

Introduction to Machine Learning

Logistic Regression

Varun Chandola

Computer Science & Engineering
State University of New York at Buffalo
Buffalo, NY, USA
chandola@buffalo.edu



University at Buffalo
Department of Computer Science
and Engineering
School of Engineering and Applied Sciences



Classification

Linear Classifiers

- Linear Classification via Hyperplanes

Logistic Regression

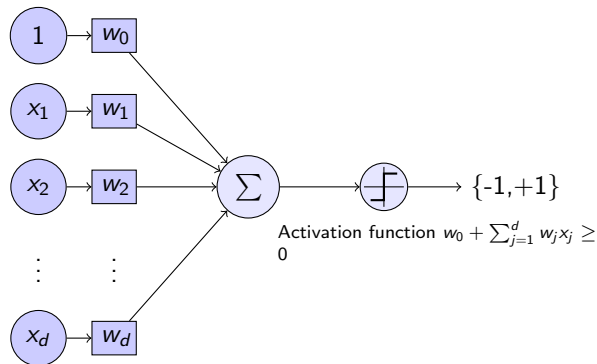
- Using Gradient Descent for Learning Weights

- Using Newton's Method

Supervised Learning - Classification

- ▶ Target y is categorical
- ▶ e.g., $y \in \{-1, +1\}$ (binary classification)
- ▶ A possible problem formulation: Learn f such that $y = f(\mathbf{x})$

Linear Classifiers

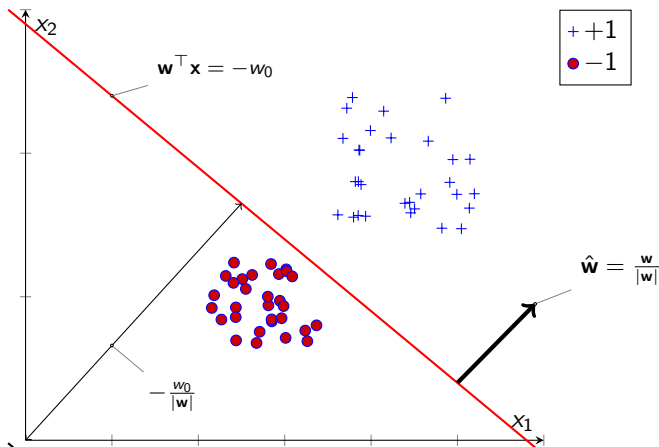


inputs weights

Decision Rule

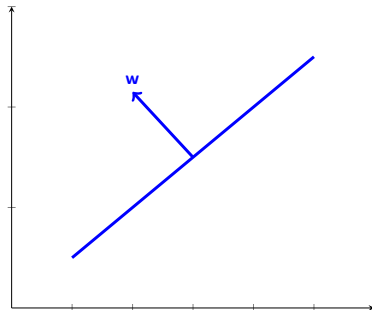
$$y_i = \begin{cases} -1 & \text{if } w_0 + \mathbf{w}^\top \mathbf{x}_i < 0 \\ +1 & \text{if } w_0 + \mathbf{w}^\top \mathbf{x}_i \geq 0 \end{cases}$$

Geometric Interpretation



Linear Hyperplane

- ▶ Separates a D -dimensional space into two half-spaces
- ▶ Defined by $\mathbf{w} \in \mathbb{R}^D$
 - ▶ *Orthogonal* to the hyperplane
 - ▶ This \mathbf{w} goes through the origin
 - ▶ How do you check if a point lies “above” or “below” \mathbf{w} ?
 - ▶ What happens for points **on** \mathbf{w} ?



Make hyperplane not go through origin

- ▶ Add a bias w_0
 - ▶ $w_0 > 0$ - move along \mathbf{w}
 - ▶ $w_0 < 0$ - move opposite to \mathbf{w}
- ▶ How to check if point lies above or below \mathbf{w} ?
 - ▶ If $\mathbf{w}^\top \mathbf{x} + w_0 > 0$ then \mathbf{x} is *above*
 - ▶ Else, *below*

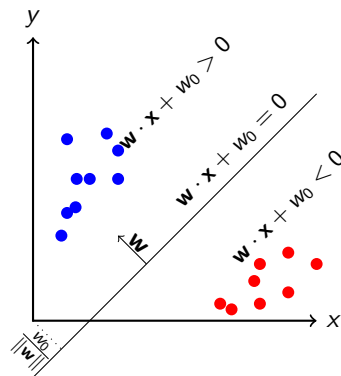
Line as a Decision Surface

- ▶ Decision boundary represented by the hyperplane \mathbf{w}
- ▶ For binary classification, \mathbf{w} points **towards** the positive class

Decision Rule

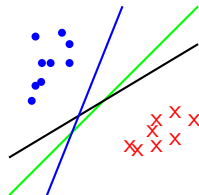
$$y = \text{sign}(\mathbf{w}^T \mathbf{x} + w_0)$$

- ▶ $\mathbf{w}^T \mathbf{x} + w_0 \geq 0 \Rightarrow y = +1$
- ▶ $\mathbf{w}^T \mathbf{x} + w_0 < 0 \Rightarrow y = -1$



What is Best Hyperplane Separator

- ▶ Find a hyperplane that separates the data
 - ▶ ... if the data is linearly separable
- ▶ But there can be many choices!
- ▶ Find the one with lowest error



- ▶ What is an appropriate loss function?

0-1 Loss

- ▶ Number of mistakes in training data

$$J(\mathbf{w}) = \min_{\mathbf{w}, w_0} \sum_{i=1}^n \mathbb{I}(y_i(\mathbf{w}^\top \mathbf{x}_i + w_0) < 0)$$

- ▶ Hard to optimize
- ▶ Solution - replace it with a mathematically manageable loss

Different Loss Functions

Note

From now on, assuming that intercept and constant terms are included in \mathbf{w} and \mathbf{x}_i , respectively.

► Squared Loss - Perceptron

$$J(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N (y_i - \mathbf{w}^\top \mathbf{x}_i)^2 \quad (1)$$

► Logistic Loss - Logistic Regression

$$J(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^n \log(1 + \exp(-y_i \mathbf{w}^\top \mathbf{x}_i)) \quad (2)$$

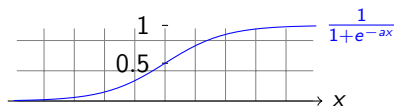
► Hinge Loss - Support Vector Machine

$$J(\mathbf{w}) = \sum_{i=1}^n \max(0, 1 - y_i \mathbf{w}^\top \mathbf{x}_i) \quad (3)$$

Logistic Regression

Geometric Interpretation

- ▶ Use regression to predict discrete values
- ▶ *Squash* output to $[0, 1]$ using sigmoid function
- ▶ Output less than 0.5 is one class and greater than 0.5 is the other



Probabilistic Interpretation

- ▶ Probability of x to belong to class +1

Logistic Loss Function

- ▶ For one training observation,
 - ▶ if $y_i = +1$, the probability of the predicted value to be $+1$

$$p_i = \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x}_i)}$$

- ▶ if $y_i = -1$, the probability of the predicted value to be -1

$$p_i = 1 - \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x}_i)} = \frac{1}{1 + \exp(\mathbf{w}^\top \mathbf{x}_i)}$$

- ▶ In general

$$p_i = \frac{1}{1 + \exp(-y_i \mathbf{w}^\top \mathbf{x}_i)}$$

- ▶ For logistic regression, the objective is to minimize the negative of the log probability:

$$J(\mathbf{w}) = - \sum_{i=1}^n \log(p_i) = \sum_{i=1}^n \log(1 + \exp(-y_i \mathbf{w}^\top \mathbf{x}_i))$$

Learning Logistic Regression Model

- ▶ Direct minimization??
 - ▶ No closed form solution for minimizing error
- ▶ Gradient Descent
- ▶ Newton's Method

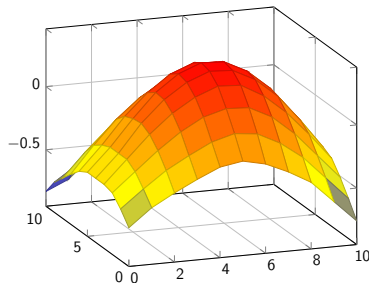
Using Gradient Descent for Learning Weights

- ▶ Compute gradient of $J(\mathbf{w})$ with respect to \mathbf{w}
- ▶ A convex function of \mathbf{w} with a unique global minima

$$\nabla J(\mathbf{w}) = -\frac{1}{n} \sum_{i=1}^n \frac{y_i}{1 + \exp(y_i \mathbf{w}^\top \mathbf{x}_i)} \mathbf{x}_i$$

- ▶ Update rule:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta \frac{d}{d\mathbf{w}_k} LL(\mathbf{w}_k)$$



Using Newton's Method

- ▶ Setting η is sometimes *tricky*
- ▶ Too large – incorrect results
- ▶ Too small – slow convergence
- ▶ Another way to speed up convergence:

Newton's Method

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta \mathbf{H}_k^{-1} \nabla J(\mathbf{w}_k)$$

Hessian

$$\mathbf{H}(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^n \frac{\exp(y_i \mathbf{w}^\top \mathbf{x}_i)}{(1 + \exp(y_i \mathbf{w}^\top \mathbf{x}_i))^2} \mathbf{x}_i \mathbf{x}_i^\top$$

References