IN5550: Neural Methods in Natural Language Processing
Sub-lecture 2.2
Linear classifiers

Andrey Kutuzov

University of Oslo

31 January 2023

Contents

Linear classifiers

Simple linear function

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) = \boldsymbol{x} \cdot \boldsymbol{W} + \boldsymbol{b} \quad (1)$$

$$\theta = \boldsymbol{W}, \boldsymbol{b} \quad (2)$$

- Function input:
  - feature vector $\boldsymbol{x} \in \mathbb{R}^{d_{in}}$;
  - each training instance is represented with d_{in} features;
  - for example, some properties of the documents.

Simple linear function

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) = \boldsymbol{x} \cdot \boldsymbol{W} + \boldsymbol{b} \tag{1}$$

$$\theta = \boldsymbol{W}, \boldsymbol{b} \tag{2}$$

- Function input:
  - feature vector $\boldsymbol{x} \in \mathbb{R}^{d_{in}}$;
  - each training instance is represented with d_{in} features;
  - for example, some properties of the documents.
- Function parameters θ:
  - matrix $\boldsymbol{W} \in \mathbb{R}^{d_{in} \times d_{out}}$
    - d_{out} is the dimensionality of the desired prediction (number of classes)
  - bias vector $\boldsymbol{b} \in \mathbb{R}^{d_{out}}$
    - bias ‘shifts’ the function output to some direction.

Training of a linear classifier

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) = \boldsymbol{x} \cdot \boldsymbol{W} + \boldsymbol{b}$$

$$\theta = \boldsymbol{W}, \boldsymbol{b}$$

- Training is finding the optimal θ.

Training of a linear classifier

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) = \boldsymbol{x} \cdot \boldsymbol{W} + \boldsymbol{b}$$

$$\theta = \boldsymbol{W}, \boldsymbol{b}$$

- Training is finding the optimal θ.
- 'Optimal' means '*producing predictions* $\hat{\boldsymbol{y}}$ *closest to the gold labels* $\boldsymbol{y}$ *on our n training instances*'.

Training of a linear classifier

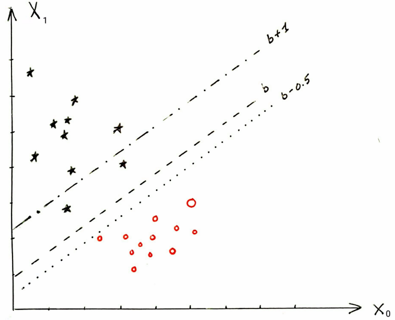
$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) = \boldsymbol{x} \cdot \boldsymbol{W} + \boldsymbol{b}$$

$$\theta = \boldsymbol{W}, \boldsymbol{b}$$

- Training is finding the optimal θ.
- 'Optimal' means '*producing predictions* $\hat{\boldsymbol{y}}$ *closest to the gold labels* $\boldsymbol{y}$ *on our n training instances*'.
- Ideally, $\hat{\boldsymbol{y}} = \boldsymbol{y}$

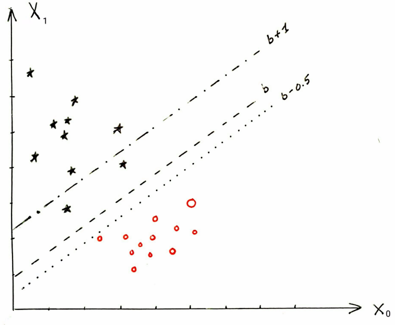
Linear classifiers

Here, training instances are represented with 2 features each ($\boldsymbol{x} = [x_0, x_1]$) and labeled with 2 class labels ($\boldsymbol{y} = \{black, red\}$):



Linear classifiers

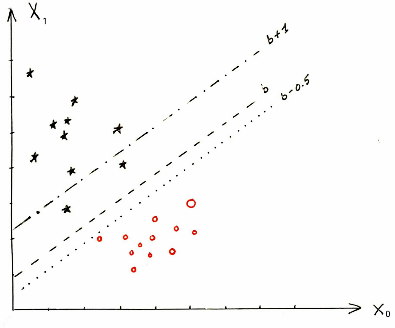
Here, training instances are represented with 2 features each ($\boldsymbol{x} = [x_0, x_1]$) and labeled with 2 class labels ($\boldsymbol{y} = \{black, red\}$):



- Parameters of $f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) = \boldsymbol{x} \cdot \boldsymbol{W} + \boldsymbol{b}$ define the line (or hyperplane) separating the instances.

Linear classifiers

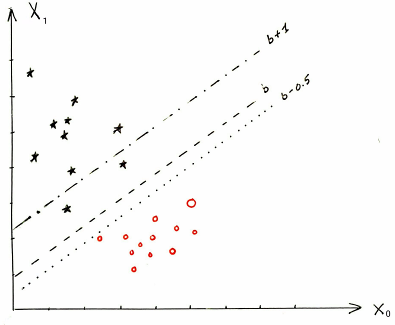
Here, training instances are represented with 2 features each ($\boldsymbol{x} = [x_0, x_1]$) and labeled with 2 class labels ($\boldsymbol{y} = \{black, red\}$):



- Parameters of $f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) = \boldsymbol{x} \cdot \boldsymbol{W} + \boldsymbol{b}$ define the line (or hyperplane) separating the instances.
- This decision boundary is actually our learned classifier.

Linear classifiers

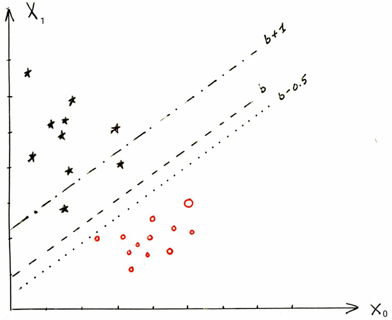
Here, training instances are represented with 2 features each ($\boldsymbol{x} = [x_0, x_1]$) and labeled with 2 class labels ($\boldsymbol{y} = \{black, red\}$):



- Parameters of $f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) = \boldsymbol{x} \cdot \boldsymbol{W} + \boldsymbol{b}$ define the line (or hyperplane) separating the instances.
- This decision boundary is actually our learned classifier.
- NB: the dataset on the plot is linearly separable.

Linear classifiers

Here, training instances are represented with 2 features each ($\boldsymbol{x} = [x_0, x_1]$) and labeled with 2 class labels ($\boldsymbol{y} = \{black, red\}$):



- Parameters of $f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) = \boldsymbol{x} \cdot \boldsymbol{W} + \boldsymbol{b}$ define the line (or hyperplane) separating the instances.
- This decision boundary is actually our learned classifier.
- NB: the dataset on the plot is linearly separable.
- **Question: lines with 3 values of b are shown. Which is the best?**

Linear classifiers

What would be a general representation of text?

Linear classifiers

What would be a general representation of text?
Suppose you don't want to choose a ton of features by hand,

What would be a general representation of text?
Suppose you don't want to choose a ton of features by hand,
and you don't care about word order

Linear classifiers

What would be a general representation of text?
Suppose you don't want to choose a ton of features by hand,
and you don't care about word order

Bag of words

What would be a general representation of text?
Suppose you don't want to choose a ton of features by hand,
and you don't care about word order

Bag of words

- Each word from a pre-defined vocabulary D can be a separate feature.

What would be a general representation of text?
Suppose you don't want to choose a ton of features by hand,
and you don't care about word order

Bag of words

- Each word from a pre-defined vocabulary D can be a separate feature.
- How many times the word a appears in the document i?
  - or a binary flag $\{1, 0\}$ of whether a appeared in i at all or not.

What would be a general representation of text?
Suppose you don't want to choose a ton of features by hand,
and you don't care about word order

Bag of words

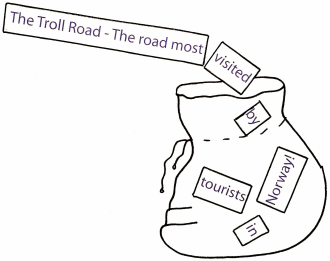
- Each word from a pre-defined vocabulary D can be a separate feature.
- How many times the word a appears in the document i?
  - or a binary flag $\{1, 0\}$ of whether a appeared in i at all or not.
- This schema is called 'bag of words' (BoW).
  - for example, if we have 1000 words in the vocabulary:
  - $\boldsymbol{x}_i \in \mathbb{R}^{1000}$

What would be a general representation of text?
Suppose you don't want to choose a ton of features by hand, and you don't care about word order

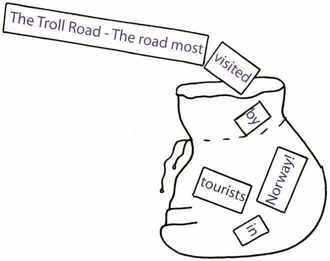
Bag of words

- Each word from a pre-defined vocabulary D can be a separate feature.
- How many times the word a appears in the document i?
  - or a binary flag $\{1, 0\}$ of whether a appeared in i at all or not.
- This schema is called 'bag of words' (BoW).
  - for example, if we have 1000 words in the vocabulary:
  - $\boldsymbol{x}_i \in \mathbb{R}^{1000}$
  - $\boldsymbol{x}_i = [20, 16, 0, 10, 0, \ldots, 3]$

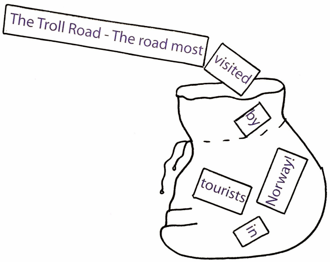
Linear classifiers



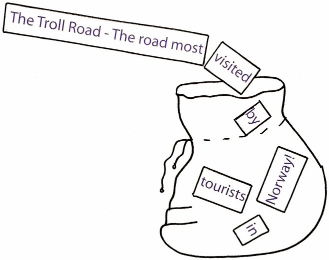
- Bag-of-Words feature vector of x can be interpreted as a sum of one-hot vectors ($\boldsymbol{o}$) for each token in it:



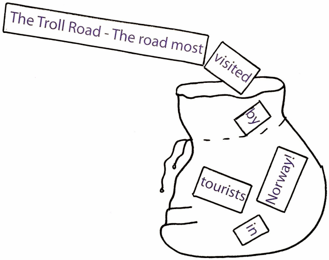
- Bag-of-Words feature vector of x can be interpreted as a sum of one-hot vectors ($\boldsymbol{o}$) for each token in it:
  - D extracted from the text above contains 10 words (lowercased): $\{$*'-', 'by', 'in', 'most', 'norway', 'road', 'the', 'tourists', 'troll', 'visited'*$\}$.



- Bag-of-Words feature vector of x can be interpreted as a sum of one-hot vectors ($\boldsymbol{o}$) for each token in it:
  - D extracted from the text above contains 10 words (lowercased): $\{$*'-', 'by', 'in', 'most', 'norway', 'road', 'the', 'tourists', 'troll', 'visited'*$\}$.
  - $\boldsymbol{o}^0 = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0]$



- Bag-of-Words feature vector of x can be interpreted as a sum of one-hot vectors ($\boldsymbol{o}$) for each token in it:
  - D extracted from the text above contains 10 words (lowercased): $\{$'-', *'by', 'in', 'most', 'norway', 'road', 'the', 'tourists', 'troll', 'visited'*$\}$.
  - $\boldsymbol{o}^0 = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0]$
  - $\boldsymbol{o}^1 = [0, 0, 0, 0, 0, 0, 0, 0, 1, 0]$
  - etc...



- Bag-of-Words feature vector of x can be interpreted as a sum of one-hot vectors ($\boldsymbol{o}$) for each token in it:
  - D extracted from the text above contains 10 words (lowercased): $\{$'-', '*by*', '*in*', '*most*', '*norway*', '*road*', '*the*', '*tourists*', '*troll*', '*visited*'$\}$.
  - $\boldsymbol{o}^0 = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0]$
  - $\boldsymbol{o}^1 = [0, 0, 0, 0, 0, 0, 0, 0, 1, 0]$
  - etc...
  - $\boldsymbol{i} = [1, 1, 1, 1, 1, 2, 2, 1, 1, 1]$ ('*the*' and '*road*' mentioned 2 times)

Linear classifiers

Can we interpret the different parts of a learned model as representations of the data?

Linear classifiers

Can we interpret the different parts of a learned model as representations of the data?

- Each of n instances (documents) is represented by a vector of features ($\boldsymbol{x} \in \mathbb{R}^{d_{in}}$).

Can we interpret the different parts of a learned model as representations of the data?

- Each of n instances (documents) is represented by a vector of features ($\boldsymbol{x} \in \mathbb{R}^{d_{in}}$).
- Inversely, each feature can be represented by a vector of instances (documents) it appears in ($\boldsymbol{feature} \in \mathbb{R}^{n}$).

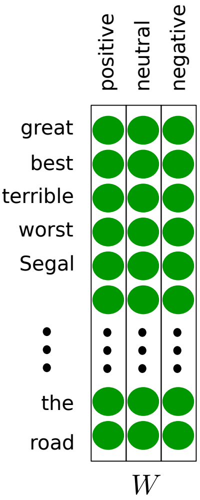
Can we interpret the different parts of a learned model as representations of the data?

- Each of n instances (documents) is represented by a vector of features ($\boldsymbol{x} \in \mathbb{R}^{d_{in}}$).
- Inversely, each feature can be represented by a vector of instances (documents) it appears in ($\boldsymbol{feature} \in \mathbb{R}^{n}$).
- Together these learned representations form a $\boldsymbol{W}$ matrix, part of θ.
  - Thus, it contains data both about the instances and their features (more about this later).

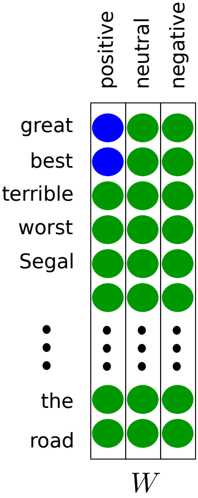
Can we interpret the different parts of a learned model as representations of the data?

- Each of n instances (documents) is represented by a vector of features ($\boldsymbol{x} \in \mathbb{R}^{d_{in}}$).
- Inversely, each feature can be represented by a vector of instances (documents) it appears in ($\boldsymbol{feature} \in \mathbb{R}^{n}$).
- Together these learned representations form a $\boldsymbol{W}$ matrix, part of θ.
  - Thus, it contains data both about the instances and their features (more about this later).
- Feature engineering is deciding what features of the instances we will use during the training.

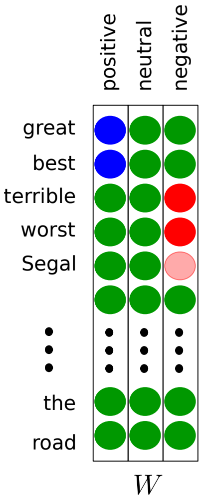
Linear classifiers



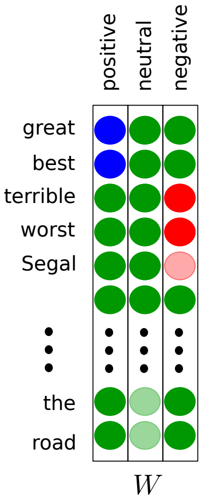
Linear classifiers



Linear classifiers



Linear classifiers



Linear classifiers

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) = \boldsymbol{x} \cdot \boldsymbol{W} + \boldsymbol{b}$$

Output of binary classification

Binary decision ($d_{out} = 1$):

Linear classifiers

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) = \boldsymbol{x} \cdot \boldsymbol{W} + \boldsymbol{b}$$

Output of binary classification

Binary decision ($d_{out} = 1$):

- '*Is this message spam or not?*'

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) = \boldsymbol{x} \cdot \boldsymbol{W} + \boldsymbol{b}$$

Output of binary classification

Binary decision ($d_{out} = 1$):

- '*Is this message spam or not?*'
- $\boldsymbol{W}$ is a vector, $\boldsymbol{b}$ is a scalar.

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) = \boldsymbol{x} \cdot \boldsymbol{W} + \boldsymbol{b}$$

Output of binary classification

Binary decision ($d_{out} = 1$):

- '*Is this message spam or not?*'
- $\boldsymbol{W}$ is a vector, $\boldsymbol{b}$ is a scalar.
- The prediction $\hat{\boldsymbol{y}}$ is also a scalar: either 1 ('yes') or -1 ('no').

Linear classifiers

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) = \boldsymbol{x} \cdot \boldsymbol{W} + \boldsymbol{b}$$

Output of binary classification

Binary decision ($d_{out} = 1$):

- '*Is this message spam or not?*'
- $\boldsymbol{W}$ is a vector, $\boldsymbol{b}$ is a scalar.
- The prediction $\hat{\boldsymbol{y}}$ is also a scalar: either 1 ('yes') or -1 ('no').
- NB: the model can output any number, but we convert all negatives to -1 and all positives to 1 (*sign* function).

$$\theta = (\boldsymbol{W} \in \mathbb{R}^{d_{in}}, \boldsymbol{b} \in \mathbb{R}^1)$$

Linear classifiers

$$f(x; W, b) = x \cdot W + b$$

$$\underset{x}{\begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}} \cdot \underset{W}{\begin{bmatrix} 0 & 0 & 1 & 1 & 1 \end{bmatrix}} + \underset{b}{0.5} = \underset{\hat{y}}{\text{sign}(1.5) = 1}$$

Linear classifiers

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) = \boldsymbol{x} \cdot \boldsymbol{W} + \boldsymbol{b}$$

Output of multi-class classification

Linear classifiers

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) = \boldsymbol{x} \cdot \boldsymbol{W} + \boldsymbol{b}$$

Output of multi-class classification

Multi-class decision ($d_{out} = k$)

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) = \boldsymbol{x} \cdot \boldsymbol{W} + \boldsymbol{b}$$

Output of multi-class classification

Multi-class decision ($d_{out} = k$)

- '*Which languages appear in this tweet?*'

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) = \boldsymbol{x} \cdot \boldsymbol{W} + \boldsymbol{b}$$

Output of multi-class classification

Multi-class decision ($d_{out} = k$)

- '*Which languages appear in this tweet?*'
- $\boldsymbol{W}$ is a matrix, $\boldsymbol{b}$ is a vector of k components.

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) = \boldsymbol{x} \cdot \boldsymbol{W} + \boldsymbol{b}$$

Output of multi-class classification

Multi-class decision ($d_{out} = k$)

- '*Which languages appear in this tweet?*'
- $\boldsymbol{W}$ is a matrix, $\boldsymbol{b}$ is a vector of k components.
- The prediction $\hat{\boldsymbol{y}}$ is also a one-hot vector of k components.

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) = \boldsymbol{x} \cdot \boldsymbol{W} + \boldsymbol{b}$$

Output of multi-class classification

Multi-class decision ($d_{out} = k$)

- '*Which languages appear in this tweet?*'
- $\boldsymbol{W}$ is a matrix, $\boldsymbol{b}$ is a vector of k components.
- The prediction $\hat{\boldsymbol{y}}$ is also a one-hot vector of k components.
- The component corresponding to the correct language has the value of 1, others are zeros, for example:
  $\hat{\boldsymbol{y}} = [0, 0, 1, 0]$ (for $k = 4$)

$$\theta = (\boldsymbol{W} \in \mathbb{R}^{d_{in} \times d_{out}}, \boldsymbol{b} \in \mathbb{R}^{d_{out}})$$

$$f(x; W, b) = x \cdot W + b$$

$$\underbrace{\begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}}_{x} \cdot \underbrace{\begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \end{bmatrix}}_{W} + \underbrace{\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}}_{b} = \text{argmax}(\begin{bmatrix} 2 & 2 & 4 \end{bmatrix})$$

$$= 3$$

($\hat{y}$)

Linear classifiers

Log-linear classification

If we care about how confident is the classifier about each decision:

Linear classifiers

Log-linear classification

If we care about how confident is the classifier about each decision:

- Map the predictions to the range of $[0, 1]$...

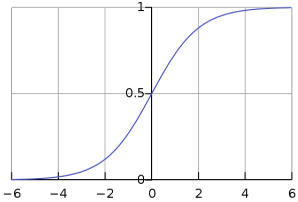
Log-linear classification

If we care about how confident is the classifier about each decision:

- Map the predictions to the range of [0, 1]...
- ...by a squashing function, for example, sigmoid:

$$\hat{y} = \sigma(f(\boldsymbol{x})) = \frac{1}{1 + e^{-(f(\boldsymbol{x}))}} \tag{3}$$

- The result is the probability of the prediction!



$\sigma(\boldsymbol{x})$

Linear classifiers

- For multi-class cases, log-linear models produce probabilities for all classes, for example:
  $\hat{\boldsymbol{y}} = [0.4, 0.1, 0.9, 0.5]$ (for $k = 4$)

Linear classifiers

- For multi-class cases, log-linear models produce probabilities for all classes, for example:
  $\hat{\boldsymbol{y}} = [0.4, 0.1, 0.9, 0.5]$ (for $k = 4$)
- We choose the one with the highest score:

$$\hat{y} = \arg\max_{i} \hat{\boldsymbol{y}}_{[i]} = \hat{\boldsymbol{y}}_{[2]} \qquad (4)$$

Linear classifiers

- For multi-class cases, log-linear models produce probabilities for all classes, for example:
  $\hat{\boldsymbol{y}} = [0.4, 0.1, 0.9, 0.5]$ (for $k = 4$)
- We choose the one with the highest score:

$$\hat{y} = \arg\max_{i} \hat{\boldsymbol{y}}_{[i]} = \hat{\boldsymbol{y}}_{[2]} \tag{4}$$

- But often it is more convenient to transform scores into a probability distribution, using the softmax function:

$$\hat{\boldsymbol{y}} = \mathit{softmax}(\boldsymbol{x}\boldsymbol{W} + \boldsymbol{b}) \tag{5}$$

$$\hat{\boldsymbol{y}}_{[i]} = \frac{e^{(\boldsymbol{x}\boldsymbol{W}+\boldsymbol{b})_{[i]}}}{\sum_j e^{(\boldsymbol{x}\boldsymbol{W}+\boldsymbol{b})_{[j]}}} \tag{6}$$

Linear classifiers

- For multi-class cases, log-linear models produce probabilities for all classes, for example:
  $\hat{\boldsymbol{y}} = [0.4, 0.1, 0.9, 0.5]$ (for $k = 4$)
- We choose the one with the highest score:

$$\hat{y} = \arg\max_i \hat{\boldsymbol{y}}_{[i]} = \hat{\boldsymbol{y}}_{[2]} \tag{4}$$

- But often it is more convenient to transform scores into a probability distribution, using the softmax function:

$$\hat{\boldsymbol{y}} = softmax(\boldsymbol{x}\boldsymbol{W} + \boldsymbol{b}) \tag{5}$$

$$\hat{\boldsymbol{y}}_{[i]} = \frac{e^{(\boldsymbol{x}\boldsymbol{W}+\boldsymbol{b})_{[i]}}}{\sum_j e^{(\boldsymbol{x}\boldsymbol{W}+\boldsymbol{b})_{[j]}}} \tag{6}$$

- $\hat{\boldsymbol{y}} = softmax([0.4, 0.1, 0.9, 0.5]) = [0.22, 0.16, 0.37, 0.25]$
  - (all scores sum to 1)

Linear classifiers

$$f(x; W, b) = x \cdot W + b$$