Deep Learning for Natural Language Processing

Cornelia Caragea

Computer Science

University of Illinois at Chicago

Credits for slides: Piyush Rai, Andrew Ng

Logistic Regression

Outline

Lecture Structure:

- Logistic Regression
- Gradient Descent

Last Time: Linear Regression

- Given training data $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \cdots, (\mathbf{x}^{(N)}, y^{(N)})\}$
- Inputs $\mathbf{x}^{(i)}$: n-dimensional vectors (R^n), targets $y^{(i)}$: scalars (R)
- In the linear model: target is a linear function of the model parameters

$$y = h_{\mathbf{w}}(\mathbf{x}) = w_0 + \sum_{j=1}^{n} w_j x_j = w_0 + \mathbf{w}^T \mathbf{x}$$

- $\mathbf{x} = [x_1, \cdots, x_n]$
- w_j's and w_0 are the model parameters (w_0 is an offset)
  - $\mathbf{w} = [w_1, \cdots, w_n]$, the **weight vector** (to be learned from $\mathcal{D}$)
  - Parameters define the mapping from the inputs to targets

How to choose **w**?

Linear Regression: Gradient Descent Solution

- One solution: *Iterative* minimization of the cost function

  Cost Function:

  $$E(w_0, w_1, \cdots, w_n) = \frac{1}{2N} \sum_{i=1}^{N} \left(h_w(x^{(i)}) - y^{(i)} \right)^2,$$

  Goal:

  $$\min_{w_0, w_1, \cdots, w_n} E(w_0, w_1, \cdots, w_n),$$

- How: Using Gradient Descent (GD)
- A general recipe for iteratively optimizing similar loss functions
- Gradient Descent rule:
  - Initialize the weight vector $\mathbf{w} = \mathbf{w}^0$
  - Update $\mathbf{w}$ by moving along the direction of negative gradient $-\frac{\partial E}{\partial \mathbf{w}}$

From Regression to Classification

Linear Classification

- **Goal:** Assign input vector $\mathbf{x}$ to one of the K discrete classes $\mathcal{C}_k$.
- Generally, the input space is divided into decision regions, whose boundaries are called *decision boundaries.*
- For linear models, decision boundaries are linear functions of the input vector $\mathbf{x}$.
- Data sets whose classes can be separated *exactly* by linear decision boundaries are said to be linearly separable.

Linear Classification

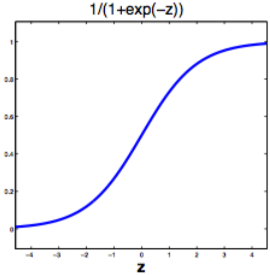
- **Goal:** Assign input vector $\mathbf{x}$ to one of the K discrete classes $\mathcal{C}_k$.
- Generally, the input space is divided into decision regions, whose boundaries are called *decision boundaries.*
- For linear models, decision boundaries are linear functions of the input vector $\mathbf{x}$.
- Data sets whose classes can be separated *exactly* by linear decision boundaries are said to be linearly separable.
- Examples of binary classification ($y \in \{0, 1\}$):
  - Email: spam / not spam?
  - Tumor: malignant / benign?

Linear Classification

- In regression problems, y is a real number, $h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ (in the simplest case), where $h_{\mathbf{w}}(\mathbf{x})$ can be any real-valued number.
- In classification problems, we wish to predict discrete class labels, or more generally posterior probabilities that lie in the range $(0, 1)$, i.e., $0 \leq h_{\mathbf{w}}(\mathbf{x}) \leq 1$.
  - *Generalized linear models*: transform the linear function of $\mathbf{w}$ using a nonlinear function $\sigma(\cdot)$: $h_{\mathbf{w}}(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x})$.

Logistic Regression for Binary Classification

- Generalized linear model for classification where $\sigma(\cdot)$ is the logistic sigmoid function, i.e., $\sigma(z) = \frac{1}{1+e^{-z}}$



- Properties of σ:
  - Symmetry: $\sigma(-z) = 1 - \sigma(z)$
  - Inverse: $z = ln(\sigma/1 - \sigma)$ (aka logit function)
  - Derivative: $d\sigma/dz = \sigma(1 - \sigma)$

Logistic Regression for Binary Classification

Transform the linear function of $\mathbf{w}$ using $\sigma(\cdot)$

- Hypothesis Representation for Logistic Regression:

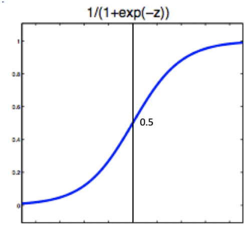
$$h_{\mathbf{w}}(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}},$$

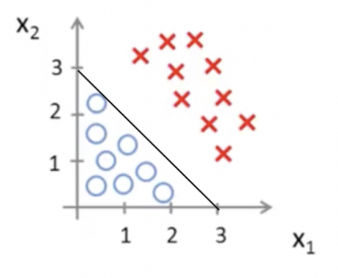
  where $\mathbf{x}$ is a feature vector

- Hypothesis Output Interpretation:
  - $h_{\mathbf{w}}(\mathbf{x}) = P(y = 1|\mathbf{x}, \mathbf{w})$ - the confidence in the predicted label
  - $P(y = 0|\mathbf{x}, \mathbf{w}) = 1 - P(y = 1|\mathbf{x}, \mathbf{w})$
- Logistic regression seen as probabilistic discriminative model
  - Directly models conditional probabilities $P(y|\mathbf{x})$

Decision Boundary

- How does the decision boundary look like for Logistic Regression?
  - Suppose predict $y = 1$ if $h_{\mathbf{w}}(\mathbf{x}) \geq 0.5 \Leftrightarrow \mathbf{w}^T\mathbf{x} \geq 0$
  - Predict $y = 0$ if $h_{\mathbf{w}}(\mathbf{x}) < 0.5 \Leftrightarrow \mathbf{w}^T\mathbf{x} < 0$
- Decision boundary: $\mathbf{w}^T\mathbf{x} = 0$.
  - Hence, the decision boundary is linear $\Rightarrow$ Logistic Regression is a linear classifier (note: it is possible to kernelize and make it nonlinear)





Cost Function

- Training set: $\{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \cdots, (\mathbf{x}^{(N)}, y^{(N)})\}$
- Hypothesis representation:

$$h_{\mathbf{w}}(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}}$$

- How to choose parameters $\mathbf{w}$?

Cost Function

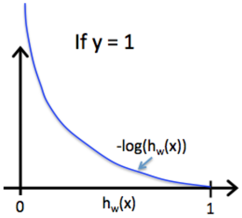
- Training set: $\{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \cdots, (\mathbf{x}^{(N)}, y^{(N)})\}$
- Hypothesis representation:

$$h_{\mathbf{w}}(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$

- How to choose parameters $\mathbf{w}$?
- Previously, for linear regression, $E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{N} (h_{\mathbf{w}}(\mathbf{x}^{(i)}) - y^{(i)})^2$
- For logistic regression, $E(\mathbf{w}) = \sum_{i=1}^{N} Cost(h_{\mathbf{w}}(\mathbf{x}^{(i)}), y^{(i)})$

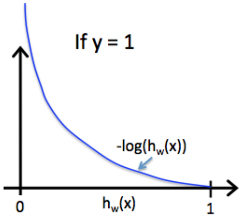
Cost Function

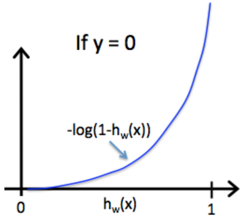
- $Cost(h_{\mathbf{w}}(\mathbf{x}), y) = -\log(h_{\mathbf{w}}(\mathbf{x}))$ if $y = 1$
- $Cost(h_{\mathbf{w}}(\mathbf{x}), y) = -\log(1 - h_{\mathbf{w}}(\mathbf{x}))$ if $y = 0$
- If $y = 1$
  - if $h_{\mathbf{w}}(\mathbf{x}) = 1$, $Cost = 0$
  - If $h_{\mathbf{w}}(\mathbf{x}) \to 0$, $Cost \to \infty$
  - Captures intuition that if $h_{\mathbf{w}}(\mathbf{x}) = 0$, but $y = 1$, we will penalize the learning algorithm by a very large cost.



Cost Function

- $Cost(h_{\mathbf{w}}(\mathbf{x}), y) = -\log(h_{\mathbf{w}}(\mathbf{x}))$ if $y = 1$
- $Cost(h_{\mathbf{w}}(\mathbf{x}), y) = -\log(1 - h_{\mathbf{w}}(\mathbf{x}))$ if $y = 0$
- If $y = 1$
  - if $h_{\mathbf{w}}(\mathbf{x}) = 1$, $Cost = 0$
  - If $h_{\mathbf{w}}(\mathbf{x}) \to 0$, $Cost \to \infty$
  - Captures intuition that if $h_{\mathbf{w}}(\mathbf{x}) = 0$, but $y = 1$, we will penalize the learning algorithm by a very large cost.





Cost Function

Cost Function for Logistic Regression:

$$
\begin{aligned}
E(\mathbf{w}) &= \sum_{i=1}^{N} Cost(h_{\mathbf{w}}(\mathbf{x}^{(i)}), y^{(i)}) \\
&= -\left[\sum_{i=1}^{N} y^{(i)} \log(h_{\mathbf{w}}(\mathbf{x}^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\mathbf{w}}(\mathbf{x}^{(i)}))\right]
\end{aligned}
$$

To fit parameters $\mathbf{w}$:

$$\min_{\mathbf{w}} E(\mathbf{w})$$

To make a prediction given a new $\mathbf{x}$: Output

$$h_{\mathbf{w}}(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$

Gradient Descent

Cost Function for Logistic Regression:

$$E(\mathbf{w}) = -\left[\sum_{i=1}^{N} y^{(i)} \log(h_{\mathbf{w}}(\mathbf{x}^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\mathbf{w}}(\mathbf{x}^{(i)}))\right]$$

Want:

$$\min_{\mathbf{w}} E(\mathbf{w})$$

Repeat until convergence $\{$

$$w_j := w_j - \alpha \frac{\partial E(\mathbf{w})}{\partial w_j}$$

$\}$ (simultaneously update all w_j).

Gradient Descent

Cost Function for Logistic Regression:

$$E(\mathbf{w}) = -\left[\sum_{i=1}^{N} y^{(i)} \log(h_{\mathbf{w}}(\mathbf{x}^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\mathbf{w}}(\mathbf{x}^{(i)}))\right]$$

Want:

$$\min_{\mathbf{w}} E(\mathbf{w})$$

Repeat until convergence {

$$w_j := w_j - \alpha \sum_{i=1}^{N} (h_{\mathbf{w}}(\mathbf{x}^{(i)}) - y^{(i)}) x_j^{(i)}$$

} (simultaneously update all w_j).

The algorithm looks the same as for linear regression! Is it?

Gradient Descent

Cost Function for Logistic Regression:

$$E(\mathbf{w}) = -\left[\sum_{i=1}^{N} y^{(i)} \log(h_{\mathbf{w}}(\mathbf{x}^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\mathbf{w}}(\mathbf{x}^{(i)}))\right]$$

Want:

$$\min_{\mathbf{w}} E(\mathbf{w})$$

Repeat until convergence $\{$

$$w_j := w_j - \alpha \sum_{i=1}^{N} (h_{\mathbf{w}}(\mathbf{x}^{(i)}) - y^{(i)}) x_j^{(i)}$$

$\}$ (simultaneously update all w_j).

The algorithm looks the same as for linear regression! Is it? No.

Logistic Regression as a Neural Network