

Advanced Natural Language Processing

About the course

Grading formula:

Total = Rounding($0.4 * \text{HW} + 0.3 * \text{Midterm} + 0.3 * \text{Exam}$)

- 6 homework assignments
- If the grade ≥ 8 before rounding, you can skip the exam

Links:

Telegram chat: <https://t.me/+fVXUjKGd0VNmYTdi>

GitHub: <https://github.com/ashaba1in/hse-nlp/tree/eng>

About homeworks

All homework will require writing code and a report

Code implementation:

- You can do it as you like
- Better to built it as a project, because
 1. It's easier not to get confused in the code when there's a lot of it
 2. It's easier for us to check structured homework
 3. The project can be posted on git and bragged about

Report:

- PDF document describing the work done
- *We* need it to simplify the check
- *You* need it to learn to summarize your work

Course topics

1. Text Classification
2. Text Generation, RNN
3. Transformers
4. BERT and GPT
5. Quantization and Distillation
6. Modern Transformer Architectures
7. Parameter-Efficient Fine-Tuning
8. Instruction tuning
9. Retrieval-Augmented Generation
10. AI Safety
11. State-Space Models
12. Text Diffusion Models
13. Multimodal Models

Text classification

Agenda

- Types of classification tasks
- Generative and discriminative models
- Neural networks for text

Types of NLP classification tasks

Binary classification

- Is message a spam or not

Multi-class classification

- How urgently do you need to respond to the client?

Multi-label classification

- What is the topic of the paper?

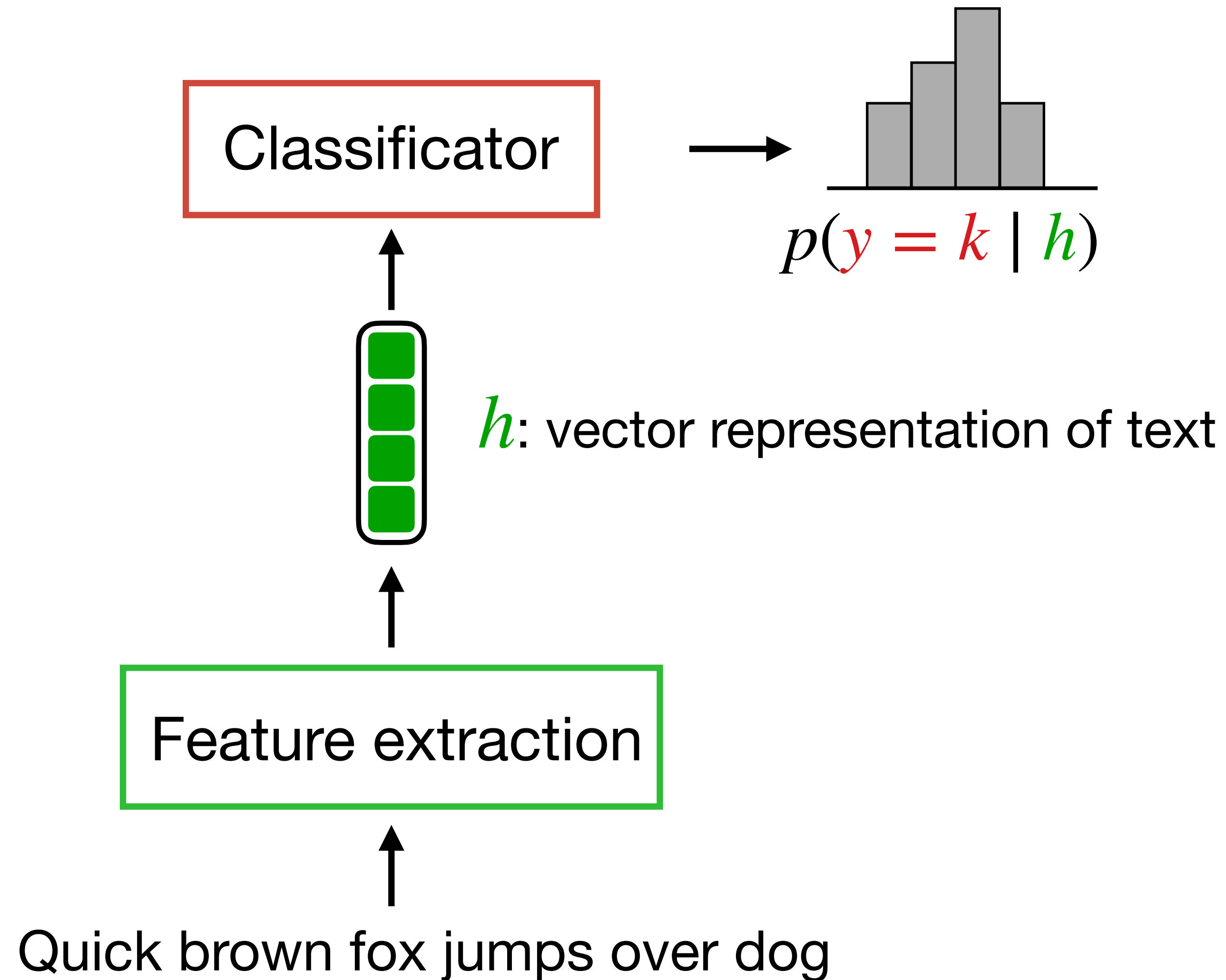
Word classification

- Named Entity Recognition (NER)
- ~~Text generation (spoilers)~~

Datasets for classification

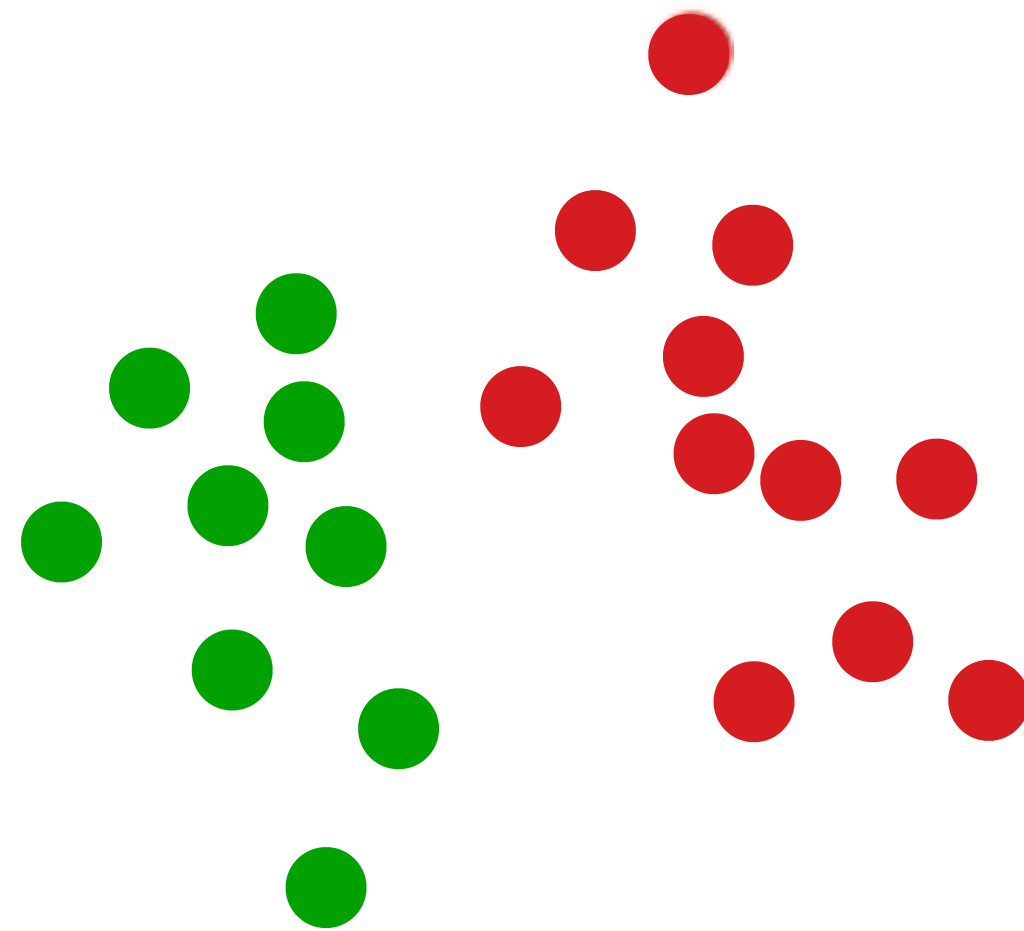
Name	Task	Classes	Size	Average Length	Metric
SST	tonality	5 or 2	11,855	19	Accuracy
Yelp	tonality	5 or 2	280,000	179	Accuracy
IMDb	tonality	2	50,000	271	Accuracy
QQP	paraphrasing	2	404,291	22	F1 / Accuracy
CoLA	grammaticality	2	10,657	9	Matthew's Corr
AG News	topic	4	120,000	44	Accuracy
Yahoo! Answers	topic	10	1,400,000	131	Accuracy
DBpedia	topic	14	560,000	67	Accuracy

General scheme of the solution



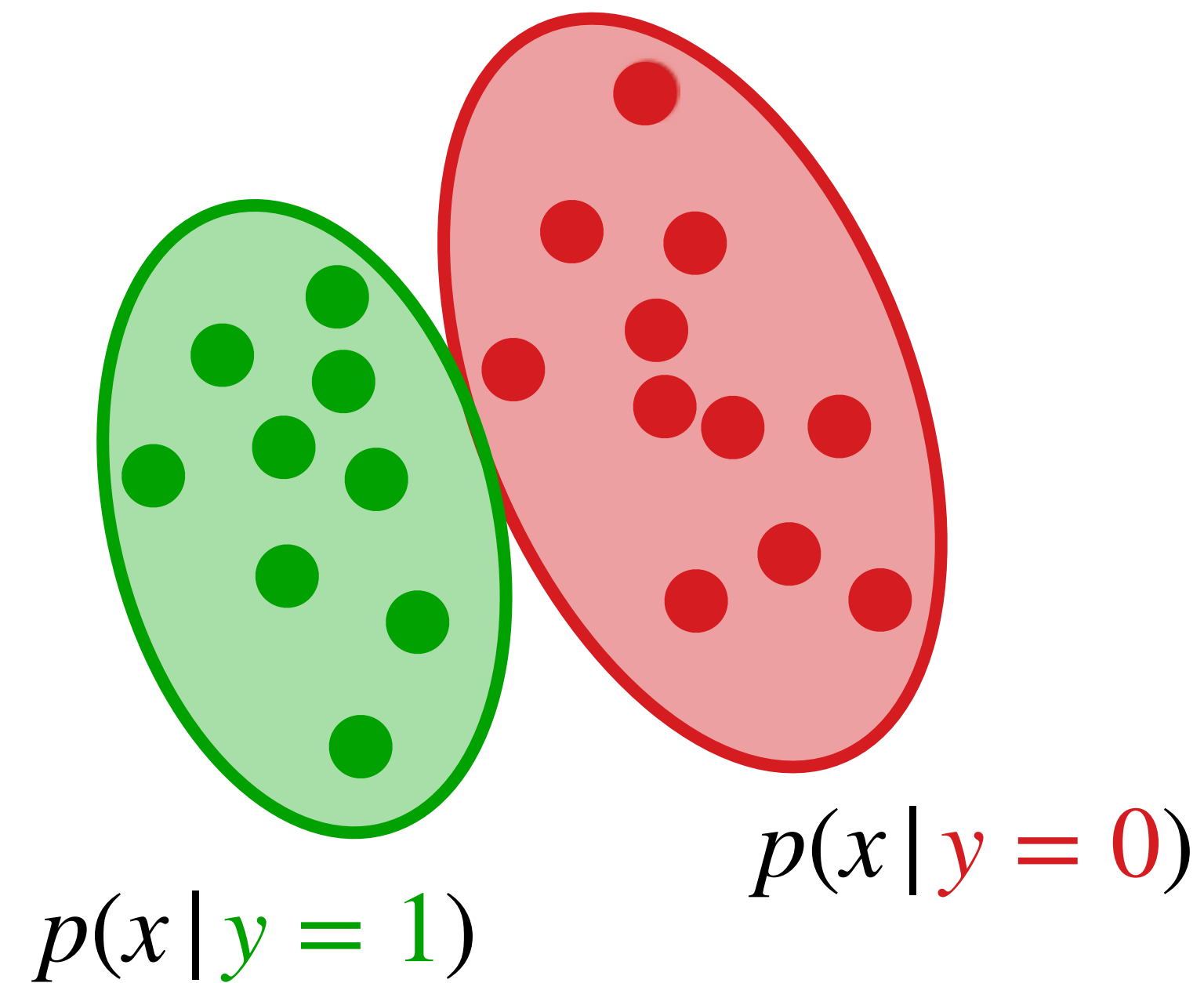
Generative and discriminative models

Example of data distribution for two classes

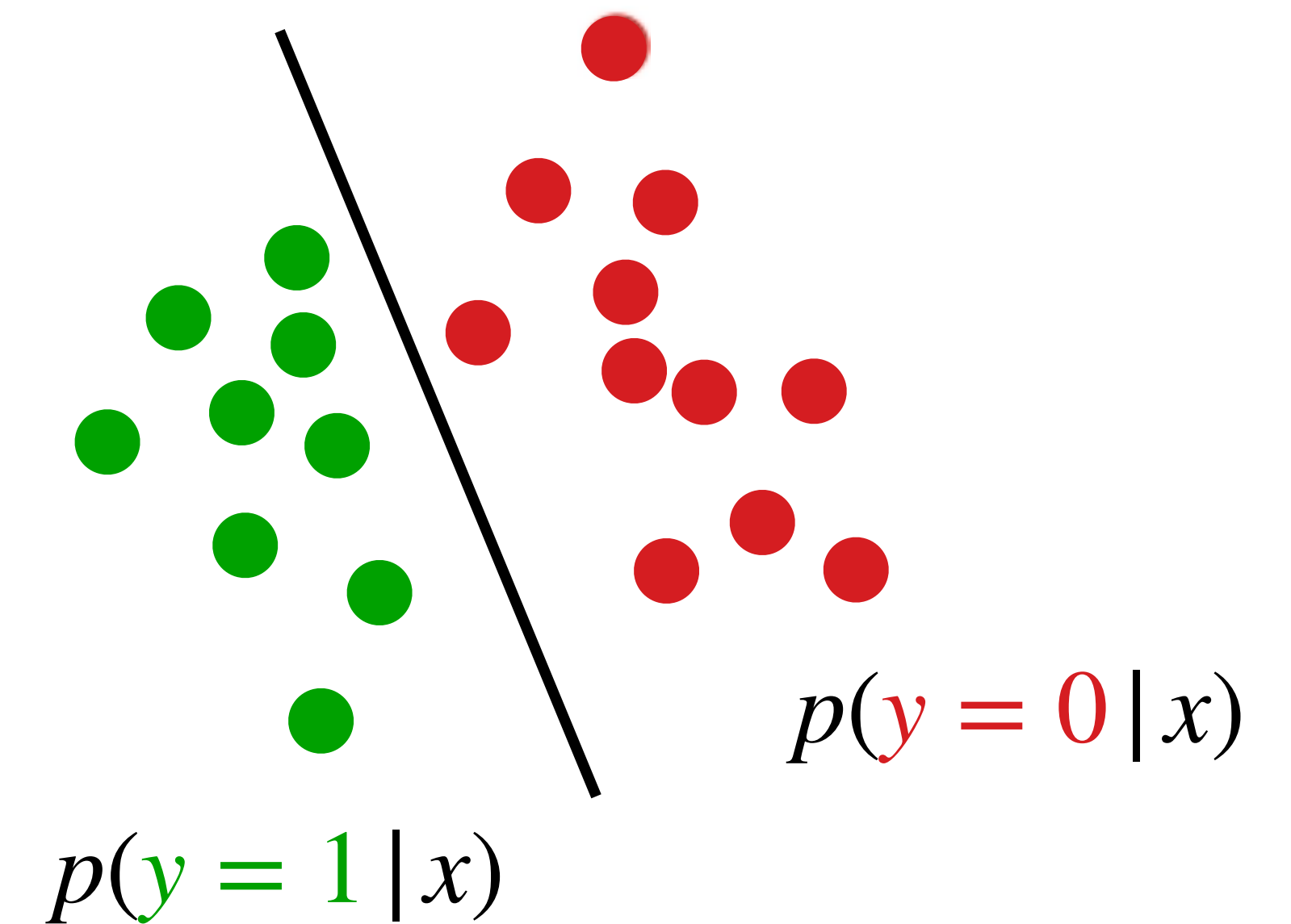


Generative and discriminative models

Generative

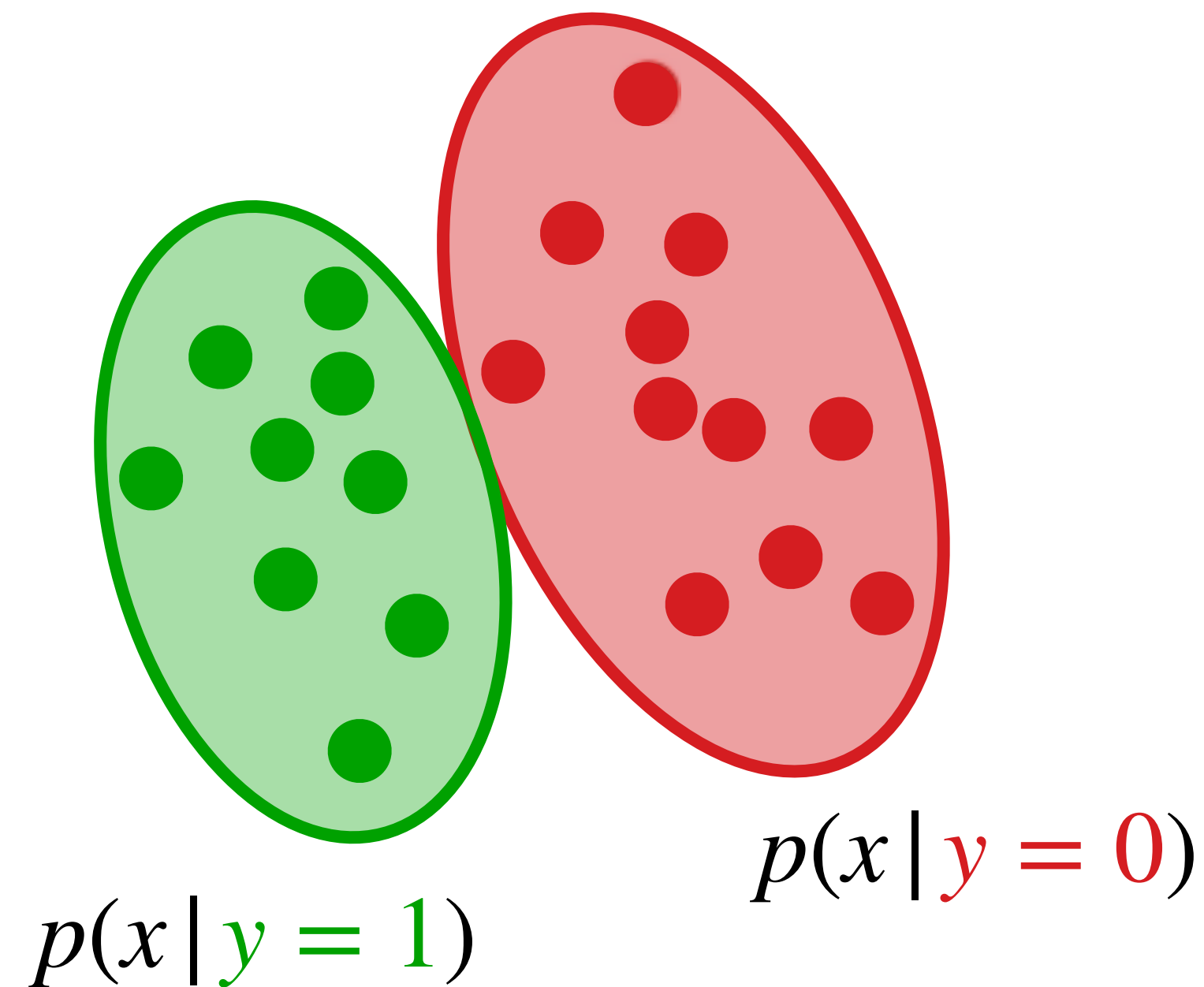


Discriminative



Generative and discriminative models

Generative

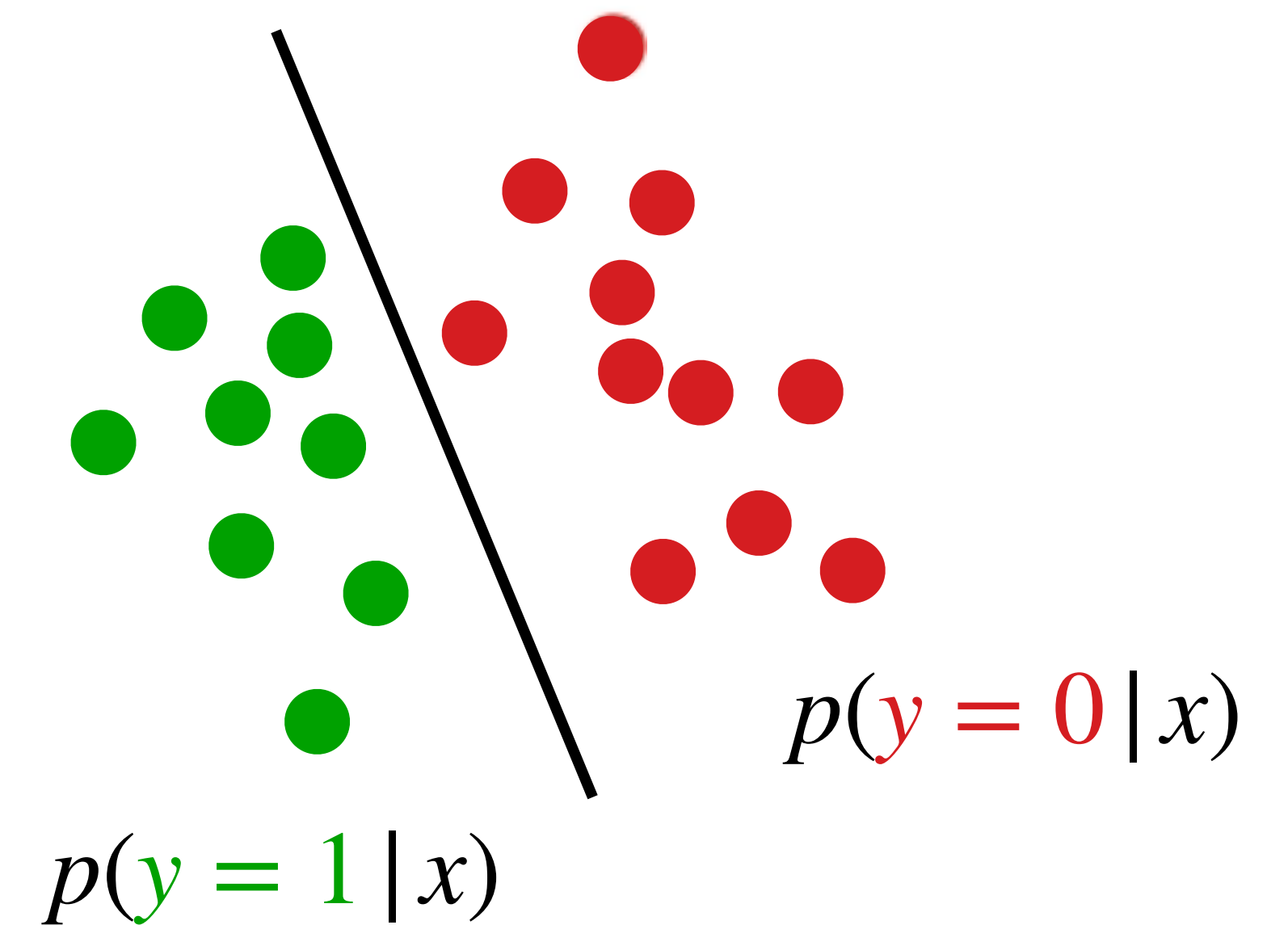


Learn : $p(x | y = k)$

Predict:

$$\hat{y} = \arg \max_y p(y, x) = \arg \max_y p(x | y)p(y)$$

Discriminative



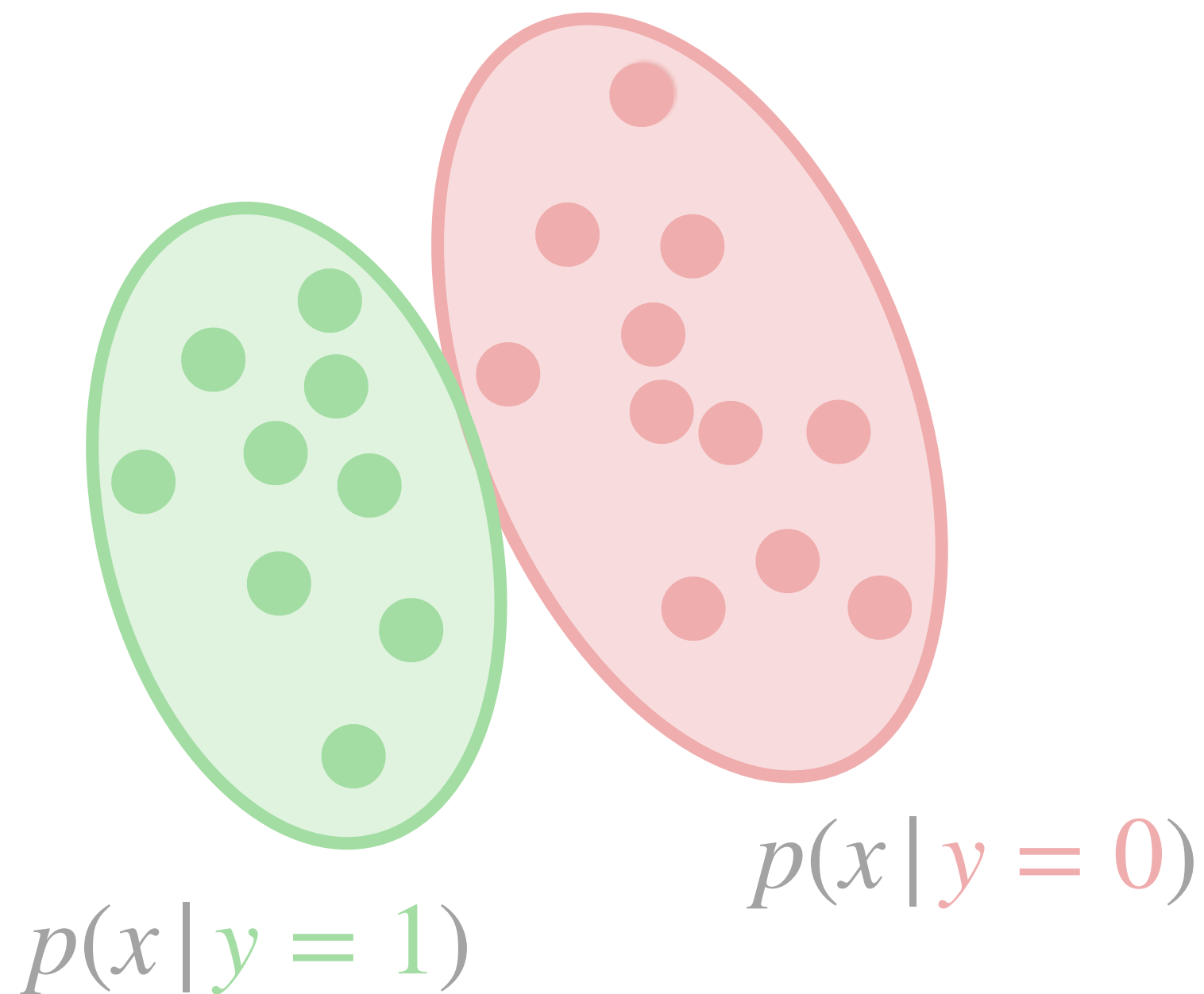
Learn : $p(y = k | x)$

Predict:

$$\hat{y} = \arg \max_y p(y | x)$$

Generative and discriminative models

Generative

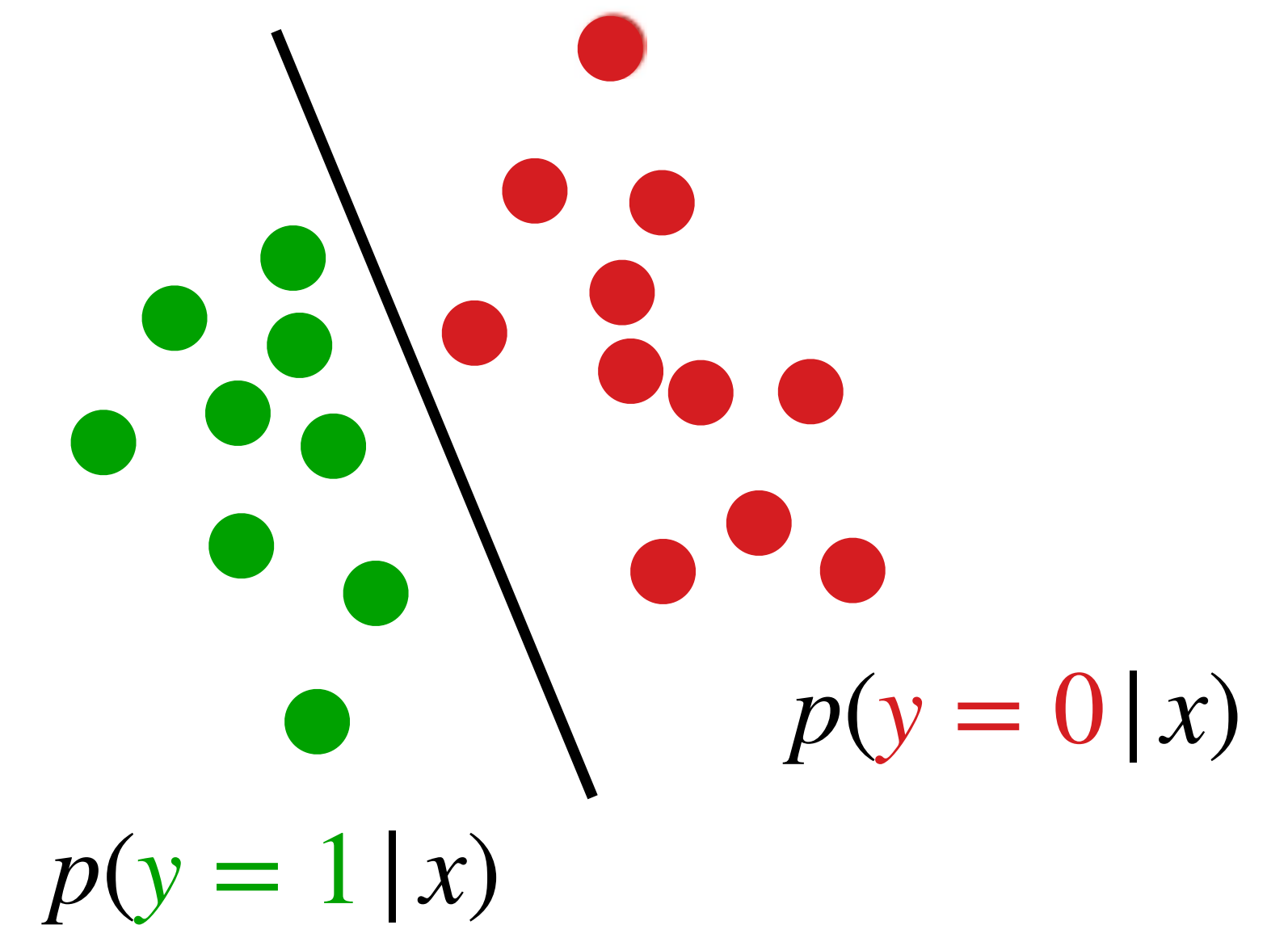


Learn : $p(x | y = k)$

Predict:

$$\hat{y} = \arg \max_y p(y, x) = \arg \max_y p(x | y)p(y)$$

Discriminative



Learn : $p(y = k | x)$

Predict:

$$\hat{y} = \arg \max_y p(y | x)$$

Almost all models
are discriminative

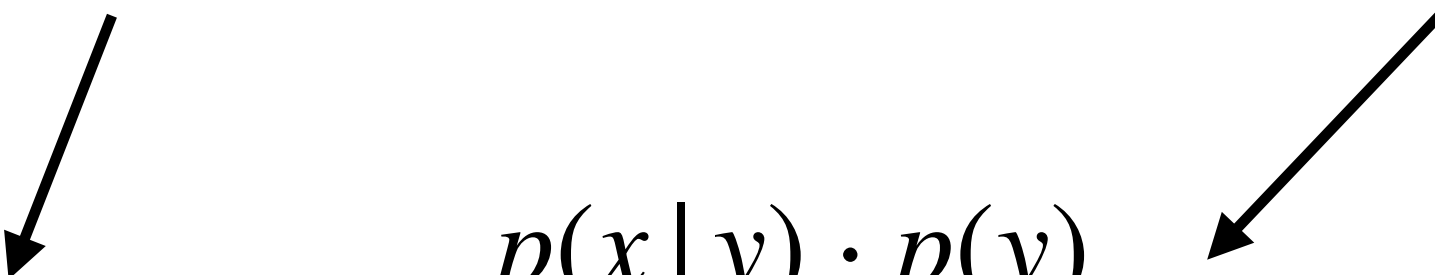
When are generative models useful?

- When there are *outliers* in the data
- When the *distribution* of the test sample is different
- When the *train size is small* and a discriminative model overfits

Naive Bayes

Bayes' theorem

$p(x)$ does not depend on y

$$y = \arg \max_y p(y | x) = \arg \max_y \frac{p(x | y) \cdot p(y)}{p(x)} = \arg \max_y p(x | y)p(y)$$


How to derive $p(x | y)$ and $p(y)$?

How to derive $p(x | y)$ and $p(y)$?

Let's calculate the proportions of each class in the sample

$$p(y = k) = \frac{1}{N} \sum_{i=1}^N [y_i = k]$$

We assume that:

- Word order is not important
- The probability of a word does not depend on neighbouring words

$$p(x | y = k) = p(x_1, \dots, x_n | y = k) \approx \prod_{i=1}^n p(x_i | y = k)$$

Why does this work?

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

For simple tasks, this assumption makes sense!

$$\begin{aligned} p(\text{very good food} \mid y = -) \\ &= p(\text{very} \mid y = -) \\ &\times p(\text{good} \mid y = -) \\ &\times p(\text{food} \mid y = -) \end{aligned}$$

$$\begin{aligned} p(\text{very good food} \mid y = +) \\ &= p(\text{very} \mid y = +) \\ &\times p(\text{good} \mid y = +) \\ &\times p(\text{food} \mid y = +) \end{aligned}$$

Why does this work?

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

For simple tasks, this assumption makes sense!

$$\begin{aligned} p(\text{very good food} \mid y = -) \\ &= p(\text{very} \mid y = -) \\ &\times \frac{p(\text{good} \mid y = -)}{p(\text{food} \mid y = -)} \end{aligned}$$

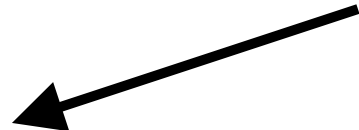
$$\begin{aligned} p(\text{very good food} \mid y = +) \\ &= p(\text{very} \mid y = +) \\ &\times \frac{p(\text{good} \mid y = +)}{p(\text{food} \mid y = +)} \end{aligned}$$

Key words

$$p(\text{good} \mid y = -) < p(\text{good} \mid y = +)$$

How to derive $p(x_i | y)$?

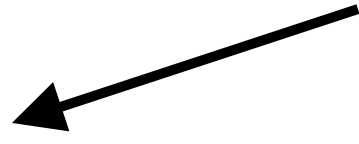
How many times did the word x_i
appear in texts with the label k

$$p(x_i | y = k) = \frac{N(x_i, y = k)}{\sum_{j=1}^{|V|} N(x_j, y = k)}$$


What if $N(x_i, y = k) = 0$?

How to derive $p(x_i | y)$?

How many times did the word x_i
appear in texts with the label k

$$p(x_i | y = k) = \frac{N(x_i, y = k)}{\sum_{j=1}^{|V|} N(x_j, y = k)}$$


What if $N(x_i, y = k) = 0$?

$$\begin{aligned} & p(\text{very good Bratwurst} \mid y = +) \\ &= p(\text{very} \mid y = +) \\ &\times p(\text{good} \mid y = +) \\ &\times \underline{p(\text{Bratwurst} \mid y = +)} = 0 \\ &= 0 \end{aligned}$$

Laplace smoothing

$$p(x_i | y = k) = \frac{N(x_i, y = k) + \delta}{\sum_{j=1}^{|V|} N(x_j, y = k) + |V| \cdot \delta} \quad \delta \in [0,1]$$

If $\delta = 1$, then smoothing is called Laplace smoothing

How to predict?

$$\hat{y} = \operatorname{argmax}_y p(x|y) \cdot p(y)$$

x = "very good food"

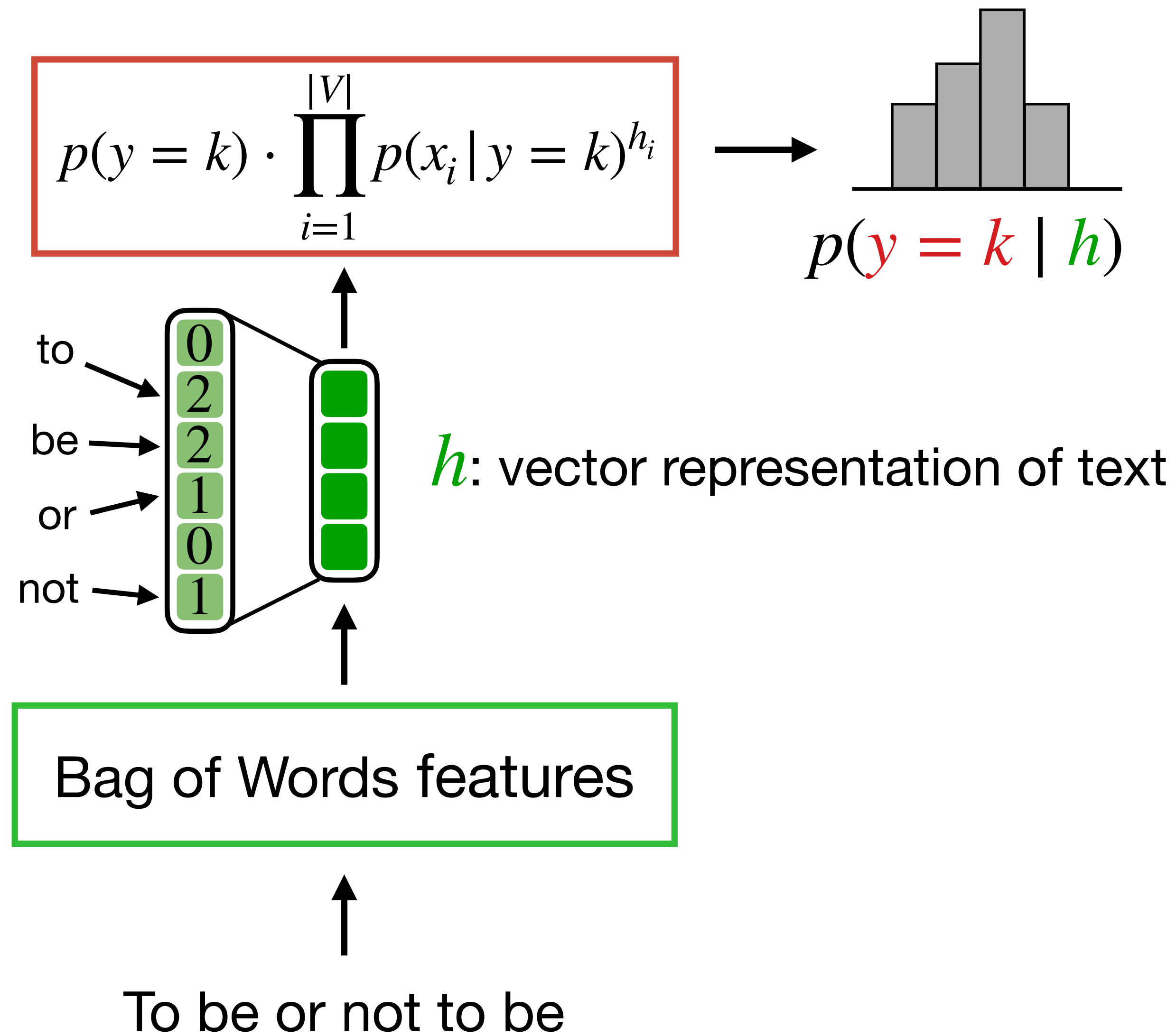
$$\begin{aligned} & p(\text{very good food} \mid y = -) p(y = -) \\ &= p(\text{very} \mid y = -) \\ & \times \frac{p(\text{good} \mid y = -)}{p(\text{food} \mid y = -)} \\ & \times p(y = -) \end{aligned}$$

<

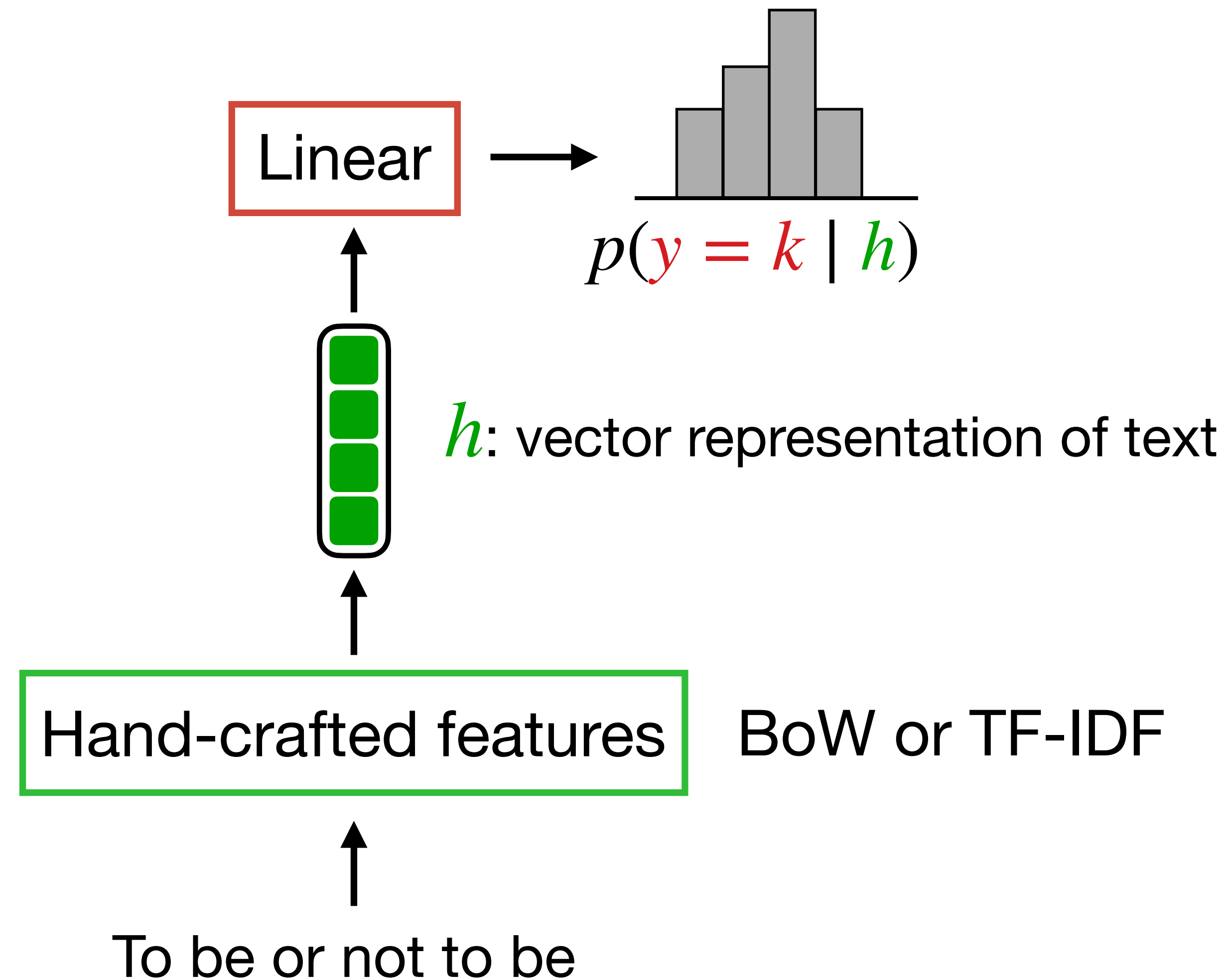
$$\begin{aligned} & p(\text{very good food} \mid y = +) p(y = +) \\ &= p(\text{very} \mid y = +) \\ & \times \frac{p(\text{good} \mid y = +)}{p(\text{food} \mid y = +)} \\ & \times p(y = +) \end{aligned}$$

If $p(y = -) \approx p(y = +)$

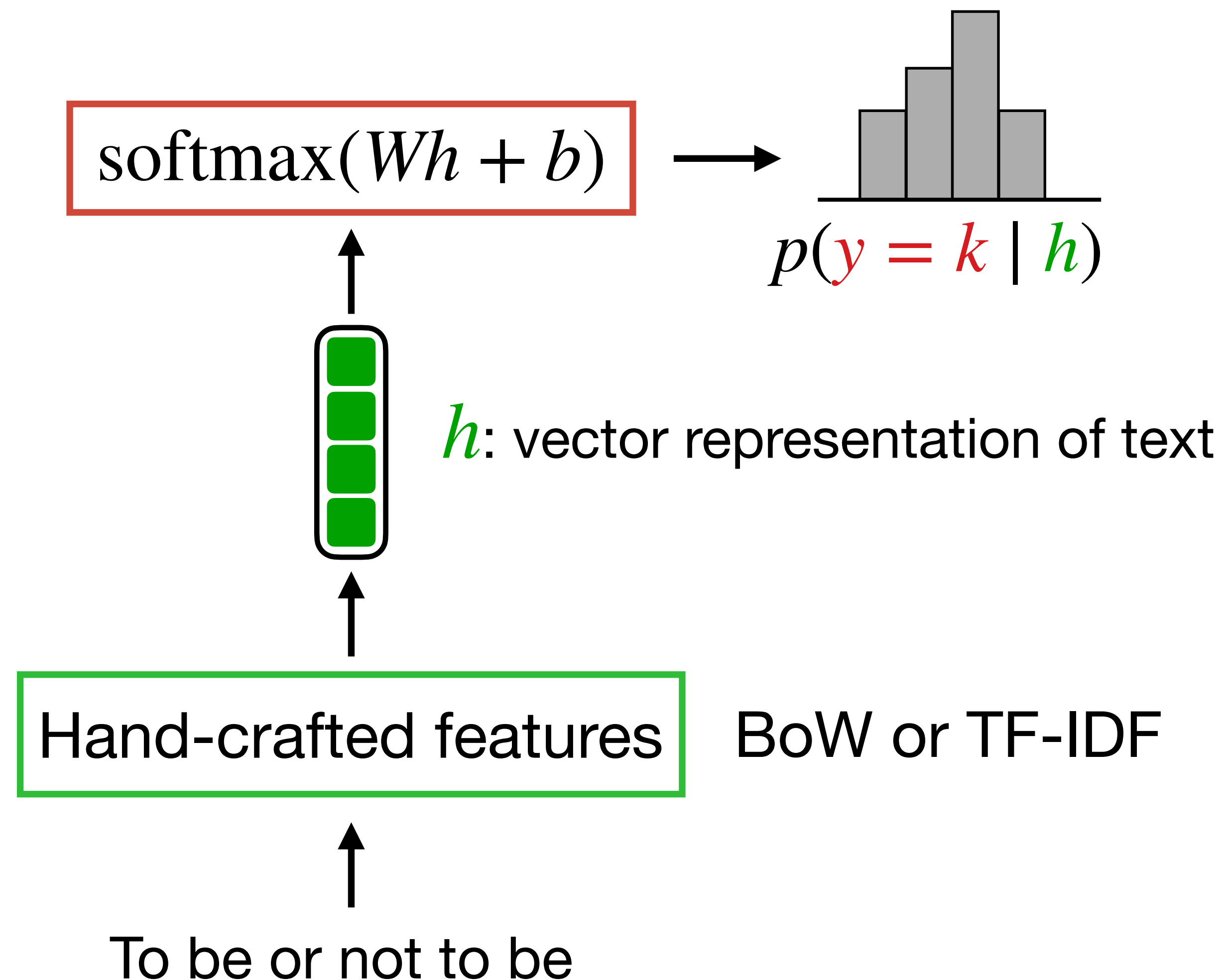
Naive Bayes



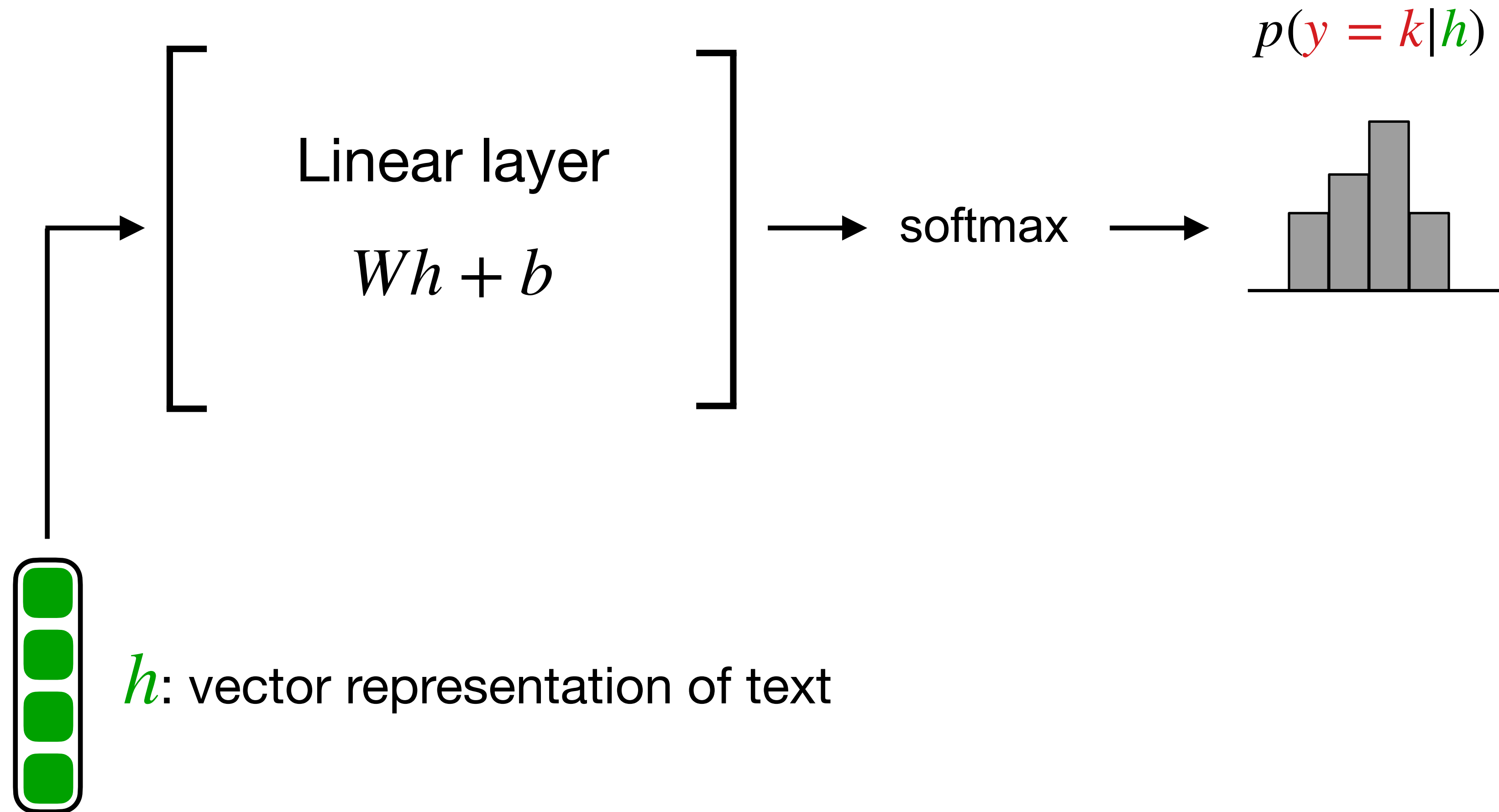
Logistic Regression



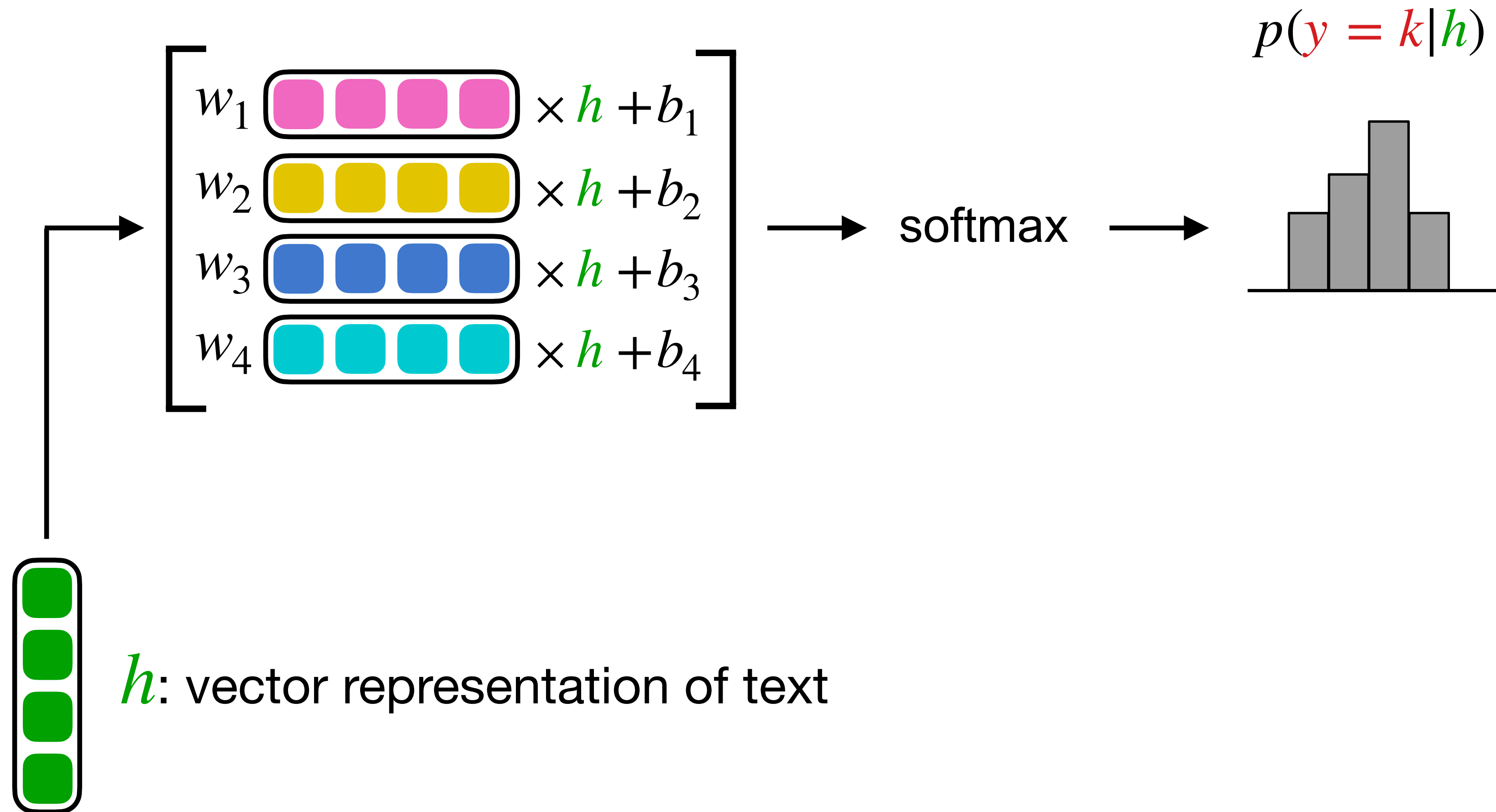
Logistic Regression



Linear layer



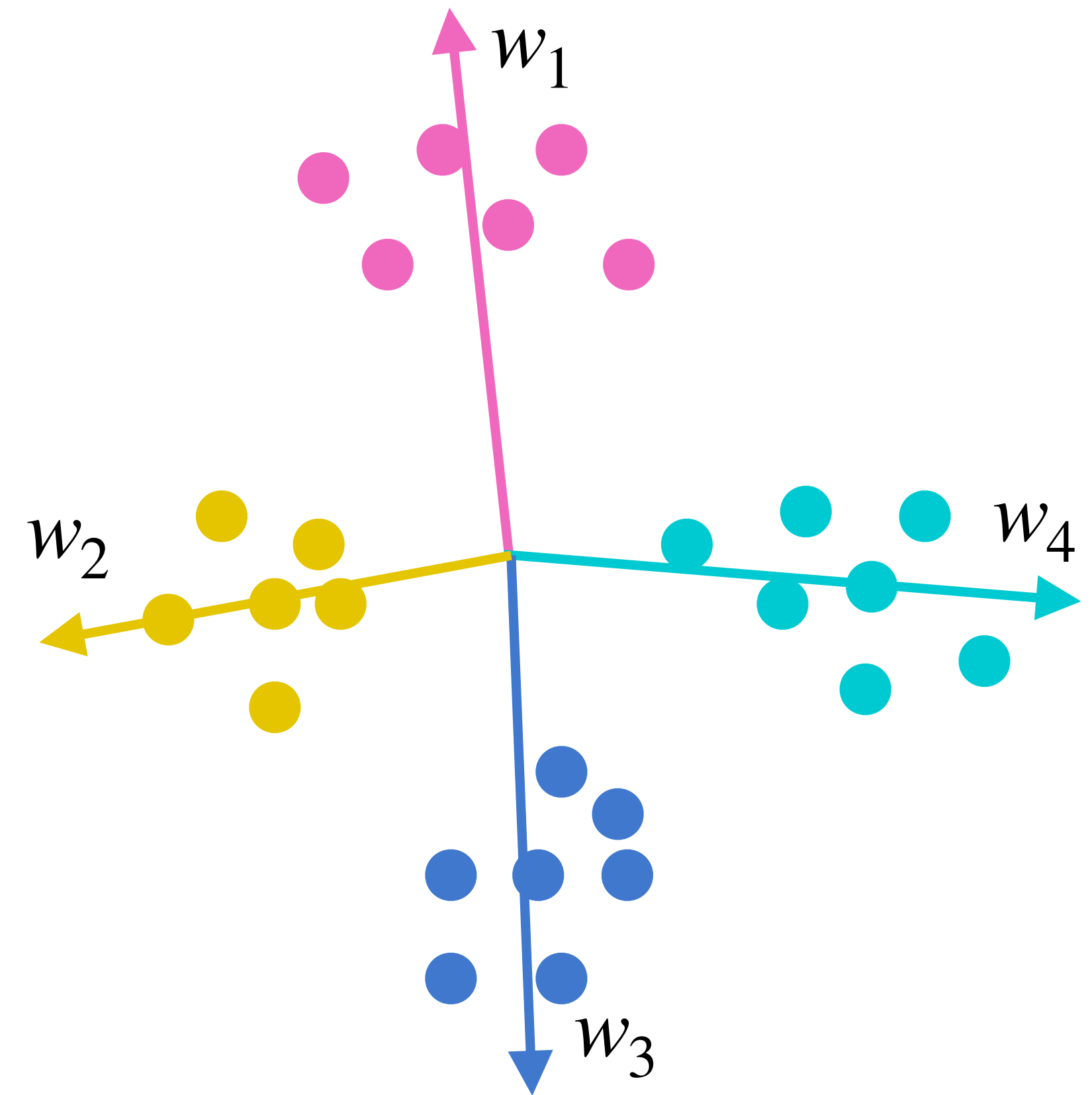
Linear layer



Linear layer

The vectors of the linear layer for each class should be correlated with the vector representations of the class elements.

The scalar product of the vectors is **maximal** when they are **co-directed**.



Disadvantages of approaches

- They don't take into account the relationship between words
- They don't take into account the word order

$p(y = + \mid \text{It is good, not bad})$

\parallel

$p(y = + \mid \text{It is bad, not good})$

- Features are extracted manually

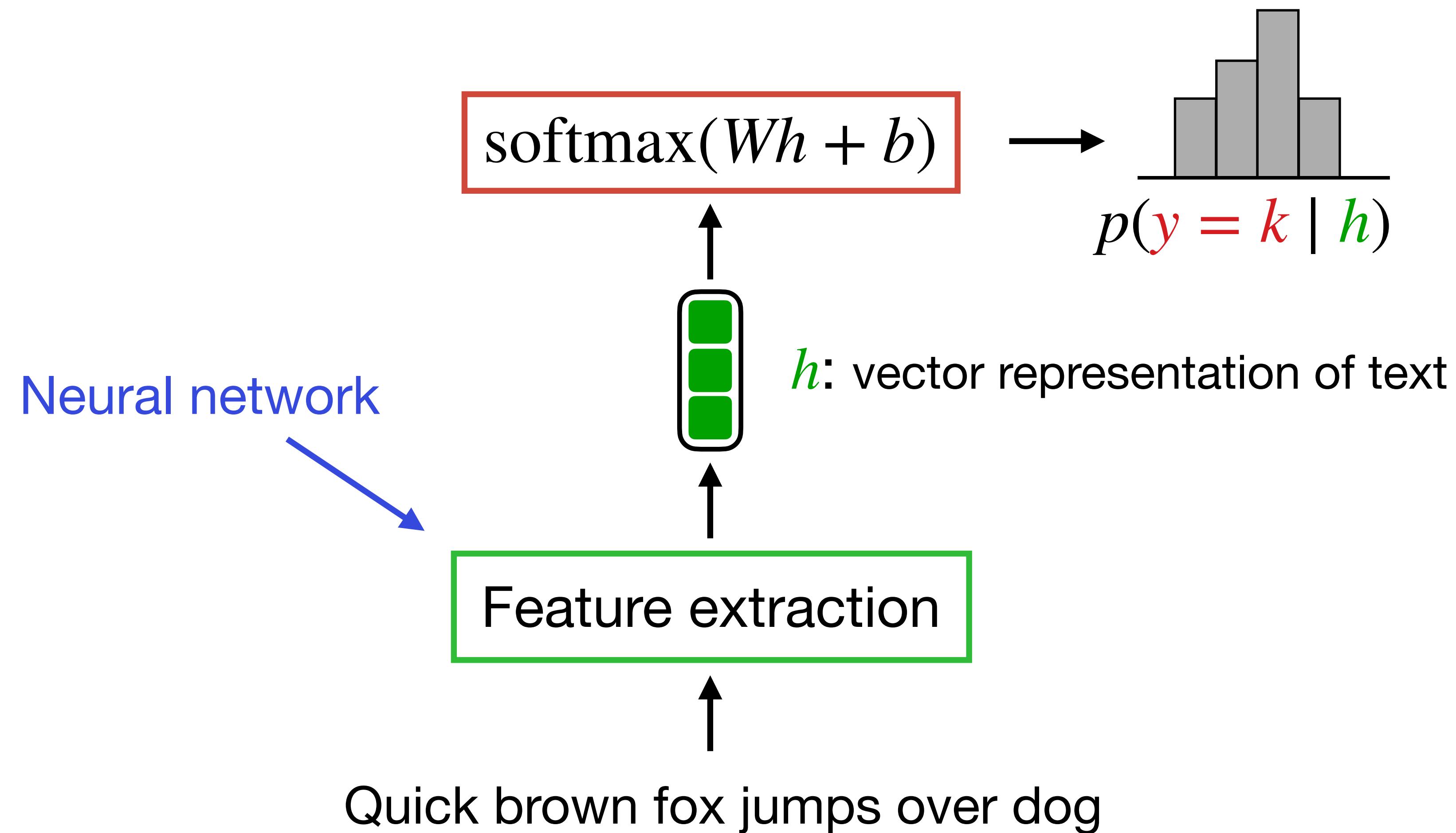
Advantages of approaches

- Fairly good at simple tasks
- Working speed
- Learning time
- Interpretability

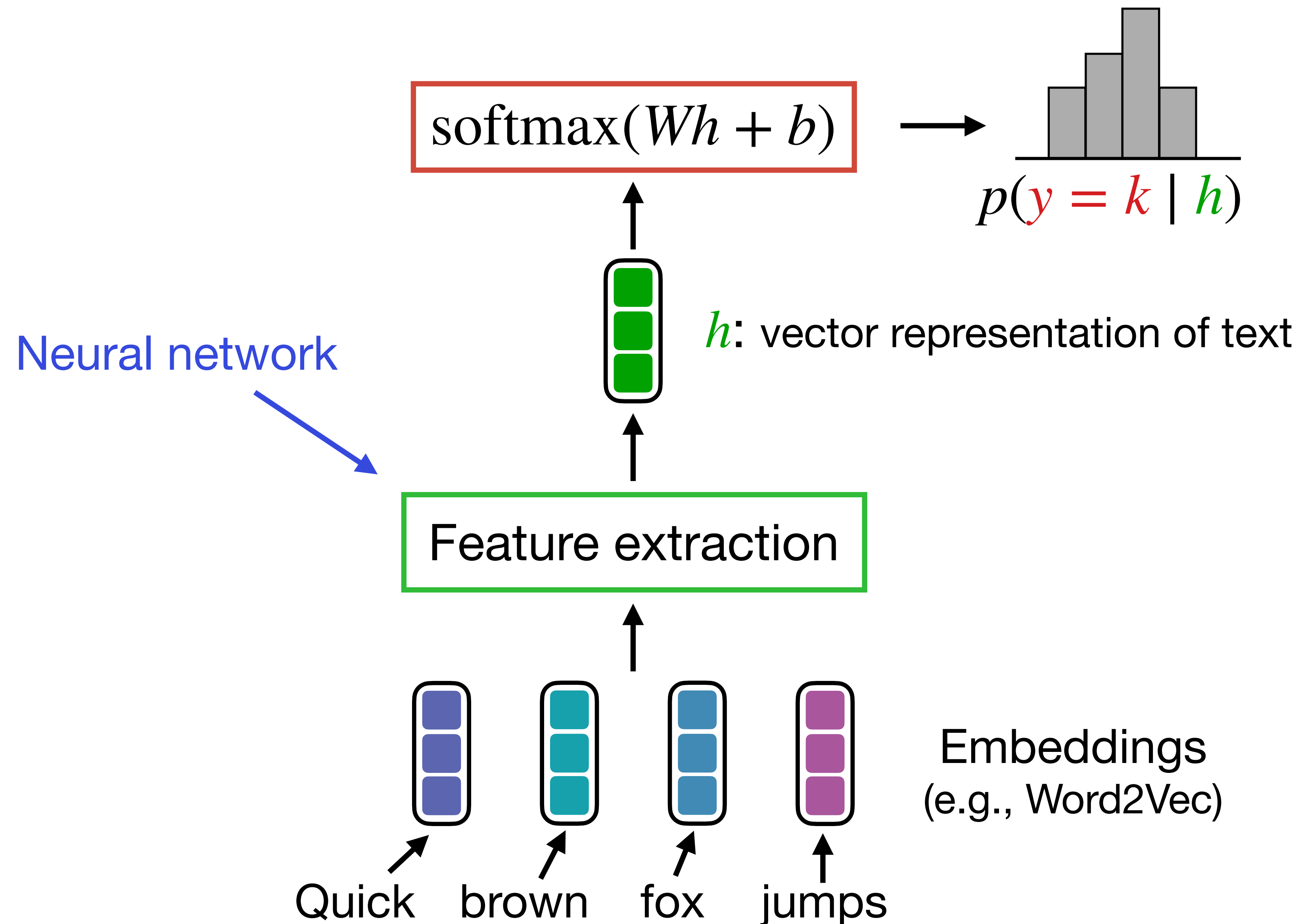
Interpretability is very important when the cost of error is high

- Medical diagnosis
- Sentencing in court

Neural networks

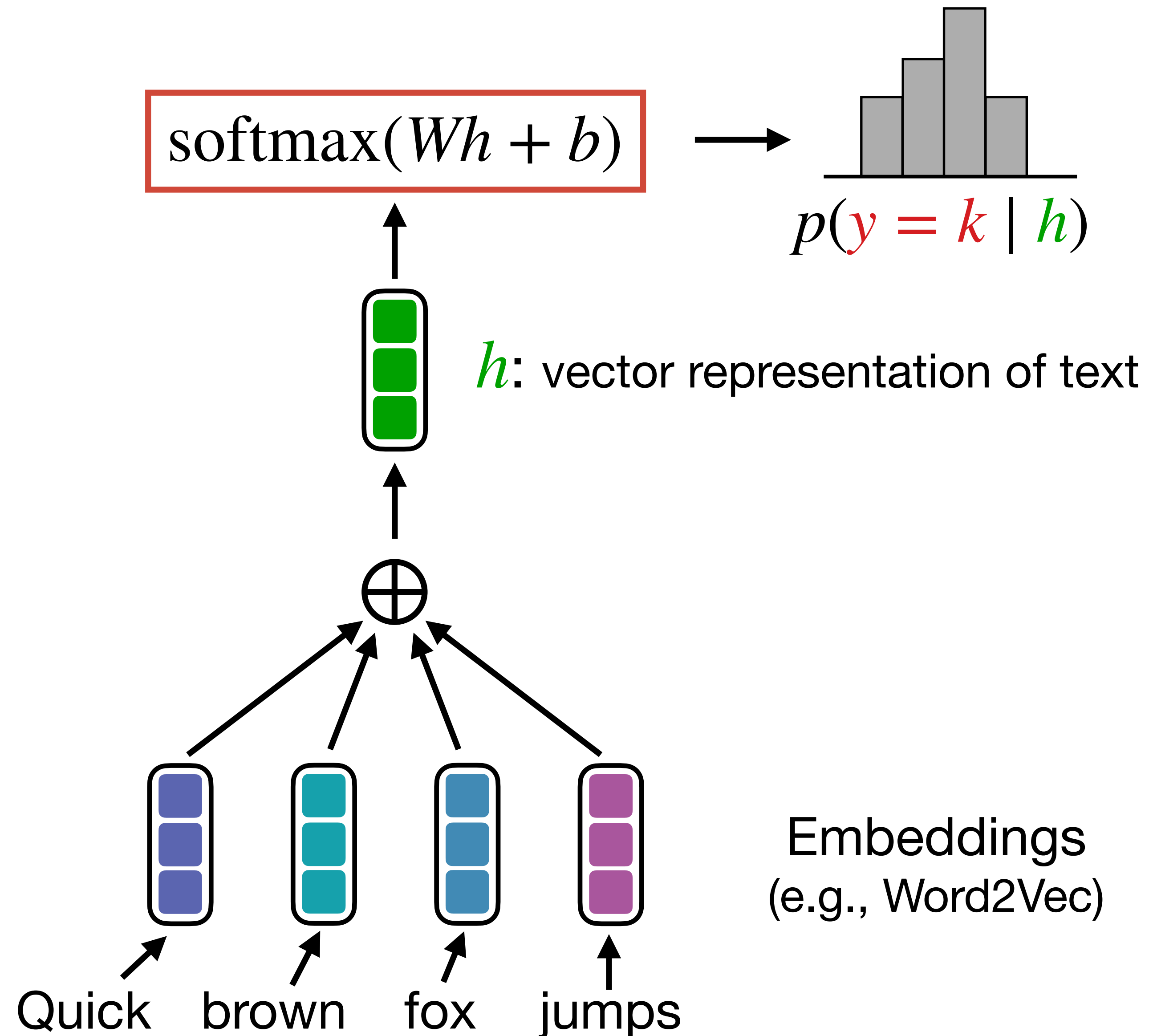


How to extract features?



Bag of Embeddings

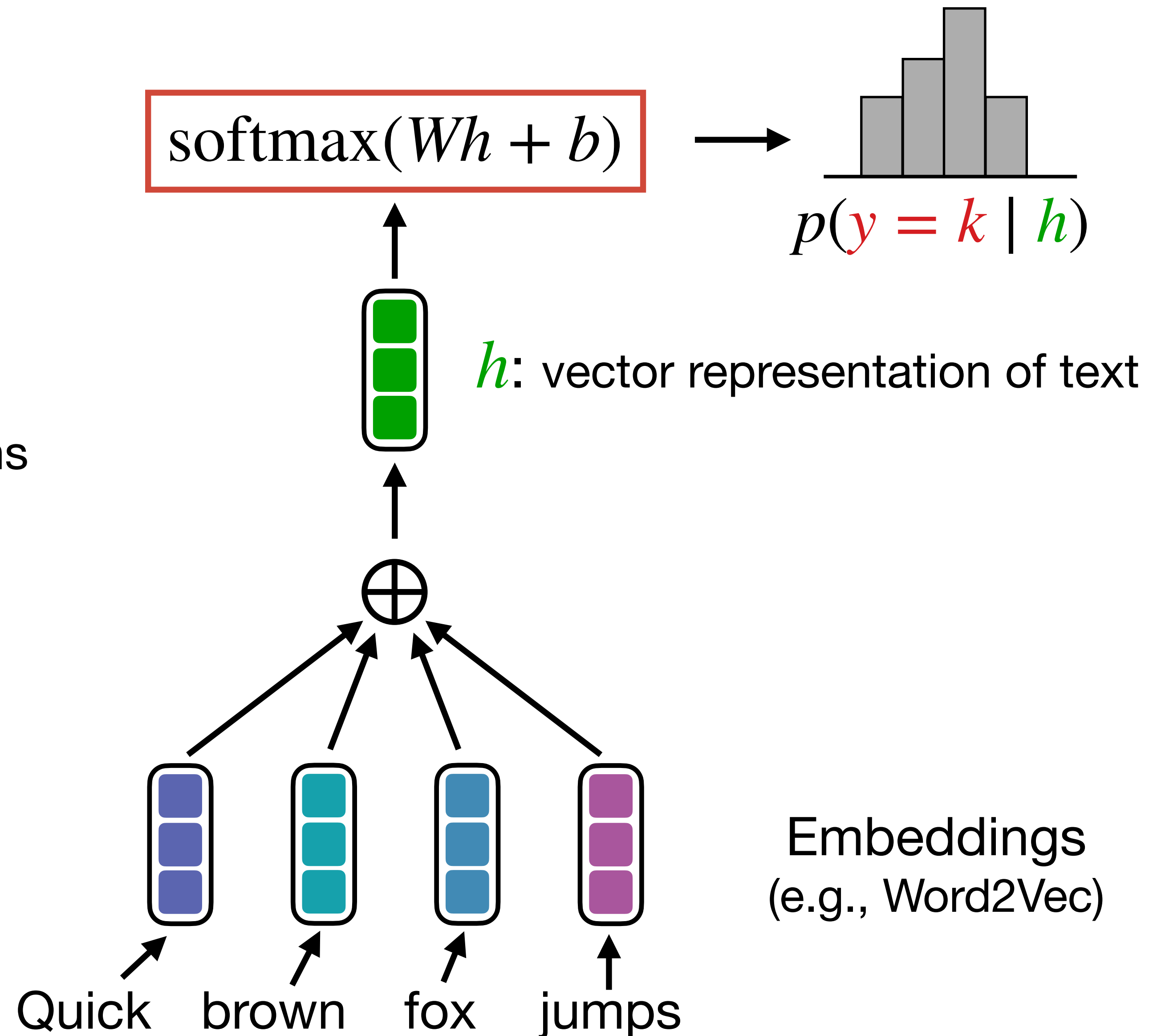
Represent text as a sum of embeddings



Bag of Embeddings

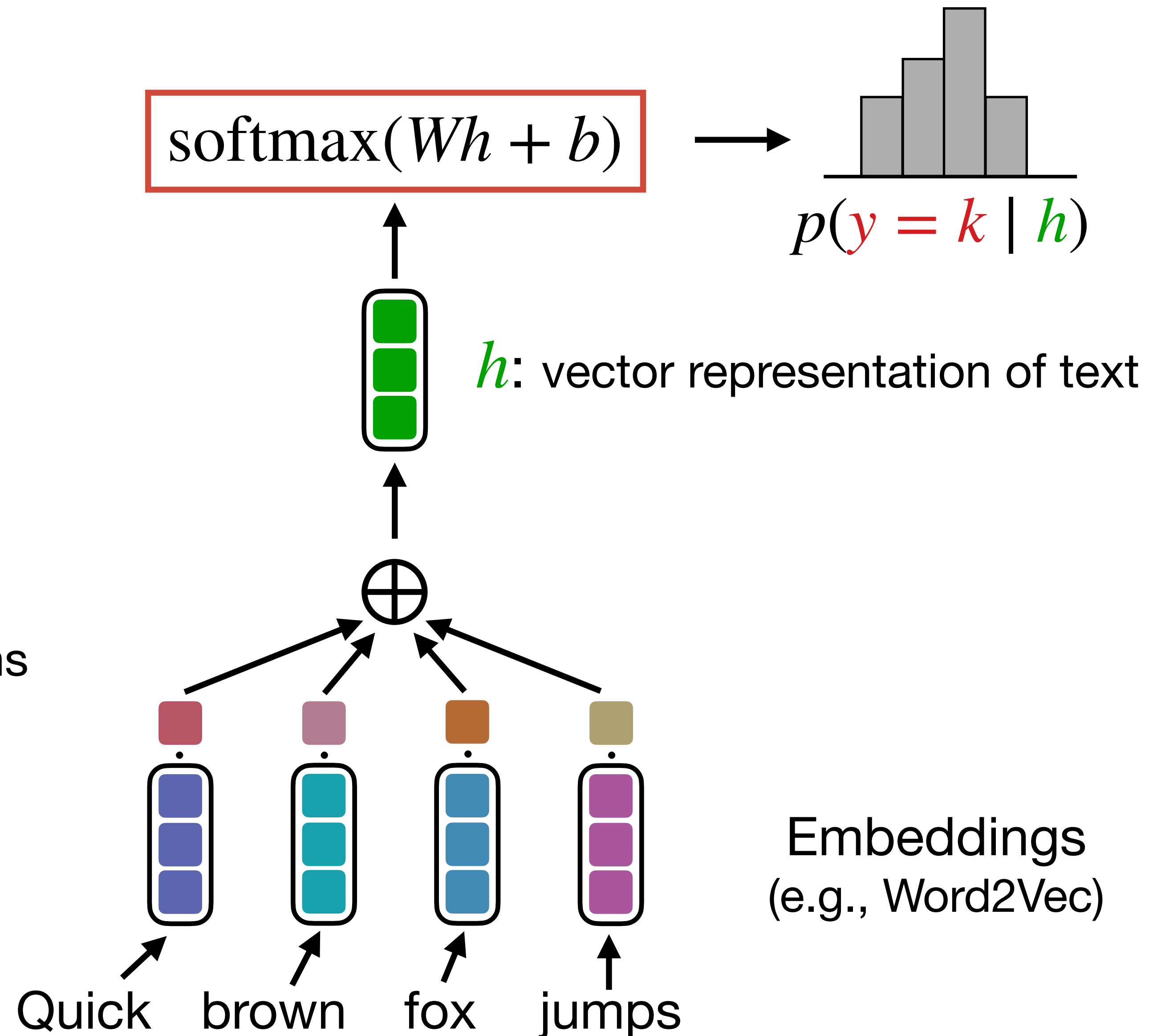
Represent text as a sum of embeddings

- + Very easy to implement
- Do not consider word relations
- Neutral words have large cumulative weight



Weighted Bag of Embeddings

- Multiply embeddings by TF-IDF weights
 - Sum up all embeddings
- + Still easy to implement
- + Less important words have lower weight
- Do not consider word relations



Most important

- Text/word classification is the most popular task
- Very often, simple methods perform well
- Quality directly depends on how good features were extracted