Building Linear Models from Scratch

Dr. Fayyaz Minhas

Department of Computer Science
University of Warwick
https://warwick.ac.uk/fac/sci/dcs/teaching/material/cs909/
Building Linear Discriminants

• Representation
  – Features
  – Linear Function
    \[ f(x; w) = w_1 x^{(1)} + w_2 x^{(2)} + \cdots + w_d x^{(d)} + b = 0 \]

• Evaluation
  – Misclassification

• Optimization
  – Find a line that minimizes misclassifications
  – How done: Visual reckoning / Constraint Satisfaction

• Why Study Linear Models?
A more mathematical look

- Linear Discriminants
- The linear discriminant function is given by

\[
 f(x; w) = w_1 x^{(1)} + w_2 x^{(2)} + ... + w_d x^{(d)} + b = w^T x + b
\]

\[
 f(x'; w') = w_1 x'^{(1)} + w_2 x'^{(2)} + ... + w_d x'^{(d)} + b = w'^T x'
\]

\[
 w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix}, \quad x = \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ \vdots \\ x^{(d)} \end{bmatrix}
\]

\[
 w' = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \\ b \end{bmatrix}, \quad x' = \begin{bmatrix} x'^{(1)} \\ x'^{(2)} \\ \vdots \\ x'^{(d)} \\ 1 \end{bmatrix}
\]

\[
 f(x) = w^T x + b = 0
\]

w & b are ‘learned’ from the training data using some error criterion.
Classification Loss Function

• A misclassification is an error
  – If a training example has a label of $y = +1$, then its discriminant function score $f(x)$ should be _____
  
  – If a training example has a label of $y = -1$, then its discriminant function score $f(x)$ should be _____
  
  – Thus, we have an error whenever: __________
Classification Loss Function

• A misclassification is an error
  – If a training example has a label of $y = +1$, then its discriminant function score $f(x)$ should be $> 0$

  – If a training example has a label of $y = -1$, then its discriminant function score $f(x)$ should be $< 0$

  – Thus, we have an error whenever: $yf(x) < 0$
0-1 Loss/Error

• Consider a single example:

  – Our error function is: 
    $$l(f(x), y) = \begin{cases} 
    0 & yf(x) > 0 \\
    1 & yf(x) \leq 0 
    \end{cases}$$

Formally called the zero-one loss function
0-1 Misclassification Error

• We want to find the parameters of the discriminant that minimize the loss for all examples in training

• Issues with 0-1 loss
  – Non Differentiable
  – Leads to poor optimization

• We need a “surrogate” or approximation of the loss
  – Should be continuous
  – Should be an over-approximation of the 0-1 loss
    • Generates at least as much error as the 0-1 loss would
  – Should be convex
    • Convex loss function leads to convex optimization problems which are easier to solve as they have a single minima
Surrogate Classification Loss

\[ l(f(x), y) = \begin{cases} 
  0 & \text{if } yf(x) > 1 \\
  1 - yf(x) & \text{if } yf(x) \leq 1
\end{cases} = \begin{cases} 
  0 & 1 - yf(x) < 0 \\
  1 - yf(x) & 1 - yf(x) \geq 0
\end{cases} \]

OR

\[ l(f(x), y) = \max(0, 1 - yf(x)) \]

- Hinge Loss Function
  - A convex over-approximation of the 0-1 loss
Optimization

\[
\min_w L(X, Y; w) = \sum_{i=1}^{N} \max\{0, 1 - y_i f(x_i; w)\}
\]

• How can we solve it?
  – Take the derivative and substitute to zero
  – How else can we solve it?
    • Use gradient descent
Optimization

\[ \min_w L(X, Y; w) = \sum_{i=1}^{N} l(f(x_i; w), y_i) = \sum_{i=1}^{N} \max\{0, 1 - y_i f(x_i; w)\} \]

\[ \frac{\partial L}{\partial w} = \sum_{i=1}^{N} \frac{\partial l(f(x_i; w), y_i)}{\partial w} \]

\[ \frac{\partial}{\partial w} \max\{0, 1 - y(w^T x)\} = \begin{cases} 0 & 1 - yf(x; w) < 0 \\ -yx & \text{else} \end{cases} = \begin{cases} -yx & l(f(x; w), y) > 0 \\ 0 & \text{else} \end{cases} \]
Algorithm

• Given:
  – Training Examples: \(\{(x_i, y_i) | i = 1 \ldots N\}, y_i \in \{-1, +1\}\)
• Initialize \(w^{(0)}\) at random
• Until Convergence
  – For \(i = 1 \ldots N\)
    • Pick example \(x_i\) with label \(y_i\)
    • Compute \(f(x_i) = w^{(k)^T}x + b\)
    • If \(y_i f(x_i) < 1\) then update your weight vector using gradient descent

\[
\begin{align*}
w^{(k)} &= w^{(k-1)} - \alpha \nabla l(w^{(k-1)}) = w^{(k-1)} - \alpha (-y_i x_i) = w^{(k-1)} + \alpha y_i x_i
\end{align*}
\]
REO For Perceptron

- **Representation**
  - Features
  - Discriminant
    - Linear
- **Evaluation**
  - 0/1 (Step) Loss
  - Hinge Loss
- **Optimization**
  - Using Gradient Descent

- **Given:**
  - Training Examples: \( \{(x_i, y_i)| i = 1 \ldots N\}, y_i \in \{-1, +1\} \)
  - Initialize \( w^{(0)} \) at random
  - Until Convergence
    - For \( i = 1 \ldots N \)
      - Pick example \( x_i \) with label \( y_i \)
      - Compute \( f(x_i) = w^{(k)}^T x + b \)
      - If \( y_i f(x_i) < 1 \) then update your weight vector using gradient descent

\[
  w^{(k)} = w^{(k-1)} - \alpha \nabla I(w^{(k-1)}) = w^{(k-1)} - \alpha (-y_i x_i) = w^{(k-1)} + \alpha y_i x_i
\]
Perceptron

- A simpler version of this algorithm is called: Perceptron
  - It updated weights whenever an example was misclassified \( y_if(x_i) < 0 \) instead of when \( y_if(x_i) < 1 \)
  - Rosenblatt (1962)
  - Minsky and Papert (1969, 1988)
  - This algorithm provides theoretical guarantees of convergence to a correct separating boundary
    - If the data is linearly separable and you allow the perceptron algorithm to run long enough, you will find the separating line!
    - **Perceptron Learning Rule Convergence Theorem**
Perceptron

• One of the first “artificial” neural networks

\[ f(x; \theta) = w^T x + b \]
import numpy as np
import matplotlib.pyplot as plt
import itertools

class Perceptron:
    def __init__(self, alpha=0.1, epochs=200):
        self.alpha = alpha
        self.epochs = epochs
        self.W = np.array([0])
        self.bias = np.random.randn()
        self.Lambda = 0.5
    def fit(self, X_tr, Y_tr):
        d = X_tr.shape[1]
        self.W = np.random.randn(d)
        for e in range(self.epochs):
            finished = True
            for i, x in enumerate(X_tr):
                if Y_tr[i] != self.predict(np.atleast_2d(x)):
                    finished = False
                    self.W += self.alpha * Y_tr[i] * x
                    self.bias += self.alpha * Y_tr[i]
            if finished: break
    def score(self, x):
        return np.dot(x, self.W) + self.bias
    def predict(self, x):
        return np.sign(self.score(x))

if __name__ == '__main__':
    from plotit import plotit
    X_tr = np.array([[-1, 0], [0, 1], [4, 4], [2, 3]])
    y_tr = np.array([-1, -1, 1, 1])
    clf = Perceptron()
    clf.fit(X_tr, y_tr)
    z = clf.score(X_tr)
    print("Prediction Scores:", z)
    y = clf.predict(X_tr)
    print("Prediction Labels:", y)
    plotit(X_tr, y_tr, clf=clf.score, conts=[0],
           extent=[-5, 5, -5, 5])
End of Lecture

We want to make a machine that will be proud of us.

- Danny Hillis