1. Classifiers

July 8, 2020

0.1 Basic terms

n observations, each with d features/predictors. Some observations belong to class C; some do not.

decision function

$f(x) > 0$, if $x \in classC$

$f(x) \leq 0$. if $x \notin classC$

aka: predictor function or discriminant function

decision boundary

the boundary chosen by classifier to separate items in the class from those not.

$\{x \in \mathbb{R}^d : f(x) = 0\}$

This set is a $(d-1)$-dimentional surface in $\mathbb{R}^d$. Hence it is also called an isosurface of f for the isovalue 0.

0.2 Linear Classifier

The decision boundary is a line/plane.

Decision function

$f(x) = w \cdot x + \alpha$

Decision boundary

$H = \{x : w \cdot x = -\alpha\}$

H is called a hyperplane (A line in 2D, a plane in 3D)

Theorem

Let x,y be 2 points that lie on H. Then $w \cdot (y - x) = 0$

w is called the *normal vector* of H, since w is normal (perpendicular) to H

signed distance from x to H

$\frac{1}{|w|}(w \cdot x + \alpha)$

i.e. positive on one side of H; negative on other side.

Moreover, the distance from H to the origin is α. Hence $\alpha = 0$ if and only if H passes through origin.

Definition

The input data is **linearly separable** if there exists a hyperplane that separates all the sample points in class C from all the points NOT in class C.

0.2.1 Centroid Method

Compute mean μ_C of all vectors in class C and mean μ_X of all vectors NOT in C.

decision function

$f(x) = (\mu_C - \mu_X) \cdot x - (\mu_C - \mu_X)\frac{\mu_C + \mu_X}{2}$
decision boundary
the hyperplane that perpendicular bisects line segment w/endpoints μ_C,μ_X

0.2.2 Perceptron Algorithm

Consider n sample points $X_1, X_2, ..., X_n$.

For each sample point, the label $y_i = \begin{cases} 1 & \text{if } X_i \in C \\ -1 & \text{if } X_i \notin C \end{cases}$
For simplicity, consider only decision boundaries that pass through the origin, that is, $\alpha = 0$
If it doesn't pass through origin, add a fictious dimension:

$$f(x) = w' \cdot x' = \begin{bmatrix} w_1 & w_2 & \cdots & w_d & \alpha \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \\ 1 \end{bmatrix}$$

Now we have sample points in $\mathbb{R}^{d+1}$, all lying on hyperplane $x_{d+1} = 1$.
Goal
Find weights w such that
$X_i \cdot w \geq 0$, if $y_i = 1$, and
$X_i \cdot w \leq 0$, if $y_i = -1$.
Equivalently: $y_i X_i \cdot w \geq 0$. This is called a *constraint*.
Idea
Define a *risk function* R, positive if some constrains are violated. Then we use optimization to choose w that minimizes R.
Loss function
$L(z, y_i) = \begin{cases} 0 & \text{if } y_i z \geq 0 \\ -y_i z & \text{otherwise} \end{cases}$
Risk function
$R(w) = \Sigma_{i=1}^{n} L(X_i \cdot w, y_i) = \Sigma_{i \in V} - y_i X_i \cdot w$
Where V is the set of indices i for which $y_i X_i \cdot w < 0$
if w classifies all $X_1, ..., X_n$ correctly, then $R(w) = 0$.
Otherwise, $R(w)$ is positive, and we want to find a better value of w.
Modified goal
Find w that minimizes $R(w)$
Optimization algorithm (not perceptron yet)
Given a starting point w (good choice: any $y_i X_i$)
While $R(w) > 0$: $w \leftarrow w + \epsilon \cdot (-\nabla R(w))$
return w

- $\nabla R(w)$, is the direction of the steepest ascent. Hence take a step in the opposite direction,

$$-\nabla R(w) = \Sigma_{i \in V} y_i X_i$$

- $\epsilon > 0$ is the *step size* aka *learning rate*
- Problem: Slow! Each step takes $O(nd)$ times

Optimization algorithm 2 aka **Perceptron Algorithm**
Given a starting point w
While some $y_i X_i \cdot w < 0$: $w \leftarrow w + \epsilon \cdot y_i X_i$
return w

- each step, only pick one misclassified X_i, do gradient descent on loss function $L(X_i \cdot w, y_i)$
- each step takes $O(d)$ times. (Not counting the time to search for a misclassified X_i)

0.2.3 Maximum Margin Classifiers (hard margin SVM)

The **margin** of a linear classifier is the distance from the decision boundary to the nearest sample point.

Goal
Make the margin as wide as possible
$margin = min_i \frac{1}{|w|}|w \cdot X_i + \alpha|$
Constraints
$y_i(w \cdot X_i + \alpha) \geq 1$ for $i \in [1, n]$
Notice that the right-hand side is a 1, rather than 0 as it was for the perceptron algorithm.
Intuition: not allow points falling on the boundary, that is, margin is at least larger than 0.
Optimization problem
Find w and α that maximize margin $max_{w,\alpha} min_i \frac{1}{|w|}|w \cdot X_i + \alpha| = max_{w,\alpha} \frac{1}{|w|} = min_{w,\alpha}|w|^2$
Subject to $y_i(X_i \cdot w + \alpha) \geq 1$ for all $i \in [1, n]$

- It is a quadratic program in $d + 1$ dimensions and n contraints.
- It has one unique solution / no solution.
- Use $|w|^2$ instead of $|w|$ because $|w|$ is not smooth at zero, while $|w|^2$ is smooth everywhere.
- Only applied to those linear seperatable
- Sensitive to outliers.

0.2.4 Soft-Margin Support Vector Machines (SVMs)

Allow some points to violate the margin, with slack varibles
Constrains
$y_i(X_i \cdot w + \alpha) >= 1 - \xi_i$ and
$\xi_i \geq 0$ for $i \in [1, n]$

- sample points that don't violate the margin all have $\xi_i = 0$
- Point i has nonzero ξ_i if and only if it violates the margin.
- $\frac{\xi_i}{|w|}$ is how much the point i violets the margin.

Optimization problem
Find w, α and ξ_i that minimize $|w|^2 + C\Sigma_{i=1}^n \xi_i$
Subject to $y_i(X_i \cdot w + \alpha) >= 1 - \xi_i$
$\xi_i \geq 0$ for $i \in [1, n]$

- It is a quadratic program in $d + n + 1$ dimensions and $2n$ constrains.
- $C > 0$ is a scalar regularization hyperparameter. Larger C increase the penalty of $\xi_i \neq 0$
- If C is infinite, we're back to a hard-margin SVM.