Introduction to Machine Learning

Linear Classifiers - Perceptrons and Logistic Regression

Varun Chandola

February 11, 2020

Outline

Contents

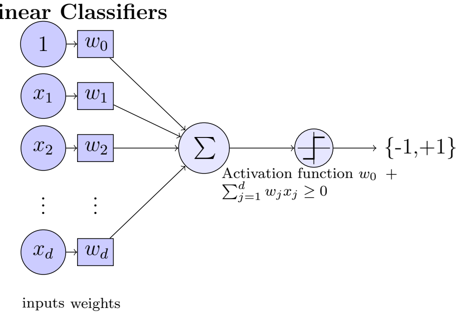
1 Classification

Supervised Learning - Classification

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

2 Linear Classifiers

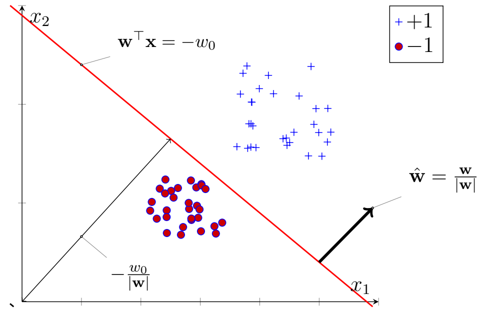
Linear Classifiers



Decision Rule

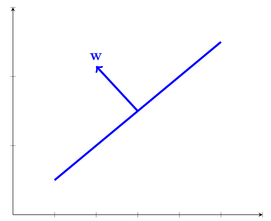
$$y_i = \begin{cases} -1 & \text{if } w_0 + \mathbf{w}^\top \mathbf{x}_i < 0 \\ +1 & \text{if } w_0 + \mathbf{w}^\top \mathbf{x}_i \geq 0 \end{cases}$$

Geometric Interpretation



2.1 Linear Classification via Hyperplanes

- Separates a D-dimensional space into two half-spaces
- Defined by $\mathbf{w} \in \Re^D$



- *Orthogonal* to the hyperplane
- This **w** goes through the origin
- How do you check if a point lies "above" or "below" **w**?
- What happens for points **on w**?

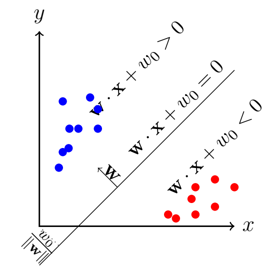
For a hyperplane that passes through the origin, a point **x** will lie above the hyperplane if $\mathbf{w}^\top\mathbf{x} > 0$ and will lie below the plane if $\mathbf{w}^\top\mathbf{x} < 0$, otherwise. This can be further understood by understanding that $bfw^\top\mathbf{x}$ is essentially equal to $|\mathbf{w}||\mathbf{x}|\cos\theta$, where θ is the angle between **w** and **x**.

- Add a bias w_0
  - $w_0 > 0$ - move along **w**
  - $w_0 < 0$ - move opposite to **w**
- How to check if point lies above or below **w**?
  - If $\mathbf{w}^\top\mathbf{x} + w_0 > 0$ then **x** is *above*
  - Else, *below*
- Decision boundary represented by the hyperplane **w**
- For binary classification, **w** points **towards** the positive class

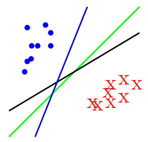
Decision Rule

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

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



y
x
$\mathbf{w}\cdot\mathbf{x} + w_0 > 0$
$\mathbf{w}\cdot\mathbf{x} + w_0 = 0$
$\mathbf{w}\cdot\mathbf{x} + w_0 < 0$
$\mathbf{w}$
$\frac{w_0}{\|\mathbf{w}\|}$



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

- 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

Learning w

- 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

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

- **Logistic Loss** - Logistic Regression

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

- **Hinge Loss** - Support Vector Machine

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

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 $\mathbf{x}$ to belong to class $+1$

Logistic Loss Function

- For one training observation,
  - if $y_i = +1$, the probability of the predicted value to be $+1$
    $$p_i = \frac{1}{1+\exp\left(-\mathbf{w}^\top \mathbf{x}_i\right)}$$
  - if $y_i = -1$, the probability of the predicted value to be -1
    $$p_i = 1 - \frac{1}{1+\exp\left(-\mathbf{w}^\top \mathbf{x}_i\right)} = \frac{1}{1+\exp\left(\mathbf{w}^\top \mathbf{x}_i\right)}$$
  - In general
    $$p_i = \frac{1}{1+\exp\left(-y_i\mathbf{w}^\top \mathbf{x}_i\right)}$$
- For logistic regression, the objective is to minimize the negative of the log probability:

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

Learning Logistic Regression Model

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

To understand why there is no closed form solution for maximizing the log-likelihood, we first differentiate $J(\mathbf{w})$ with respect to $\mathbf{w}$.

$$\begin{aligned}\nabla J(\mathbf{w}) &= \\ \frac{d}{d\mathbf{w}}J(\mathbf{w}) &= \sum_{i=1}^{n} log(1+\exp\left(-y_i\mathbf{w}^\top \mathbf{x}_i\right)) \\ &= -\frac{1}{n}\sum_{i=1}^{n}\frac{y_i}{1+\exp\left(y_i\mathbf{w}^\top \mathbf{x}_i\right)}\mathbf{x}_i\end{aligned}$$

Obviously, given that $\nabla J(\mathbf{w})$ is a non-linear function of $\mathbf{w}$, a closed form solution is not possible.

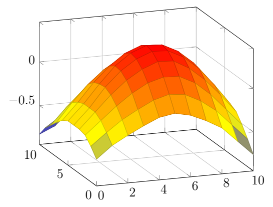
3.1 Using Gradient Descent for Learning Weights

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

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

- Update rule:

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



3.2 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

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

References