Lecture 2: Logistic Regression and Neural Networks

Pedro Savarese

TTI

2018

Table of Contents

Naive Bayes

- Learn $p(\mathbf{x}, y) = p(y)p(\mathbf{x}|y)$
- Training: Maximum Likelihood Estimation
- Issues?
  - Why learn $p(\mathbf{x}, y)$ if we only use $p(y|\mathbf{x})$?
  - Is Naive assumption realistic? Are words independent given y?
  - Is Bernoulli assumption realistic? What about repeated words?

Naive Bayes

- Let's analyze the prediction rule. Denote 'spam' as 1 and 'not spam' as 0. We predict 1 if:

$$
\begin{aligned}
& p(1|\mathbf{x}) > p(0|\mathbf{x}) \\
& p(1)p(\mathbf{x}|1) > p(0)p(\mathbf{x}|0) \\
& \frac{p(\mathbf{x}|1)}{p(\mathbf{x}|0)} > \frac{p(0)}{p(1)} \\
& \log \frac{p(\mathbf{x}|1)}{p(\mathbf{x}|0)} > \log \frac{p(0)}{p(1)} \\
& \log \frac{\prod_{v \in \mathcal{V}} p(v|1)^{1\{v \in \mathbf{x}\}}(1 - p(v|1))^{1\{v \notin \mathbf{x}\}}}{\prod_{v \in \mathcal{V}} p(v|0)^{1\{v \in \mathbf{x}\}}(1 - p(v|0))^{1\{v \notin \mathbf{x}\}}} > \log \frac{p(0)}{p(1)}
\end{aligned}
$$

Naive Bayes

Denote $1\{v \in \mathbf{x}\}$ as $\phi_v(\mathbf{x})$:

$$\log \frac{\prod_{v\in\mathcal{V}} p(v|1)^{\phi_v(\mathbf{x})}(1-p(v|1))^{1-\phi_v(\mathbf{x})}}{\prod_{v\in\mathcal{V}} p(v|0)^{\phi_v(\mathbf{x})}(1-p(v|0))^{1-\phi_v(\mathbf{x})}} > \log \frac{p(0)}{p(1)}$$

$$\sum_{v\in\mathcal{V}} \left(\phi_v(\mathbf{x}) \log \frac{p(v|1)}{p(v|0)} + (1-\phi_v(\mathbf{x})) \log \frac{1-p(v|1)}{1-p(v|0)}\right) > \log \frac{p(0)}{p(1)}$$

$$\sum_{v\in\mathcal{V}} \left(\phi_v(\mathbf{x}) \log \frac{p(v|1)(1-p(v|0))}{p(v|0)(1-p(v|1))} + \log \frac{1-p(v|1)}{1-p(v|0)}\right) > \log \frac{p(0)}{p(1)}$$

$$\sum_{v\in\mathcal{V}} \phi_v(\mathbf{x}) \log \frac{p(v|1)(1-p(v|0))}{p(v|0)(1-p(v|1))} + \sum_{v\in\mathcal{V}} \log \frac{1-p(v|1)}{1-p(v|0)} > \log \frac{p(0)}{p(1)}$$

$$\sum_{v\in\mathcal{V}} \phi_v(\mathbf{x}) \cdot w_v + c_1 > c_2$$

$$\sum_{v\in\mathcal{V}} \phi_v(\mathbf{x}) \cdot w_v > -b$$

Naive Bayes

So why all this math?

$$\sum_{v \in \mathcal{V}} \phi_v(\mathbf{x}) \cdot w_v + b > 0$$

Is just a **linear function**! A Naive Bayes model is, in reality, equivalent to learning to separate spam / not spam with a line.

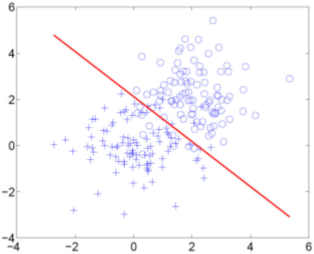


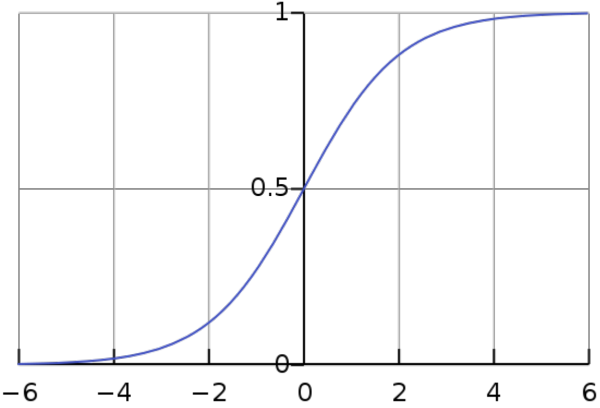
Table of Contents

Logistic Regression

- We only care about $p(y|\mathbf{x})$, so why learn $p(\mathbf{x}, y)$?
- Learn $p(y|\mathbf{x})$ **directly**, through a linear model
- Spam detection: learn $f : \mathbf{x} \to p(\text{spam}|\mathbf{x})$
- First, learn linear **score function** $s : \mathbf{x} \to \mathbb{R}$
  - We want $s(\mathbf{x})$ to be **high** if $p(\text{spam}|\mathbf{x}) \approx 1$
  - And $s(\mathbf{x})$ to be **low** if $p(\text{spam}|\mathbf{x}) \approx 0$
- Linear model for s: $s(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$
- Remaining: map $s(\mathbf{x})$ to $p(\text{spam}|\mathbf{x})$: use sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$. Facts:
  - $\sigma(\infty) = 1$
  - $\sigma(-\infty) = 0$
  - $\sigma(> 0) > 0.5$
  - $\sigma(< 0) < 0.5$

Logistic Regression

Sigmoid function σ:



Logistic Regression

- Model: $p(\text{spam}|\mathbf{x}) = \sigma\Big(\langle \mathbf{w}, \mathbf{x} \rangle + b\Big)$
- Parameters: $\mathbf{w}$ and b
- Learning: Maximum Likelihood Estimation (for **conditional** likelihood!!)

$$
\begin{aligned}
\mathcal{L} &= \prod_{(\mathbf{x},y)\in(\mathbf{X},\mathbf{y})} p(y|\mathbf{x}) \\
&= \prod_{(\mathbf{x},y)\in(\mathbf{X},\mathbf{y})} p(\text{spam}|\mathbf{x})^{1\{y=\text{spam}\}}(1 - p(\text{spam}|\mathbf{x}))^{1\{y=\text{not spam}\}} \\
&= \prod_{(\mathbf{x},y)\in(\mathbf{X},\mathbf{y})} \sigma(\langle \mathbf{w}, \mathbf{x} \rangle + b)^{1\{y=\text{spam}\}}(1 - \sigma(\langle \mathbf{w}, \mathbf{x} \rangle + b))^{1\{y=\text{not spam}}
\end{aligned}
$$

Logistic Regression

- Log-Likelihood:

$$\log \mathcal{L} = \sum_{(\mathbf{x},y)\in(\mathbf{X},\mathbf{y})} \Big(1\{y = \text{spam}\} \log \sigma(\langle \mathbf{w}, \mathbf{x}\rangle + b) + 1\{y = \text{not spam}\} \log(1 - \sigma(\langle \mathbf{w}, \mathbf{x}\rangle + b))\Big)$$

$$\frac{\partial \log \mathcal{L}}{\partial \mathbf{w}} = \sum_{(\mathbf{x},y)\in(\mathbf{X},\mathbf{y})} \mathbf{x}\Big(1\{y = \text{spam}\} - \sigma(\langle \mathbf{w}, \mathbf{x}\rangle + b)\Big)$$

$$\frac{\partial \log \mathcal{L}}{\partial b} = \sum_{(\mathbf{x},y)\in(\mathbf{X},\mathbf{y})} 1\{y = \text{spam}\} - \sigma(\langle \mathbf{w}, \mathbf{x}\rangle + b)$$

- Not possible to solve $\frac{\partial \log \mathcal{L}}{\partial \mathbf{w}} = 0$ analytically: non-linear system

Logistic Regression

- Alternative: **gradient ascent**
- $\frac{\partial \log \mathcal{L}}{\partial \mathbf{w}}$ is direction (in $\mathbf{w}$) that increases $\log \mathcal{L}$ the most
- Gradient ascent: move $\mathbf{w}$ in direction $\frac{\partial \log \mathcal{L}}{\partial \mathbf{w}}$, iteratively:

$$\mathbf{w} \leftarrow \mathbf{w} + \eta \frac{\partial \log \mathcal{L}}{\partial \mathbf{w}}$$

$$\mathbf{w} \leftarrow \mathbf{w} + \eta \sum_{(\mathbf{x},y)\in(\mathbf{X},\mathbf{y})} \mathbf{x}\Big(1\{y = \text{spam}\} - \sigma(\langle \mathbf{w}, \mathbf{x}\rangle + b)\Big)$$

$$b \leftarrow b + \eta \frac{\partial \log \mathcal{L}}{\partial b}$$

$$b \leftarrow b + \eta \sum_{(\mathbf{x},y)\in(\mathbf{X},\mathbf{y})} 1\{y = \text{spam}\} - \sigma(\langle \mathbf{w}, \mathbf{x}\rangle + b)$$

Logistic Regression

- Classifying emails: predict spam if:

$$
\begin{aligned}
p(\text{spam}|\mathbf{x}) &> p(\text{not spam}|\mathbf{x}) \\
p(\text{spam}|\mathbf{x}) &> 1 - p(\text{spam}|\mathbf{x}) \\
p(\text{spam}|\mathbf{x}) &> 0.5 \\
\sigma(\langle \mathbf{w}, \mathbf{x} \rangle + b) &> 0.5 \\
\langle \mathbf{w}, \mathbf{x} \rangle + b &> 0
\end{aligned}
$$

Looks familiar?

Table of Contents

Input Encoding

- $\mathbf{x}$ and $\mathbf{w}$ must have same dimensionality
- Must represent data as fixed-dimensional vector
- Common encoding in Natural Language Processing: $\mathbf{x}(\text{e-mail})_i = 1\{\text{word } v_i \text{ of the vocabulary } \mathcal{V} \text{ is in e-mail}\} = \phi_{v_i}(\mathbf{x})$
- Then $\mathbf{x}$(e-mail) is always $|\mathcal{V}|$-dimensional

Spam Detection

Quick coding session:

- Implement input encoding: transform emails into fixed-length vector
- Implement logistic model $p(\text{spam}|\mathbf{x}) = \sigma(s(\mathbf{x}))$
- Implement gradient computation for $\mathbf{w}$ and b
- Implement gradient ascent and train model

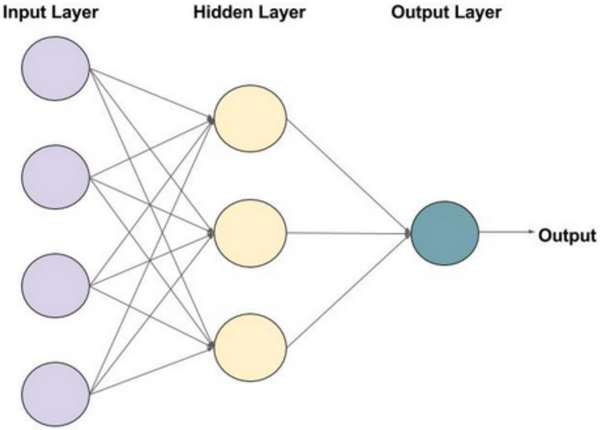
Table of Contents

Sigmoid Feedforward Neural Networks

- Logistic Regression: prediction is given by linear function
- Many simple problems are **not linearly separable**, example:
  - $\mathcal{V} = \{\text{human, dog}\}$
  - Possible combinations: $\{\}, \{\text{human}\}, \{\text{dog}\}, \{\text{human, dog}\}$
  - Task: classify single-word email versus non-single word
- Idea: model $p(\text{spam}|\mathbf{x})$ with **non-linear function** instead
- Sigmoid Feedforward Neural Networks: "composition of logistic models"

Sigmoid Feedforward Neural Networks

Sigmoid Feedforward Neural Network:



Sigmoid Feedforward Neural Networks

Sigmoid Feedforward Neural Network:

$$h_i = \sigma\left(\langle \mathbf{w}_i^{(h)}, \mathbf{x} \rangle + b_i^{(h)}\right)$$

$$p(y|\mathbf{x}) = \sigma\left(\langle \mathbf{w}^{(y)}, \mathbf{h} \rangle + b^{(y)}\right)$$

Each h_i is called a **hidden neuron**. Note that we can have as many hidden neurons as desired.

Sigmoid Feedforward Neural Networks

Advantages of Neural Networks:

- Cybenko'89: given **enough hidden neurons**, neural networks can approximate **any** function arbitrarily well
- Easy to train: gradient ascent / descent
- In practice: hidden neurons act as **feature extractors**
- Can control model complexity by adding more hidden neurons or **more layers** (Deep Learning)

Quick Coding Session:

- Understand neural network code, and difference from logistic regression
- Train neural network. Do same settings from Logistic Regression work?
- Add more layers and train network again

Is data often not linearly separable?

- Problem: vocabulary of 10 words
- Task: given email, classify as 1 if it has 5 or 6 words
- Is problem easy?