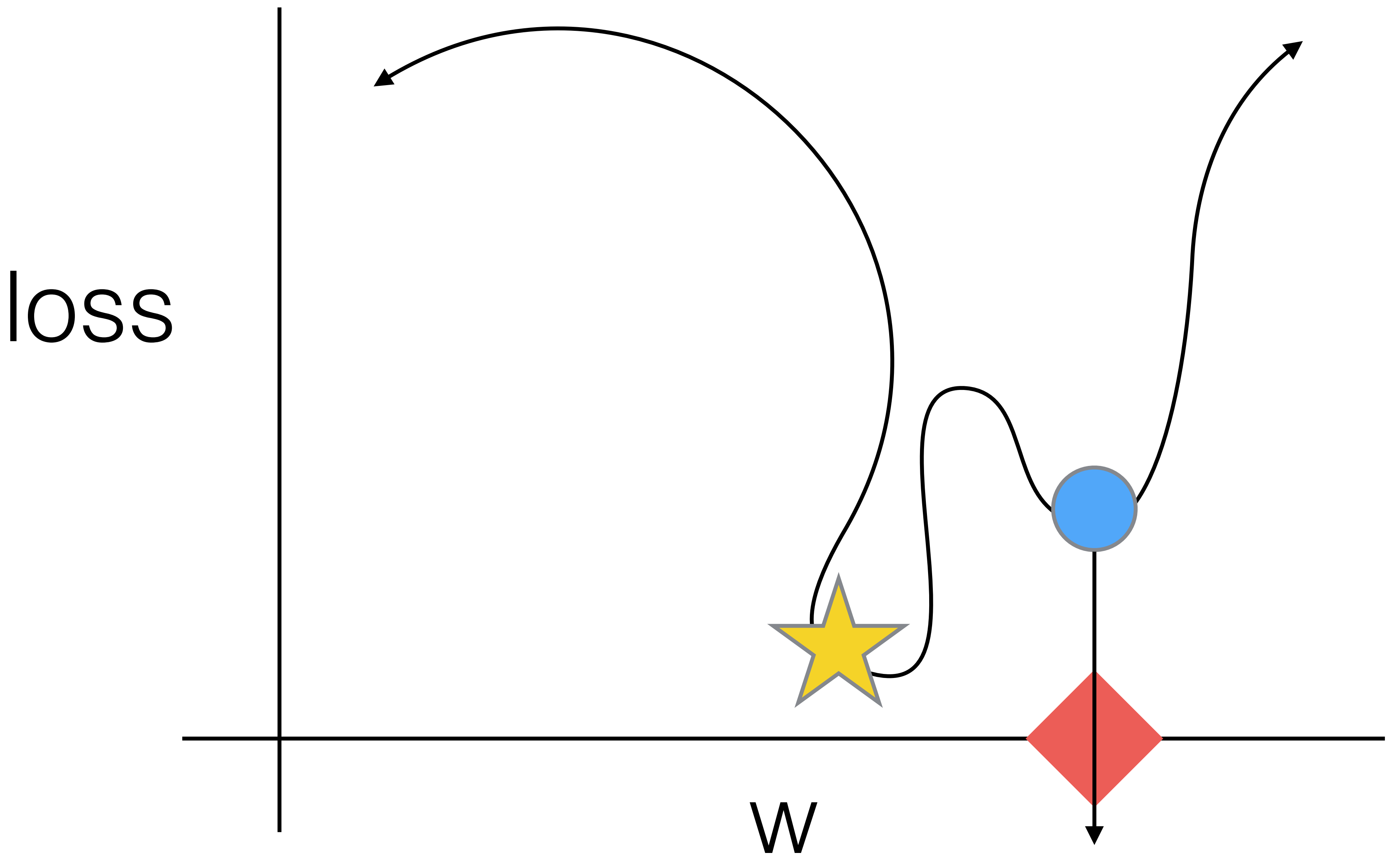



Repeat until you converge, i.e., loss cant be decreased, or you time out (like in kmeans).

# Logistic Regression Classifiers

## Training with Gradient Descent

$$-Y log\hat{Y} + (1-Y)log(1-\hat{Y})$$




Take small step to reduce loss, update parameters accordingly.

# Topics

- More Followup on Word Embeddings from SVD
- Logistic Regression and Gradient Descent
- **Multilayer Perceptrons**
- Word Embeddings from NNs

# Language Modeling Task

## Running Example

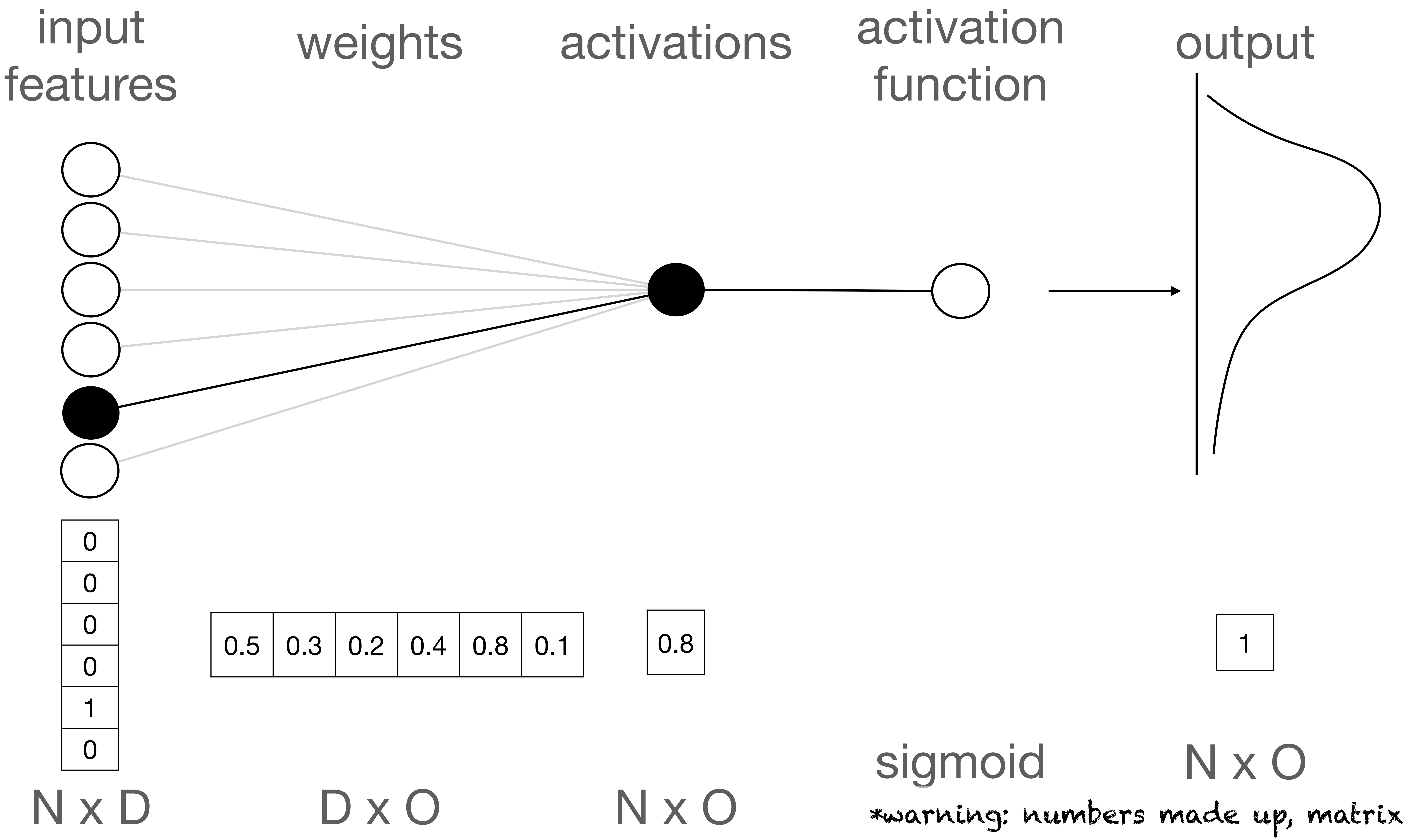
Task: Predict the next word in a sentence.

The cat sat on the ___

# Basic Perceptron

## Same as Logistic Regression

Task: Predict the next word
Input: the
Expected: cat

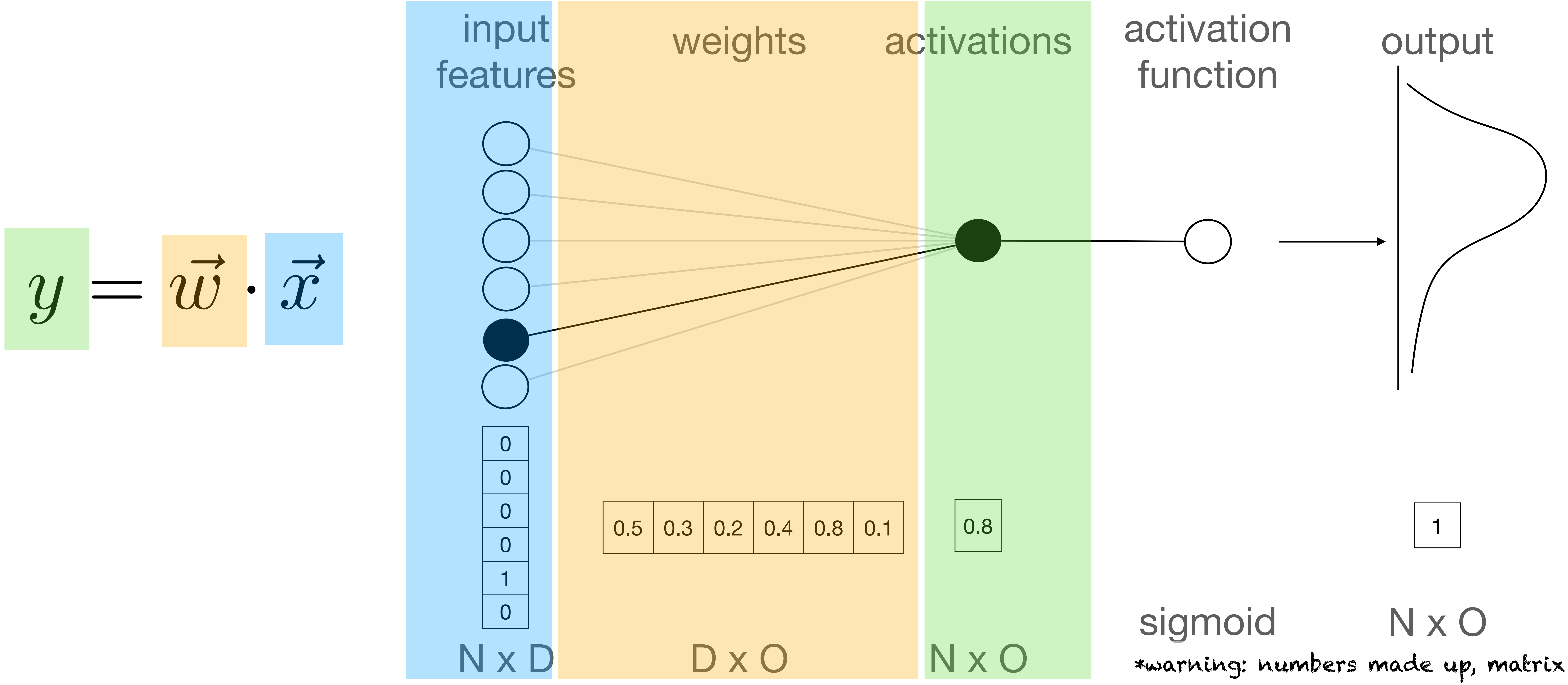



*warning: numbers made up, matrix multiplications don't necessarily work out

# Basic Perceptron

## Same as Logistic Regression

Task: Predict the next word
Input: the
Expected: cat

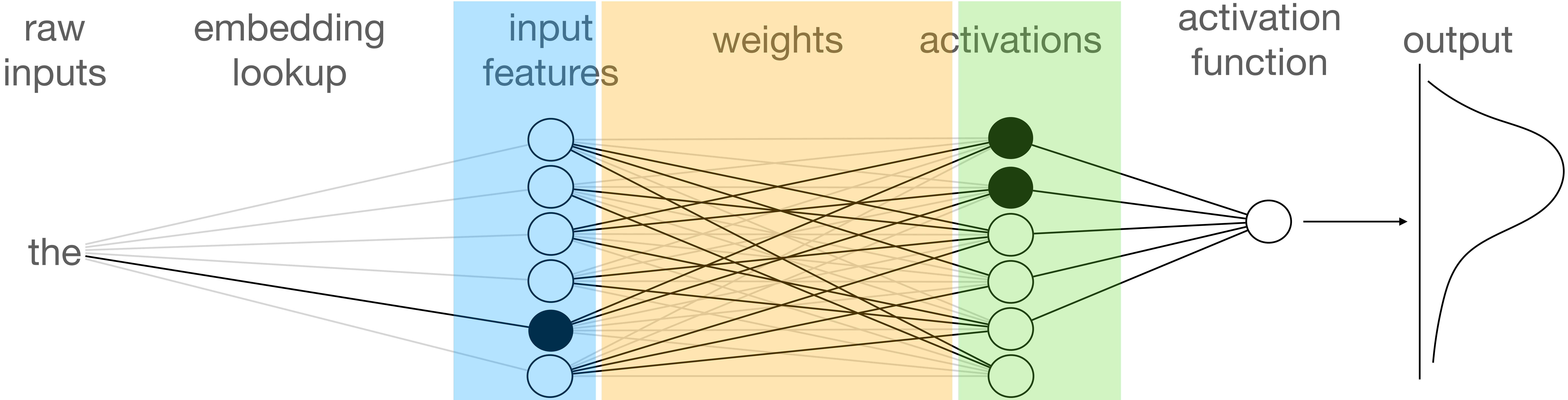



*warning: numbers made up, matrix multiplications don't necessarily work out

# Basic Perceptron

## Forward Pass

Task: Predict the next word
Input: the
Expected: cat




| | |
|---|---|
| cat | 0 |
| mat | 0 |
| on | 0 |
| sat | 0 |
| the | 1 |
| is | 0 |

N x V

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 |

V x D

| |
|---|
| 0 |
| 0 |
| 0 |
| 0 |
| 1 |
| 0 |

N x D

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 |

D x O

| |
|---|
| 1 |
| 1 |
| 0 |
| 0 |
| 0 |
| 0 |

N x O

sigmoid

| |
|---|
| 0.5 |
| 0.5 |
| 0 |
| 0 |
| 0 |
| 0 |

N x O

*warning: numbers made up, matrix multiplications don't necessarily work out

# Basic Perceptron

## Forward Pass

Task: Predict the next word
Input: the
Expected: cat


| | |
|---|---|
| cat | 0 |
| mat | 0 |
| on | 0 |
| sat | 0 |
| the | 1 |
| is | 0 |

N x V

| | | | | | | |
|---|---|---|---|---|---|---|
| cat | 1 | 0 | 0 | 0 | 0 | 0 |
| ma | 0 | 1 | 0 | 0 | 0 | 0 |
| on | 0 | 0 | 1 | 0 | 0 | 0 |
| sat | 0 | 0 | 0 | 1 | 0 | 0 |
| the | 0 | 0 | 0 | 0 | 1 | 0 |
| is | 0 | 0 | 0 | 0 | 0 | 1 |

V x D

| |
|---|
| 0 |
| 0 |
| 0 |
| 0 |
| 1 |
| 0 |

N x D

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 |

D x O

| |
|---|
| 1 |
| 1 |
| 0 |
| 0 |
| 0 |
| 0 |

N x O

softmax

| |
|---|
| 0.5 |
| 0.5 |
| 0 |
| 0 |
| 0 |
| 0 |

N x O

*warning: numbers made up, matrix multiplications don't necessarily work out

# Basic Perceptron

## Forward Pass

Task: Predict the next word
Input: the
Expected: cat


| | |
|---|---|
| cat | 0 |
| mat | 0 |
| on | 0 |
| sat | 0 |
| the | 1 |
| is | 0 |

N x V

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 |

V x D

| |
|---|
| 0 |
| 0 |
| 0 |
| 0 |
| 1 |
| 0 |

N x D

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 |

D x O

| |
|---|
| 1 |
| 1 |
| 0 |
| 0 |
| 0 |
| 0 |

N x O

softmax

| |
|---|
| 0.5 |
| 0.5 |
| 0 |
| 0 |
| 0 |
| 0 |

N x O

*warning: numbers made up, matrix multiplications don't necessarily work out

# Basic Perceptron

## Forward Pass

Task: Predict the next word
Input: the
Expected: cat


N x V (raw inputs)

| |
|---|
| 0 |
| 0 |
| 0 |
| 0 |
| 1 |
| 0 |

V x D (embedding lookup)

| | | | | | | |
|---|---|---|---|---|---|---|
| cat | 1 | 0 | 0 | 0 | 0 | 0 |
| ma | 0 | 1 | 0 | 0 | 0 | 0 |
| on | 0 | 0 | 1 | 0 | 0 | 0 |
| sat | 0 | 0 | 0 | 1 | 0 | 0 |
| the | 0 | 0 | 0 | 0 | 1 | 0 |
| is | 0 | 0 | 0 | 0 | 0 | 1 |

N x D (input features)

| |
|---|
| 0 |
| 0 |
| 0 |
| 0 |
| 1 |
| 0 |

D x O (weights)

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 |

N x O (activations)

| |
|---|
| 1 |
| 1 |
| 0 |
| 0 |
| 0 |
| 0 |

softmax

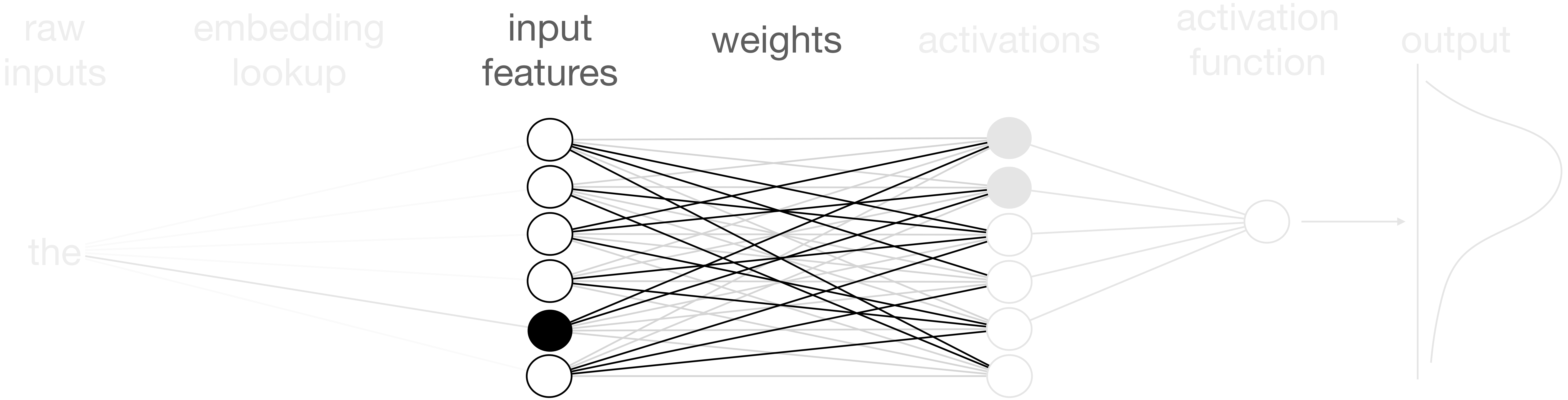
N x O (output)

| |
|---|
| 0.5 |
| 0.5 |
| 0 |
| 0 |
| 0 |
| 0 |

*warning: numbers made up, matrix multiplications don't necessarily work out

# Basic Perceptron
## Forward Pass

Task: Predict the next word
Input: the
Expected: cat




N x V

| |
|---|
| 0 |
| 0 |
| 0 |
| 0 |
| 1 |
| 0 |

V x D

| | | | | | | |
|---|---|---|---|---|---|---|
| cat | 1 | 0 | 0 | 0 | 0 | 0 |
| ma | 0 | 1 | 0 | 0 | 0 | 0 |
| on | 0 | 0 | 1 | 0 | 0 | 0 |
| sat | 0 | 0 | 0 | 1 | 0 | 0 |
| the | 0 | 0 | 0 | 0 | 1 | 0 |
| is | 0 | 0 | 0 | 0 | 0 | 1 |

N x D

| |
|---|
| 0 |
| 0 |
| 0 |
| 0 |
| 1 |
| 0 |

D x O

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 |

N x O

| |
|---|
| 1 |
| 1 |
| 0 |
| 0 |
| 0 |
| 0 |

softmax

N x O

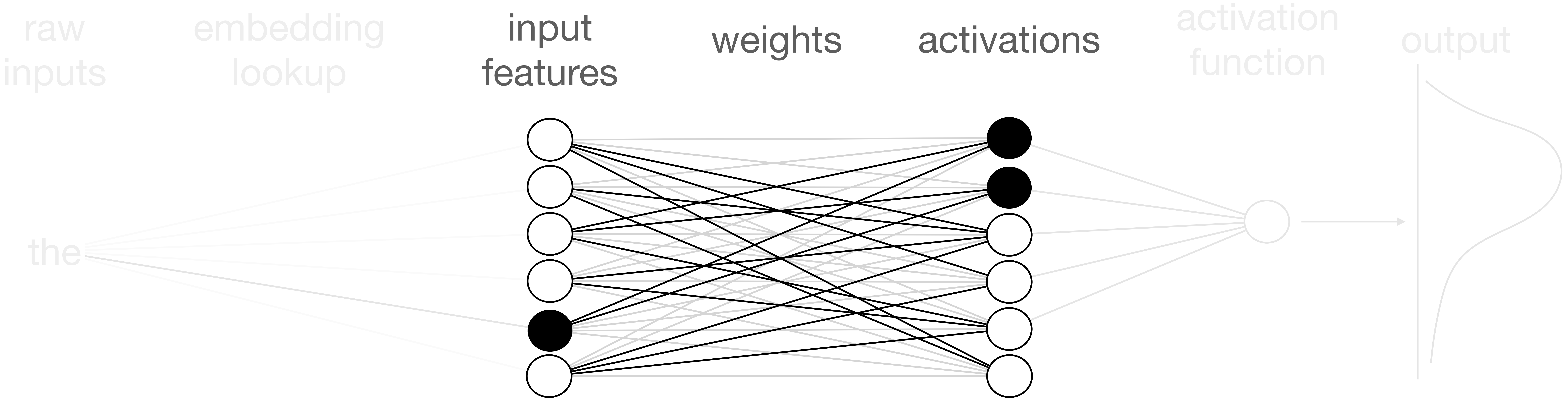
| |
|---|
| 0.5 |
| 0.5 |
| 0 |
| 0 |
| 0 |
| 0 |

*warning: numbers made up, matrix multiplications don't necessarily work out

# Basic Perceptron

## Forward Pass

Task: Predict the next word
Input: the
Expected: cat




N x V

| |
|---|
| 0 |
| 0 |
| 0 |
| 0 |
| 1 |
| 0 |

V x D

| | | | | | | |
|---|---|---|---|---|---|---|
| cat | 1 | 0 | 0 | 0 | 0 | 0 |
| ma | 0 | 1 | 0 | 0 | 0 | 0 |
| on | 0 | 0 | 1 | 0 | 0 | 0 |
| sat | 0 | 0 | 0 | 1 | 0 | 0 |
| the | 0 | 0 | 0 | 0 | 1 | 0 |
| is | 0 | 0 | 0 | 0 | 0 | 1 |

N x D

| |
|---|
| 0 |
| 0 |
| 0 |
| 0 |
| 1 |
| 0 |

D x O

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 |

N x O

| |
|---|
| 1 |
| 1 |
| 0 |
| 0 |
| 0 |
| 0 |

softmax

| |
|---|
| 0.5 |
| 0.5 |
| 0 |
| 0 |
| 0 |
| 0 |

N x O

*warning: numbers made up, matrix multiplications don't necessarily work out

# Basic Perceptron

## Forward Pass

Task: Predict the next word
Input: the
Expected: cat


| |
|---|
| 0 |
| 0 |
| 0 |
| 0 |
| 1 |
| 0 |

N x V

| | | | | | | |
|---|---|---|---|---|---|---|
| cat | 1 | 0 | 0 | 0 | 0 | 0 |
| ma | 0 | 1 | 0 | 0 | 0 | 0 |
| on | 0 | 0 | 1 | 0 | 0 | 0 |
| sat | 0 | 0 | 0 | 1 | 0 | 0 |
| the | 0 | 0 | 0 | 0 | 1 | 0 |
| is | 0 | 0 | 0 | 0 | 0 | 1 |

V x D

| |
|---|
| 0 |
| 0 |
| 0 |
| 0 |
| 1 |
| 0 |

N x D

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 |

D x O

| |
|---|
| 1 |
| 1 |
| 0 |
| 0 |
| 0 |
| 0 |

N x O

softmax

| |
|---|
| 0.5 |
| 0.5 |
| 0 |
| 0 |
| 0 |
| 0 |

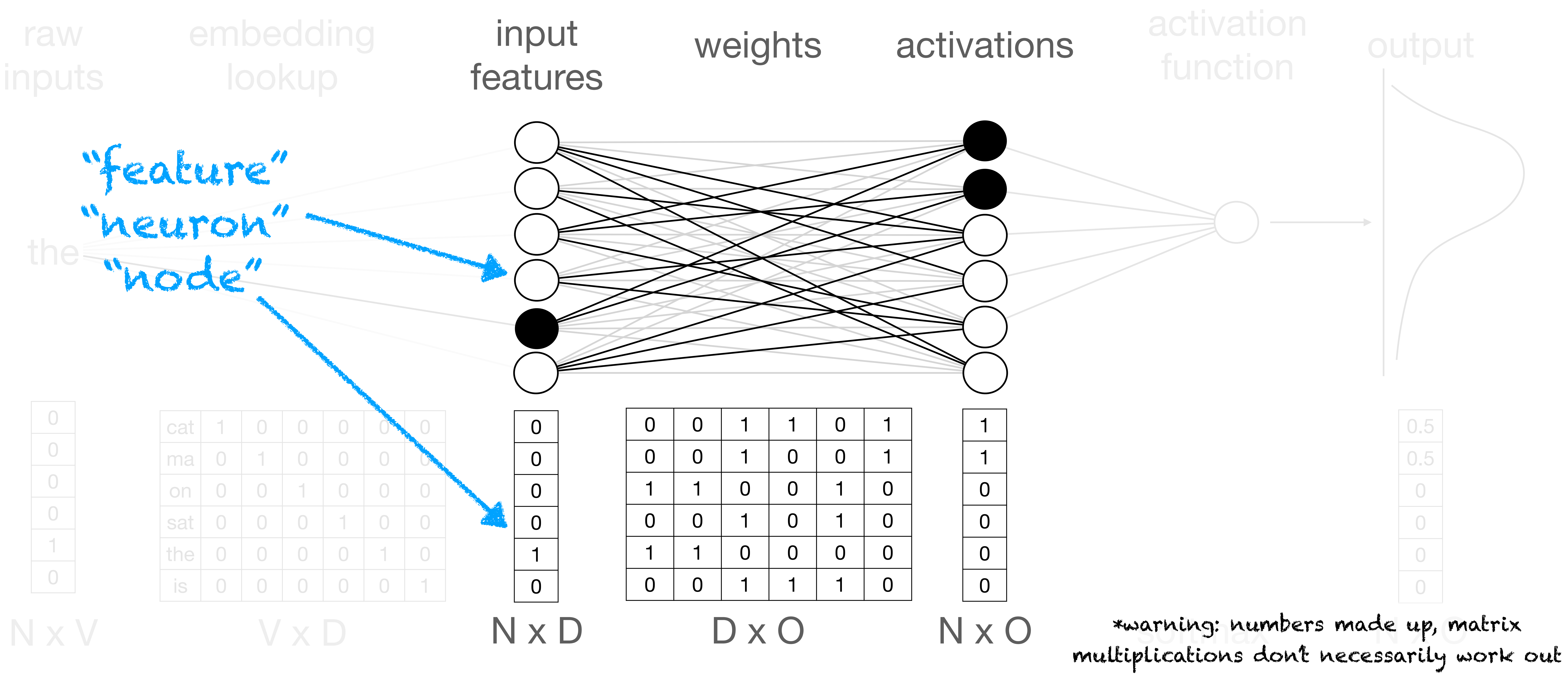
N x O

*warning: numbers made up, matrix multiplications don't necessarily work out

# Basic Perceptron

## Forward Pass

Task: Predict the next word
Input: the
Expected: cat




Input features (N x D):

| |
|---|
| 0 |
| 0 |
| 0 |
| 0 |
| 1 |
| 0 |

Weights (D x O):

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 |

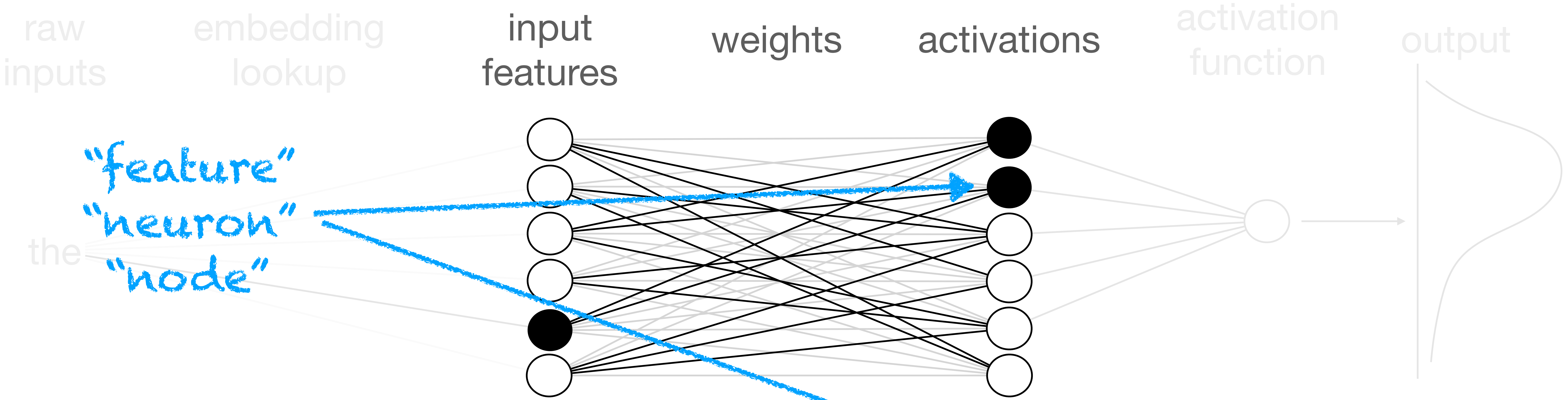
Activations (N x O):

| |
|---|
| 1 |
| 1 |
| 0 |
| 0 |
| 0 |
| 0 |

# Basic Perceptron

## Forward Pass

Task: Predict the next word
Input: the
Expected: cat




| |
|---|
| 0 |
| 0 |
| 0 |
| 0 |
| 1 |
| 0 |

N x V

| | | | | | | |
|---|---|---|---|---|---|---|
| cat | 1 | 0 | 0 | 0 | 0 | 0 |
| ma | 0 | 1 | 0 | 0 | 0 | 0 |
| on | 0 | 0 | 1 | 0 | 0 | 0 |
| sat | 0 | 0 | 0 | 1 | 0 | 0 |
| the | 0 | 0 | 0 | 0 | 1 | 0 |
| is | 0 | 0 | 0 | 0 | 0 | 1 |

V x D

| |
|---|
| 0 |
| 0 |
| 0 |
| 0 |
| 1 |
| 0 |

N x D

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 |

D x O

| |
|---|
| 1 |
| 1 |
| 0 |
| 0 |
| 0 |
| 0 |

N x O

softmax

| |
|---|
| 0.5 |
| 0.5 |
| 0 |
| 0 |
| 0 |
| 0 |

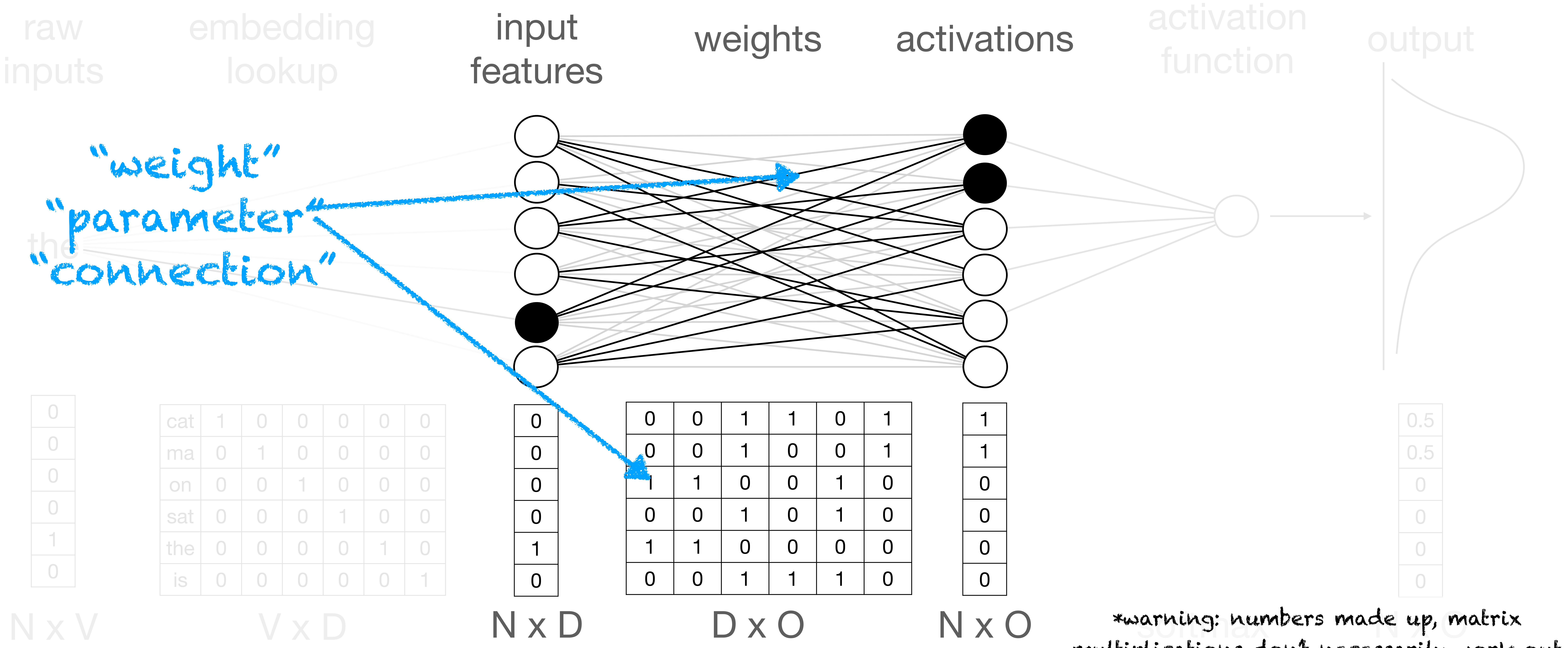
N x O

*warning: numbers made up, matrix multiplications don't necessarily work out

# Basic Perceptron

## Forward Pass

Task: Predict the next word
Input: the
Expected: cat

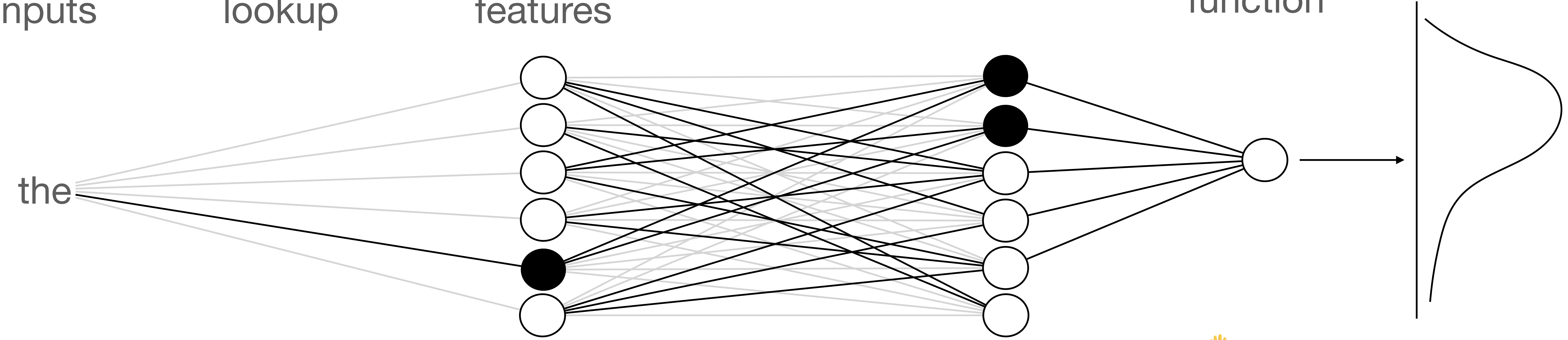



*warning: numbers made up, matrix multiplications don't necessarily work out

# Basic Perceptron
## Forward Pass

Task: Predict the next word
Input: the
Expected: cat

raw inputs | embedding lookup | input features | weights | activations | activation function | output




| | |
|---|---|
| cat | 0 |
| mat | 0 |
| on | 0 |
| sat | 0 |
| the | 1 |
| is | 0 |

N x V

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 |

V x D

| |
|---|
| 0 |
| 0 |
| 0 |
| 0 |
| 1 |
| 0 |

N x D

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 |

D x O

| |
|---|
| 1 |
| 1 |
| 0 |
| 0 |
| 0 |
| 0 |

N x O

softmax

| |
|---|
| 0.5 |
| 0.5 |
| 0 |
| 0 |
| 0 |
| 0 |

N x O

*warning: numbers made up, matrix multiplications don't necessarily work out

# Multilayer Perceptron

## Forward Pass

Task: Predict the next word
Input: the
Expected: cat


| | |
|---|---|
| cat | 0 |
| mat | 0 |
| on | 0 |
| sat | 0 |
| the | 1 |
| is | 0 |

N x V

| |
|---|
| 0 |
| 0 |
| 0 |
| 0 |
| 1 |
| 0 |

N x D

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 0 | 1 |

D x H

| |
|---|
| 1 |
| 0 |
| 0 |

N x H

| | | | | | |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 |

H x O

| |
|---|
| 1 |
| 1 |
| 0 |
| 0 |
| 0 |
| 0 |

N x O

softmax

| |
|---|
| 0.5 |
| 0.5 |
| 0 |
| 0 |
| 0 |
| 0 |

N x O

*warning: numbers made up, matrix multiplications don't necessarily work out

# Multilayer Perceptron

## Forward Pass

Task: Predict the next word
Input: the
Expected: cat


| | |
|---|---|
| cat | 0 |
| mat | 0 |
| on | 0 |
| sat | 0 |
| the | 1 |
| is | 0 |

N x V

| |
|---|
| 0 |
| 0 |
| 0 |
| 0 |
| 1 |
| 0 |

N x D

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 0 | 1 |

D x H

| |
|---|
| 1 |
| 0 |
| 0 |

N x H

H x O

Can capture abstractions over input features, e.g., "nouns" vs. "verbs"...

N x O

softmax

| |
|---|
| 0.5 |
| 0.5 |
| 0 |
| 0 |
| 0 |
| 0 |

N x O

*warning: numbers made up, matrix multiplications don't necessarily work out

# Multilayer Perceptron

## Forward Pass

Task: Predict the next word
Input: the
Expected: cat


Abstractions will capture groups that are "functionally equivalent" w.r.t. the training task.

*warning: numbers made up, matrix multiplications don't necessarily work out

# Multilayer Perceptron

## Training with Backpropagation

Parameters are randomly initialized.


| | |
|---|---|
| cat | 0 |
| mat | 0 |
| on | 0 |
| sat | 0 |
| the | 1 |
| is | 0 |

N x V

| |
|---|
| 0 |
| 0 |
| 0 |
| 0 |
| 1 |
| 0 |

N x D

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 1 | 0 |
| 0 | 0 | 1 |

D x H

| |
|---|
| 1 |
| 1 |
| 0 |

N x H

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 |

H x O

| |
|---|
| 2 |
| 0 |
| 2 |
| 1 |
| 0 |
| 1 |

N x O

softmax

| |
|---|
| 0.33 |
| 0 |
| 0.33 |
| 0.16 |
| 0 |
| 0.16 |

N x O

*warning: numbers made up, matrix multiplications don't necessarily work out

# Multilayer Perceptron
## Training with Backpropagation

I.e., predictions are random


| | |
|---|---|
| cat | 0 |
| mat | 0 |
| on | 0 |
| sat | 0 |
| the | 1 |
| is | 0 |

N x V

| |
|---|
| 0 |
| 0 |
| 0 |
| 0 |
| 1 |
| 0 |

N x D

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 1 | 0 |
| 0 | 0 | 1 |

D x H

| |
|---|
| 1 |
| 1 |
| 0 |

N x H

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 |

H x O

| |
|---|
| 2 |
| 0 |
| 2 |
| 1 |
| 0 |
| 1 |

N x O

softmax

| |
|---|
| 0.33 |
| 0 |
| 0.33 |
| 0.16 |
| 0 |
| 0.16 |

N x O

*warning: numbers made up, matrix multiplications don't necessarily work out

# Multilayer Perceptron

## Training with Backpropagation

Compare predictions to ground truth output...


| | |
|---|---|
| cat | 0 |
| mat | 0 |
| on | 0 |
| sat | 0 |
| the | 1 |
| is | 0 |

N x V

| |
|---|
| 0 |
| 0 |
| 0 |
| 0 |
| 1 |
| 0 |

N x D

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 1 | 0 |
| 0 | 0 | 1 |

D x H

| |
|---|
| 1 |
| 1 |
| 0 |

N x H

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 |

H x O

| |
|---|
| 2 |
| 0 |
| 2 |
| 1 |
| 0 |
| 1 |

N x O

| | |
|---|---|
| 0.33 | 1 |
| 0 | 0 |
| 0.33 | 0 |
| 0.16 | 0 |
| 0 | 0 |
| 0.16 | 0 |

softmax N x O

*warning: numbers made up, matrix multiplications don't necessarily work out

# Multilayer Perceptron

## Training with Backpropagation

Adjust each weight
(using gradient descent
and chain rule)


| | |
|---|---|
| cat | 0 |
| mat | 0 |
| on | 0 |
| sat | 0 |
| the | 1 |
| is | 0 |

N x V

| |
|---|
| 0 |
| 0 |
| 0 |
| 0 |
| 1 |
| 0 |

N x D

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 1 | 0 |
| 0 | 0 | 1 |

D x H

| |
|---|
| 1 |
| 1 |
| 0 |

N x H

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 |

H x O

| |
|---|
| 2 |
| 0 |
| 2 |
| 1 |
| 0 |
| 1 |

N x O

| | |
|---|---|
| 0.33 | 1 |
| 0 | 0 |
| 0.33 | 0 |
| 0.16 | 0 |
| 0 | 0 |
| 0.16 | 0 |

softmax N x O

*warning: numbers made up, matrix multiplications don't necessarily work out

# Multilayer Perceptron

## Training with Backpropagation

Adjust each weight (using gradient descent and chain rule)


N x V

| | |
|---|---|
| cat | 0 |
| mat | 0 |
| on | 0 |
| sat | 0 |
| the | 1 |
| is | 0 |

N x D

| |
|---|
| 0 |
| 0 |
| 0 |
| 0 |
| 1 |
| 0 |

D x H

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1.5 | 1.5 | 0 |
| 0 | 0 | 1 |

N x H

| |
|---|
| 1.5 |
| 1.5 |
| 0 |

H x O

| | | | | | |
|---|---|---|---|---|---|
| 2 | 0 | 1 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 |

N x O

| |
|---|
| 6 |
| 0 |
| 3 |
| 1.5 |
| 0 |
| 1.5 |

softmax

| |
|---|
| 0.5 |
| 0 |
| 0.25 |
| 0.13 |
| 0 |
| 0.13 |

N x O

| |
|---|
| 1 |
| 0 |
| 0 |
| 0 |
| 0 |
| 0 |

*warning: numbers made up, matrix multiplications don't necessarily work out

# Multilayer Perceptron

## Training with Backpropagation

Adjust each weight (using gradient descent and chain rule)


| | |
|---|---|
| cat | 0 |
| mat | 0 |
| on | 0 |
| sat | 0 |
| the | 1 |
| is | 0 |

N x V

| |
|---|
| 0 |
| 0 |
| 0 |
| 0 |
| 1 |
| 0 |

N x D

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1.5 | 1.5 | 0 |
| 0 | 0 | 1 |

D x H

| |
|---|
| 1.5 |
| 1.5 |
| 0 |

N x H

| | | | | | |
|---|---|---|---|---|---|
| 2 | 0 | 1 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 |

H x O

| |
|---|
| 6 |
| 0 |
| 3 |
| 1.5 |
| 0 |
| 1.5 |

N x O

softmax

| | |
|---|---|
| 0.5 | 1 |
| 0 | 0 |
| 0.25 | 0 |
| 0.13 | 0 |
| 0 | 0 |
| 0.13 | 0 |

N x O

*warning: numbers made up, matrix multiplications don't necessarily work out

# Topics

- More Followup on Word Embeddings from SVD
- Logistic Regression and Gradient Descent
- Multilayer Perceptrons
- **Word Embeddings from NNs**

# Word Embeddings from Neural Networks

Task: Predict the next word
Input: the
Expected: cat


| | |
|---|---|
| cat | 0 |
| mat | 0 |
| on | 0 |
| sat | 0 |
| the | 1 |
| is | 0 |

| |
|---|
| 0 |
| 0 |
| 0 |
| 0 |
| 1 |
| 0 |

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 0 | 1 |

| |
|---|
| 1 |
| 0 |
| 0 |

Can capture abstractions over input features, e.g., "nouns" vs. "verbs"...

*warning: numbers made up, matrix multiplications don't necessarily work out

# Word Embeddings from Neural Networks

Task: Predict the next word
Input: the
Expected: cat


Embedding layer can also be low-dimensional and trained with backprop!

*warning: numbers made up, matrix multiplications don't necessarily work out

# Word Embeddings from Neural Networks

Task: Predict the next word
Input: the
Expected: cat


Embedding layer can also be low-dimensional and trained with backprop!

| | |
|---|---|
| cat | 0 |
| mat | 0 |
| on | 0 |
| sat | 0 |
| the | 1 |
| is | 0 |

N x V

| | | | |
|---|---|---|---|
| 0.3 | 0.5 | 0.4 | 0.5 |
| 0.5 | 0.4 | 0.1 | 0.1 |
| 0.1 | 0.3 | 0.4 | 0.3 |
| 0.2 | 0.1 | 0.8 | 0.7 |
| 0.5 | 0.9 | 0.1 | 0.5 |
| 0.4 | 0.4 | 0.9 | 0.2 |

V x D

| |
|---|
| 0.5 |
| 0.1 |
| 0.3 |
| 0.2 |

N x D

| | | |
|---|---|---|
| 0.5 | 0.5 | 0.8 |
| 0.1 | 0.3 | 0.7 |
| 0.2 | 0.8 | 0.2 |
| 0.3 | 0.2 | 0.8 |

D x H

N x H

H x O

*warning: numbers made up, matrix multiplications don't necessarily work out

# Word Embeddings from Neural Networks

Task: Predict the next word
Input: the
Expected: cat


No different than other hidden states. But often gets special attention as the "word representation"

| | |
|---|---|
| cat | 0 |
| mat | 0 |
| on | 0 |
| sat | 0 |
| the | 1 |
| is | 0 |

N x V

| | | | |
|---|---|---|---|
| 0.3 | 0.5 | 0.4 | 0.5 |
| 0.5 | 0.4 | 0.1 | 0.1 |
| 0.1 | 0.3 | 0.4 | 0.3 |
| 0.2 | 0.1 | 0.8 | 0.7 |
| 0.5 | 0.9 | 0.1 | 0.5 |
| 0.4 | 0.4 | 0.9 | 0.2 |

V x D

| |
|---|
| 0.5 |
| 0.1 |
| 0.3 |
| 0.2 |

N x D

| | | |
|---|---|---|
| 0.5 | 0.5 | 0.8 |
| 0.1 | 0.3 | 0.7 |
| 0.2 | 0.8 | 0.2 |
| 0.3 | 0.2 | 0.8 |

D x H

*warning: numbers made up, matrix multiplications don't necessarily work out

# Word Embeddings from Neural Networks

Task: Predict the next word
Input: the
Expected: cat


No different than other hidden states. But often gets special attention as the "word representation"

| | |
|---|---|
| cat | 0 |
| mat | 0 |
| on | 0 |
| sat | 0 |
| the | 1 |
| is | 0 |

N x V

| | | | |
|---|---|---|---|
| 0.3 | 0.5 | 0.4 | 0.5 |
| 0.5 | 0.4 | 0.1 | 0.1 |
| 0.1 | 0.3 | 0.4 | 0.3 |
| 0.2 | 0.1 | 0.8 | 0.7 |
| 0.5 | 0.9 | 0.1 | 0.5 |
| 0.4 | 0.4 | 0.9 | 0.2 |

V x D

| |
|---|
| 0.5 |
| 0.1 |
| 0.3 |
| 0.2 |

N x D

| | | |
|---|---|---|
| 0.5 | 0.5 | 0.8 |
| 0.1 | 0.3 | 0.7 |
| 0.2 | 0.8 | 0.2 |
| 0.3 | 0.2 | 0.8 |

D x H

| |
|---|
| 1 |
| 0 |
| 0 |

N x H

| | | | | | |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 |

H x O

| |
|---|
| 1 |
| [illegible] |
| 0 |
| 0 |
| 0 |

N x O

| |
|---|
| 0.5 |
| 0.5 |
| 0 |
| 0 |
| 0 |
| 0 |

softmax N x O

*warning: numbers made up, matrix multiplications don't necessarily work out

# Word Embeddings from Neural Networks

**Efficient Estimation of Word Representations in Vector Space**

**Tomas Mikolov**
Google Inc., Mountain View, CA
tmikolov@google.com

**Kai Chen**
Google Inc., Mountain View, CA
kaichen@google.com

**Greg Corrado**
Google Inc., Mountain View, CA
gcorrado@google.com

**Jeffrey Dean**
Google Inc., Mountain View, CA
jeff@google.com

**Distributed Representations of Words and Phrases and their Compositionality**

**Tomas Mikolov**
Google Inc.
Mountain View
mikolov@google.com

**Ilya Sutskever**
Google Inc.
Mountain View
ilyasu@google.com

**Kai Chen**
Google Inc.
Mountain View
kai@google.com

**Greg Corrado**
Google Inc.
Mountain View
gcorrado@google.com

**Jeffrey Dean**
Google Inc.
Mountain View
jeff@google.com

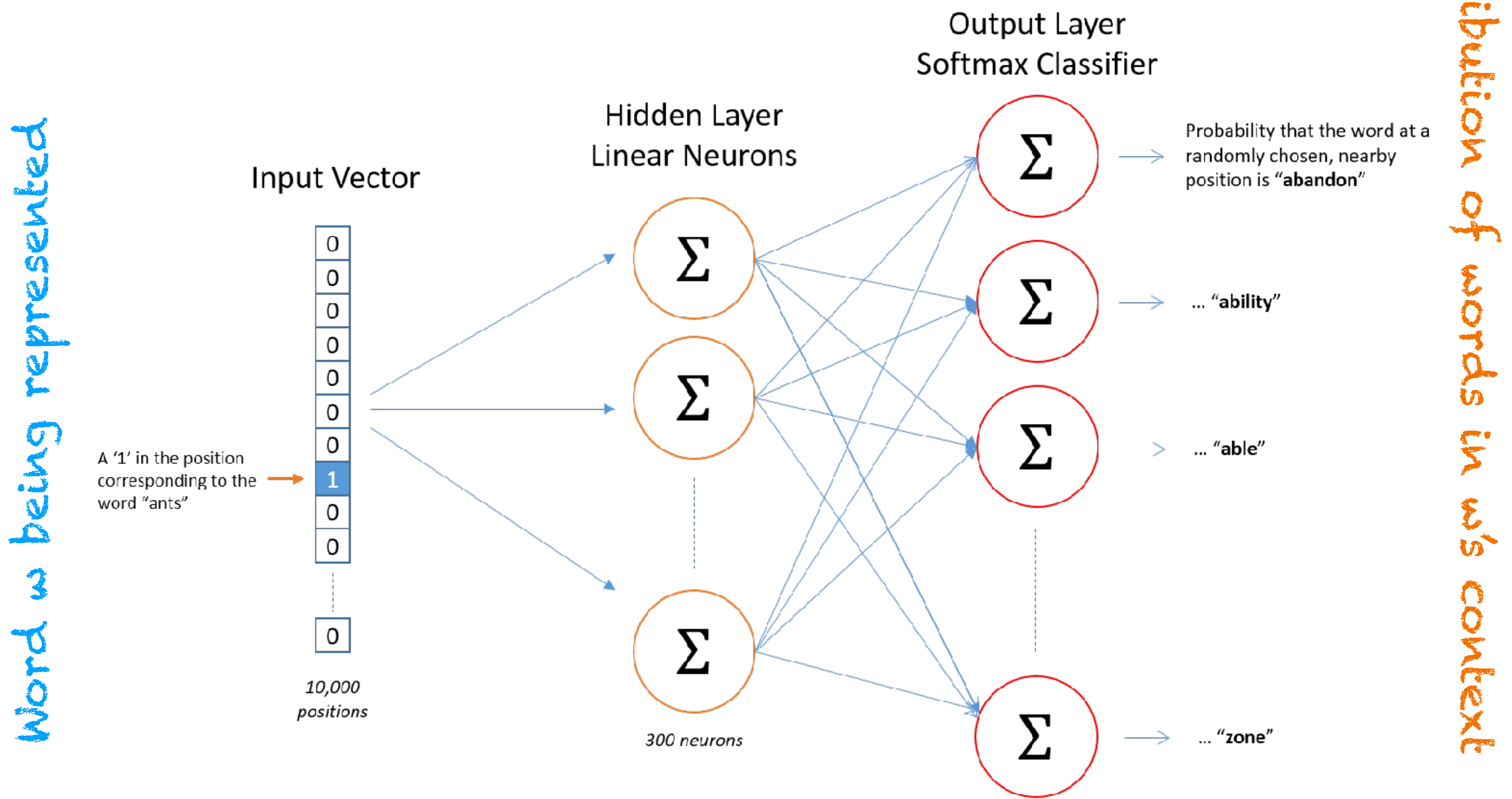
# Word Embeddings from Neural Networks

Distribution of words in w's context

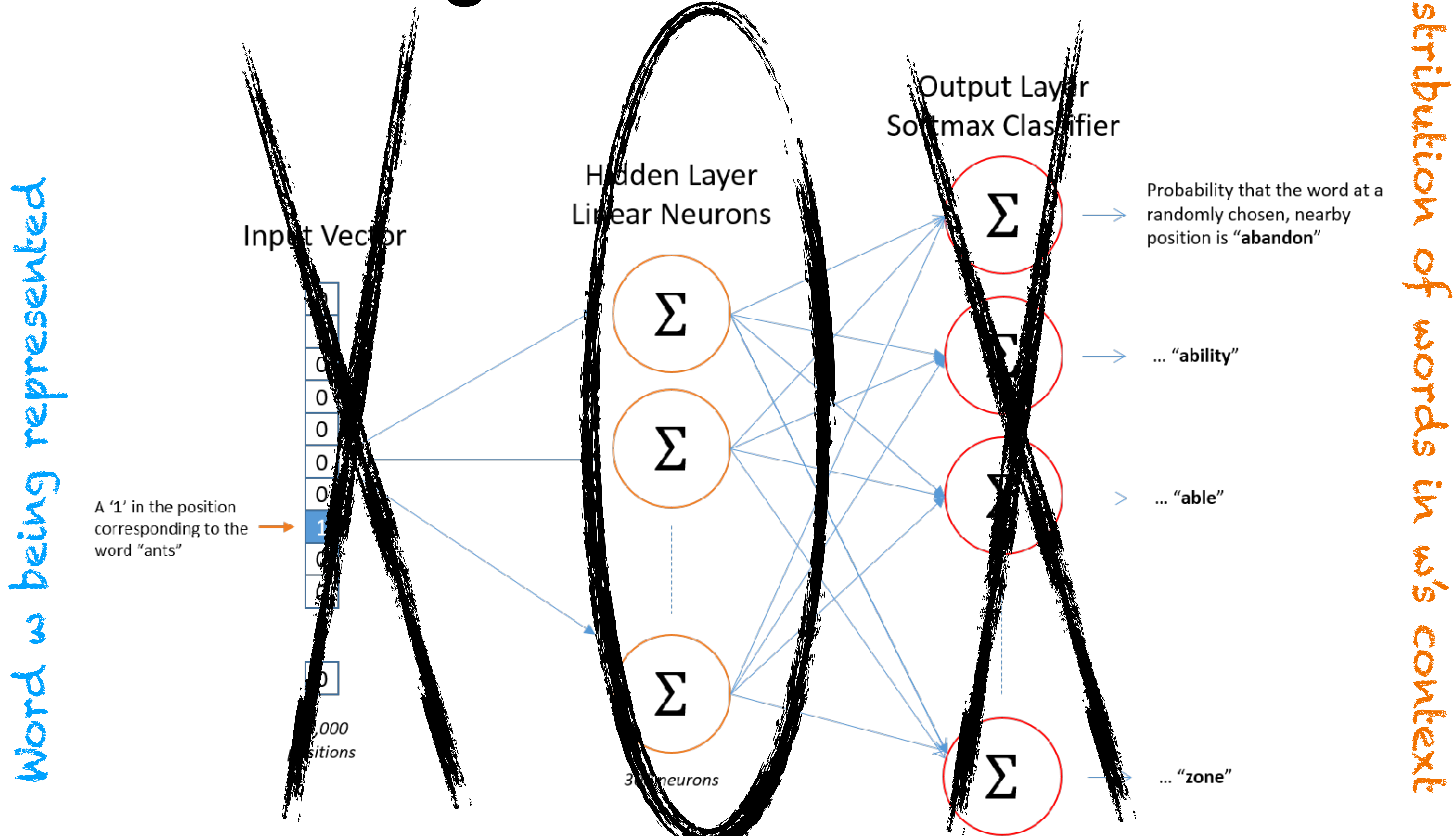
Word w being represented

| | cat | kitten | cute | adorable | gradients |
|---|---|---|---|---|---|
| cat | 0 | 0 | 1 | 1 | 0 |
| kitten | 0 | 0 | 1 | 1 | 0 |
| cute | 1 | 1 | 0 | 0 | 0 |
| adorable | 1 | 1 | 1 | 0 | 0 |
| gradients | 0 | 0 | 1 | 1 | 1 |

# Word Embeddings from Neural Networks

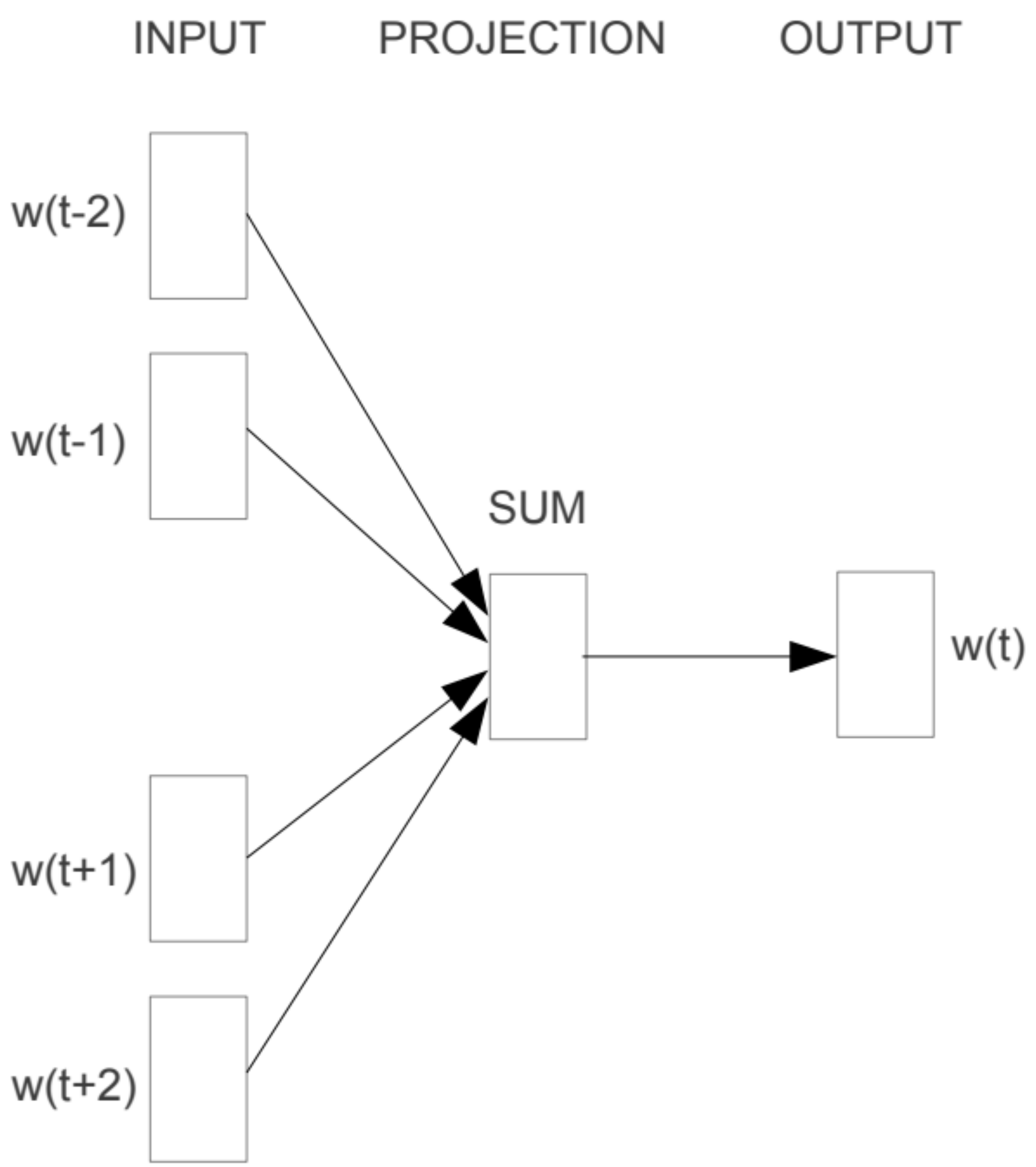


# Word Embeddings from Neural Networks



# Word Embeddings from Neural Networks

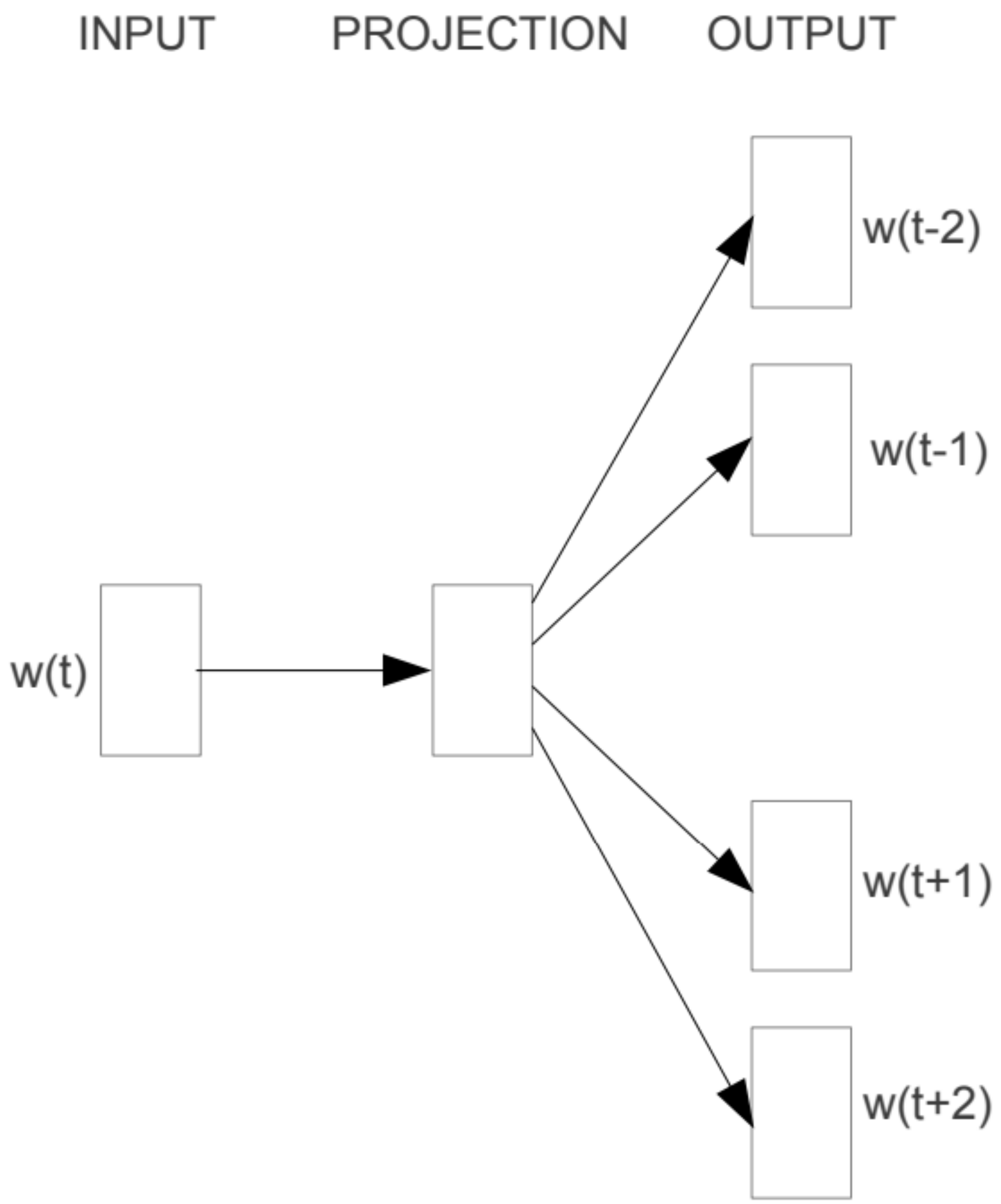
## Continuous Bag of Words (CBOW)




Given context, predict the word

# Word Embeddings from Neural Networks

## SkipGram




Given the word, predict the context

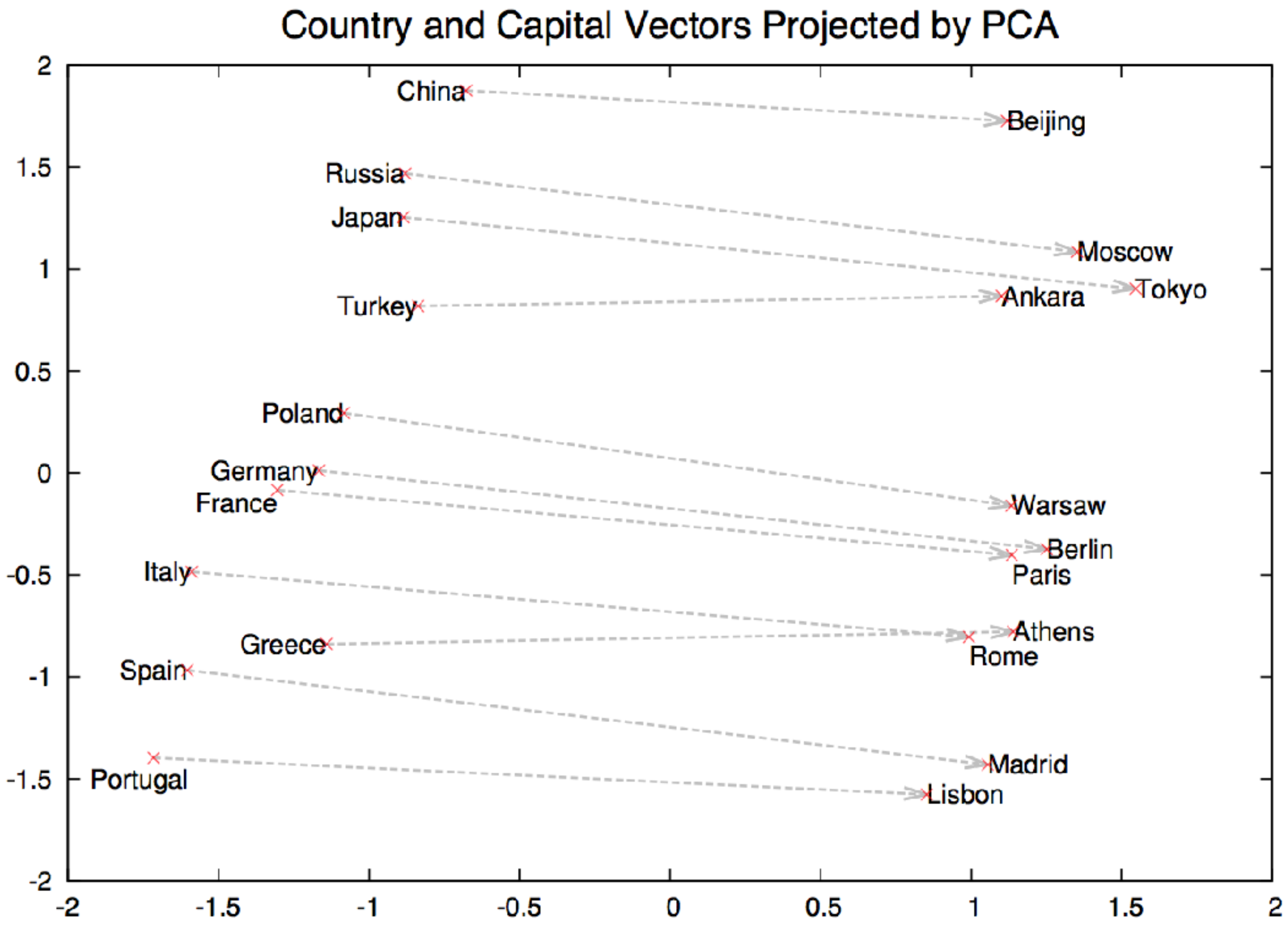
# Pr

## Eval


magenta
maroon
purple
pink
orange
yellow
red
green
blue
gray
beige
black
white
navy
apple
raspberry
strawberry
cherry
blueberry
peach
tomato
avocado
mango
banana
kiwi
pineapple
lemon
lime

# Pretrained Word Embeddings

## Evaluations of word2vec embeddings



# Pretrained Word Embeddings

## Evaluations of word2vec embeddings

Table 8: *Examples of the word pair relationships, using the best word vectors from Table 4 (Skip-gram model trained on 783M words with 300 dimensionality).*

| Relationship | Example 1 | Example 2 | Example 3 |
|---|---|---|---|
| France - Paris | Italy: Rome | Japan: Tokyo | Florida: Tallahassee |
| big - bigger | small: larger | cold: colder | quick: quicker |
| Miami - Florida | Baltimore: Maryland | Dallas: Texas | Kona: Hawaii |
| Einstein - scientist | Messi: midfielder | Mozart: violinist | Picasso: painter |
| Sarkozy - France | Berlusconi: Italy | Merkel: Germany | Koizumi: Japan |
| copper - Cu | zinc: Zn | gold: Au | uranium: plutonium |
| Berlusconi - Silvio | Sarkozy: Nicolas | Putin: Medvedev | Obama: Barack |
| Microsoft - Windows | Google: Android | IBM: Linux | Apple: iPhone |
| Microsoft - Ballmer | Google: Yahoo | IBM: McNealy | Apple: Jobs |
| Japan - sushi | Germany: bratwurst | France: tapas | USA: pizza |

# Pretrained Word Embeddings

## NNs vs. SVD

# Pretrained Word Embeddings

## NNs vs. SVD

- Same basic idea! Dimensionality reduction leads to good abstractions

# Pretrained Word Embeddings

## NNs vs. SVD

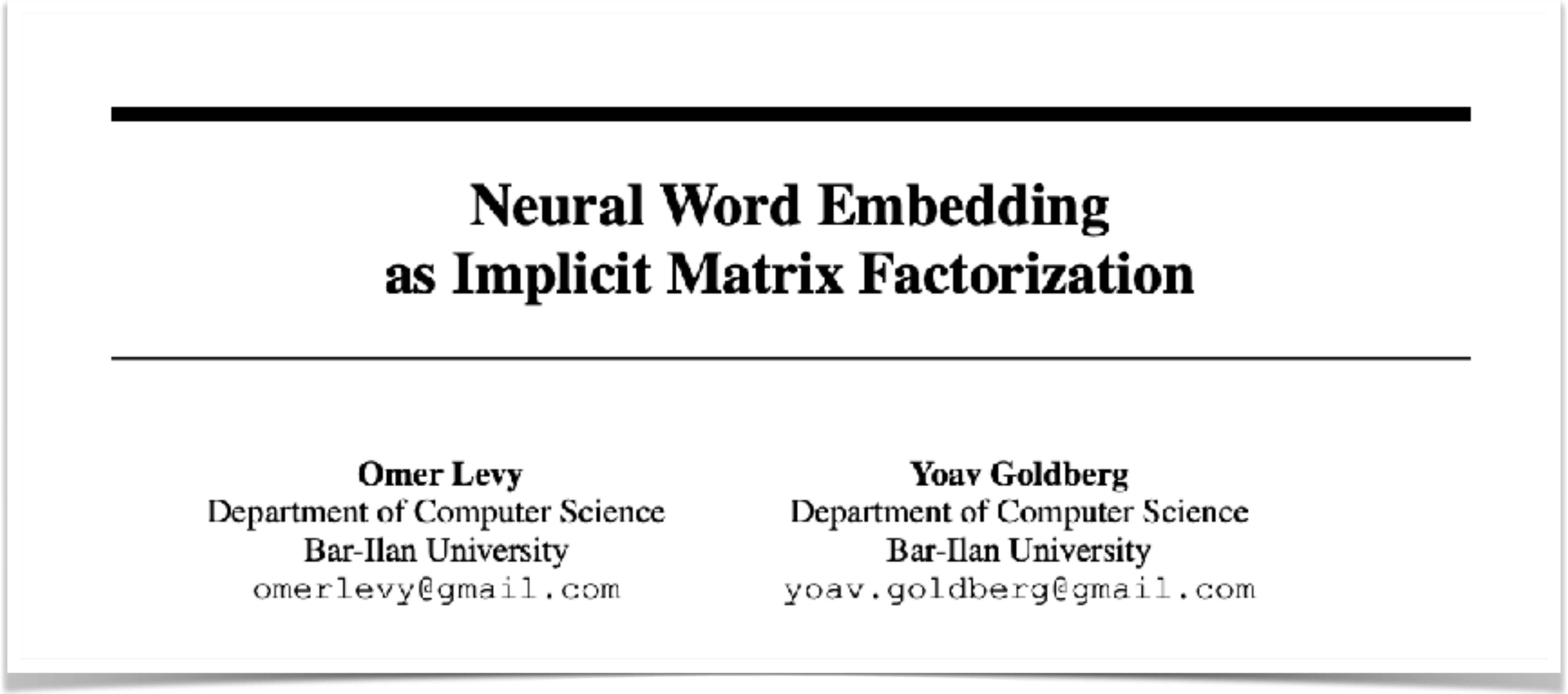
- Same basic idea! Dimensionality reduction leads to good abstractions
  - In fact, the two methods are provably equivalent in the simplest case

> **Neural Word Embedding as Implicit Matrix Factorization**
>
> **Omer Levy**
> Department of Computer Science
> Bar-Ilan University
> omerlevy@gmail.com
>
> **Yoav Goldberg**
> Department of Computer Science
> Bar-Ilan University
> yoav.goldberg@gmail.com

# Pretrained Word Embeddings

## NNs vs. SVD

- Same basic idea! Dimensionality reduction leads to good abstractions
  - In fact, the two methods are provably equivalent in the simplest case
- But embeddings from NNs can become more powerful (and harder to interpret) as:



**Neural Word Embedding as Implicit Matrix Factorization**


**Omer Levy**
Department of Computer Science
Bar-Ilan University
omerlevy@gmail.com

**Yoav Goldberg**
Department of Computer Science
Bar-Ilan University
yoav.goldberg@gmail.com

# Pretrained Word Embeddings

## NNs vs. SVD

- Same basic idea! Dimensionality reduction leads to good abstractions
  - In fact, the two methods are provably equivalent in the simplest case
- But embeddings from NNs can become more powerful (and harder to interpret) as:
  - We add more layers

**Neural Word Embedding as Implicit Matrix Factorization**

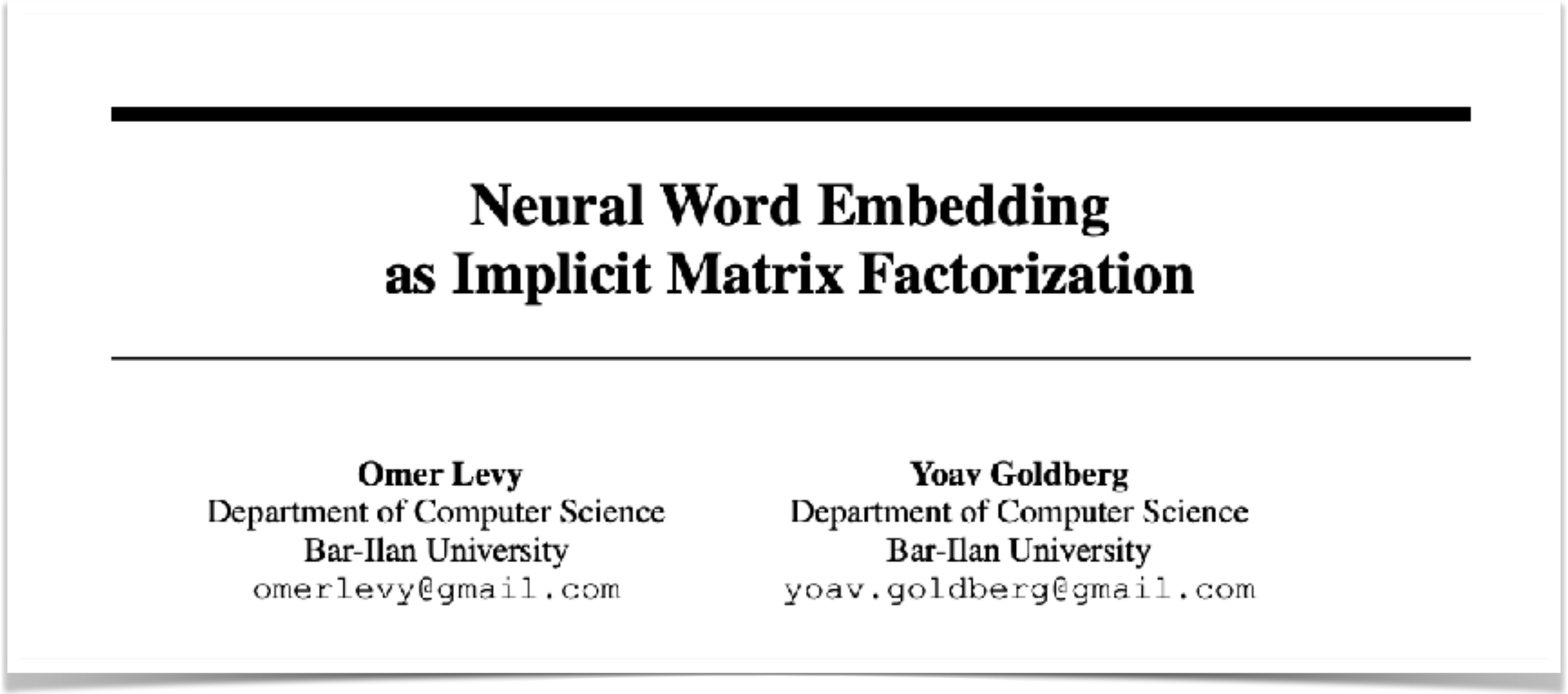
**Omer Levy**
Department of Computer Science
Bar-Ilan University
omerlevy@gmail.com

**Yoav Goldberg**
Department of Computer Science
Bar-Ilan University
yoav.goldberg@gmail.com

# Pretrained Word Embeddings

## NNs vs. SVD

- Same basic idea! Dimensionality reduction leads to good abstractions
  - In fact, the two methods are provably equivalent in the simplest case
- But embeddings from NNs can become more powerful (and harder to interpret) as:
  - We add more layers
  - We add more non-linearity



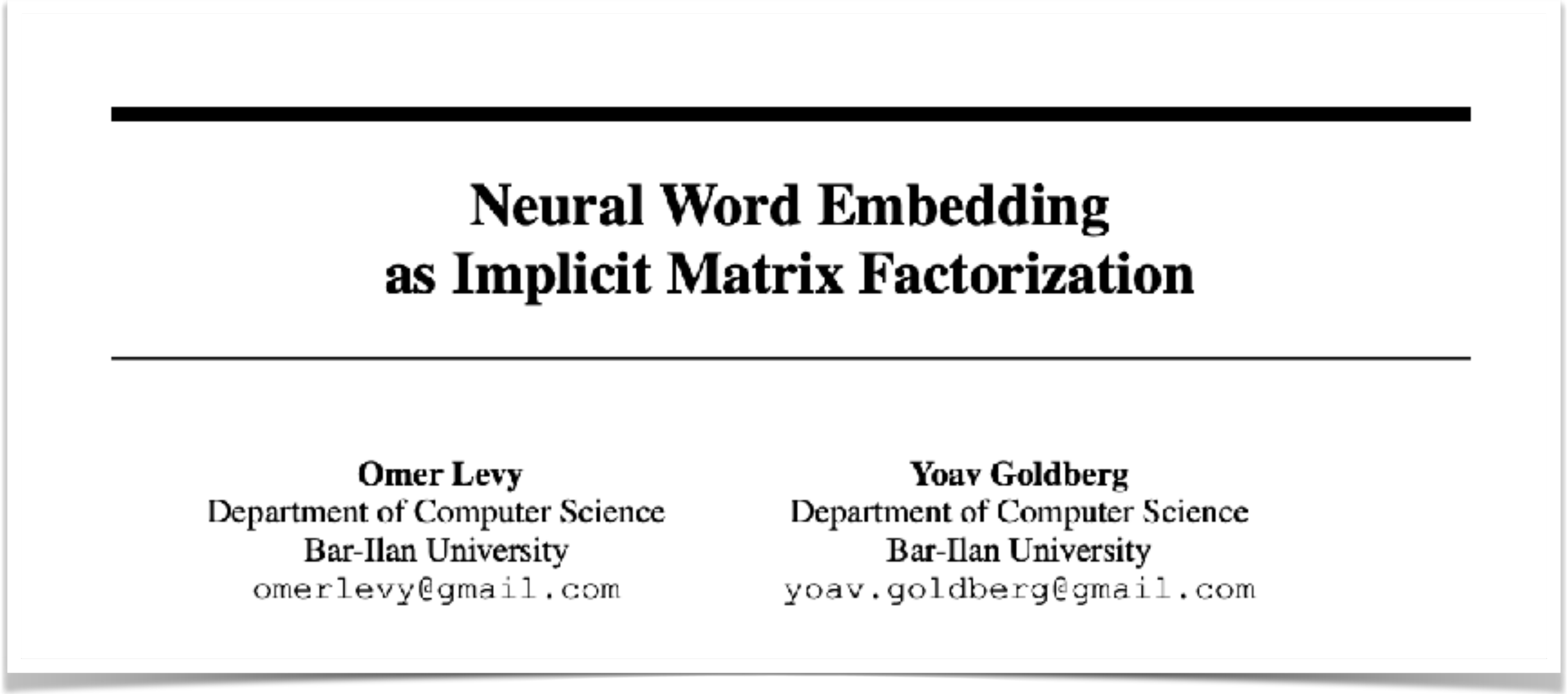
**Neural Word Embedding as Implicit Matrix Factorization**

**Omer Levy**
Department of Computer Science
Bar-Ilan University
omerlevy@gmail.com

**Yoav Goldberg**
Department of Computer Science
Bar-Ilan University
yoav.goldberg@gmail.com

# Pretrained Word Embeddings

## NNs vs. SVD



**Neural Word Embedding as Implicit Matrix Factorization**

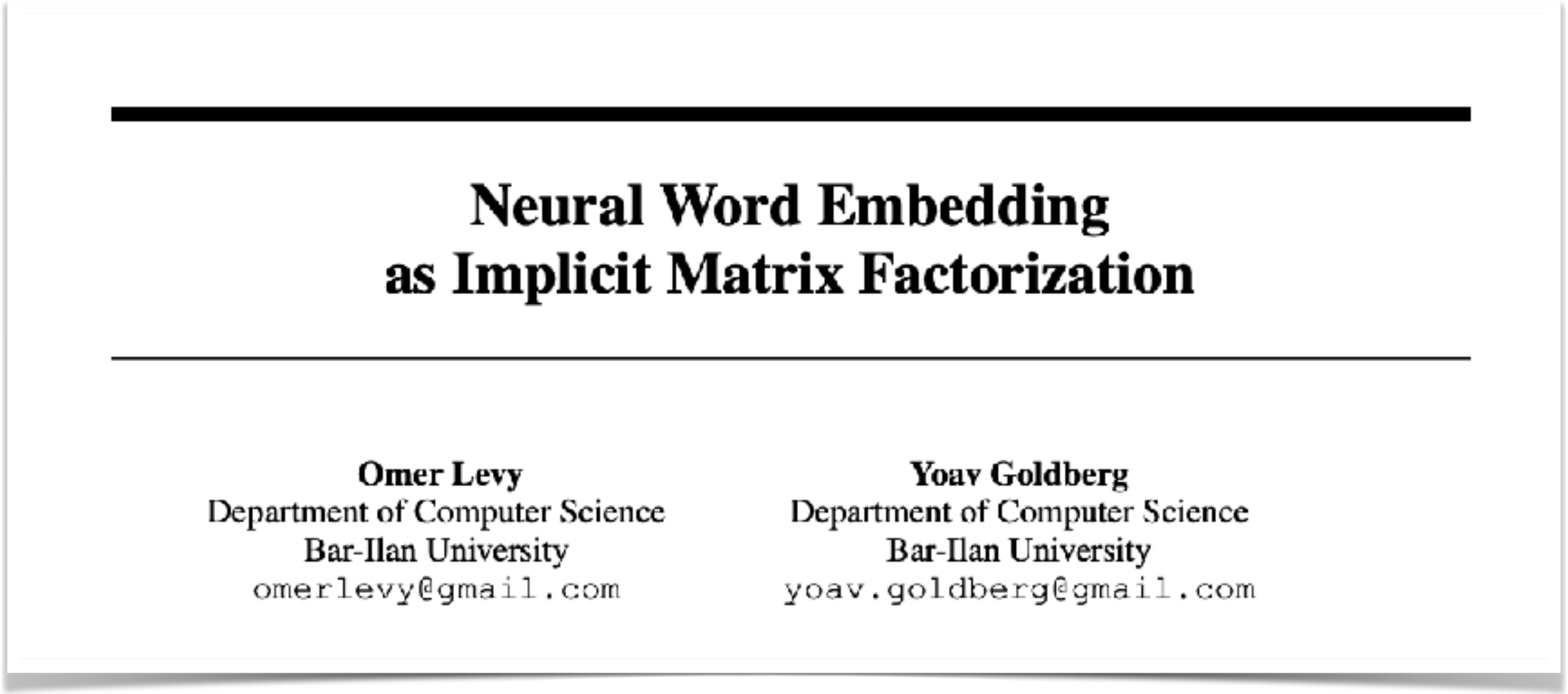
**Omer Levy**
Department of Computer Science
Bar-Ilan University
omerlevy@gmail.com

**Yoav Goldberg**
Department of Computer Science
Bar-Ilan University
yoav.goldberg@gmail.com

- Same basic idea! Dimensionality reduction leads to good abstractions
  - In fact, the two methods are provably equivalent in the simplest case
- But embeddings from NNs can become more powerful (and harder to interpret) as:
  - We add more layers
  - We add more non-linearity
  - We invent more complex training objectives

# Pretrained Word Embeddings

## NNs vs. SVD


**Neural Word Embedding as Implicit Matrix Factorization**

**Omer Levy**
Department of Computer Science
Bar-Ilan University
omerlevy@gmail.com

**Yoav Goldberg**
Department of Computer Science
Bar-Ilan University
yoav.goldberg@gmail.com

- Same basic idea! Dimensionality reduction leads to good abstractions
    - In fact, the two methods are provably equivalent in the simplest case
- But embeddings from NNs can become more powerful (and harder to interpret) as:
    - We add more layers
    - We add more non-linearity
    - We invent more complex training objectives
    - More next lecture(s)!