Classification, Halfspaces, the Perceptron algorithm

Chris Cornwell

Feb 25, 2025

Outline

Classification tasks

Half-space model

Perceptron algorithm

Outline

Classification tasks

Half-space model

Perceptron algorithm

Example of Classification

Use some model to determine a digit that was (hand)written in an image

`0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, or `9`.

Example of Classification

Use some model to determine a digit that was (hand)written in an image

$$\texttt{0, 1, 2, 3, 4, 5, 6, 7, 8,} \text{ or } 9.$$

- Convert image to a vector (*in some way*) $\longrightarrow \mathbf{x}$.
- Your model's output: $\hat{y}(\mathbf{x})$ is the (predicted) digit.

Provided with your data, an "observation" $y \in \{0, 1, \ldots, 9\}$ of the digit being written.

Example of Classification

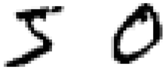
Use some model to determine a digit that was (hand)written in an image

$$\texttt{0, 1, 2, 3, 4, 5, 6, 7, 8,} \text{ or } 9.$$

- Convert image to a vector (*in some way*) $\longrightarrow$ $\mathbf{x}$.
- Your model's output: $\hat{y}(\mathbf{x})$ is the (predicted) digit.

Provided with your data, an "observation" $y \in \{0, 1, \ldots, 9\}$ of the digit being written.

y and $\hat{y}$ are numbers on number line; but, use them like labels (or, separate buckets) to group points $\mathbf{x}$. When $y = 5$, predicting $\hat{y} = 4$ is not any better than $\hat{y} = 0$.



Close only counts in ~~horseshoes~~ ...Regression

In linear regression, on indpt. variables $x_0, x_1, \ldots, x_{d-1}$, had (affine) linear function $\hat{y} = p_0 x_0 + p_1 x_1 + \ldots + p_{d-1} x_{d-1} + p_d$; values of function $\leftrightarrow$ prediction $\hat{y}$; error term ε, so that $y = \hat{y} + \varepsilon$.

[1] Should consider the output y here to be a random variable, with distribution that depends on $\mathbf{x}$. In simple linear regression, ε is a random variable and $y = p_0 x_0 + p_1 + \varepsilon$.

Close only counts in ~~horseshoes~~ ...Regression

In linear regression, on indpt. variables $x_0, x_1, \ldots, x_{d-1}$, had (affine) linear function $\hat{y} = p_0 x_0 + p_1 x_1 + \ldots + p_{d-1} x_{d-1} + p_d$;
values of function $\leftrightarrow$ prediction $\hat{y}$; error term ε, so that $y = \hat{y} + \varepsilon$.
An observation y for each data point $\mathbf{x} = (x_0, x_1, \ldots, x_{d-1}) \in \mathbb{R}^d$. The **linear model** $\mathbf{x} \mapsto \hat{y}$ approximates $\mathbf{x} \mapsto y$.[1]

[1] Should consider the output y here to be a random variable, with distribution that depends on $\mathbf{x}$. In simple linear regression, ε is a random variable and $y = p_0 x_0 + p_1 + \varepsilon$.

Close only counts in ~~horseshoes~~ ...Regression

In linear regression, on indpt. variables $x_0, x_1, \ldots, x_{d-1}$, had (affine) linear function $\hat{y} = p_0 x_0 + p_1 x_1 + \ldots + p_{d-1} x_{d-1} + p_d$;
values of function $\leftrightarrow$ prediction $\hat{y}$; error term ε, so that $y = \hat{y} + \varepsilon$.
An observation y for each data point $\mathbf{x} = (x_0, x_1, \ldots, x_{d-1}) \in \mathbb{R}^d$. The **linear model** $\mathbf{x} \mapsto \hat{y}$ approximates $\mathbf{x} \mapsto y$.[1]
We expect that $|y - \hat{y}|$ is almost never exactly 0; a good model: one where $|y - \hat{y}|$ is small on average (but, still positive).

"Regression"

[1] Should consider the output y here to be a random variable, with distribution that depends on $\mathbf{x}$. In simple linear regression, ε is a random variable and $y = p_0 x_0 + p_1 + \varepsilon$.

Close only counts in ~~horseshoes~~ ...Regression

In linear regression, on indpt. variables $x_0, x_1, \ldots, x_{d-1}$, had (affine) linear function $\hat{y} = p_0 x_0 + p_1 x_1 + \ldots + p_{d-1} x_{d-1} + p_d$;
values of function $\leftrightarrow$ prediction $\hat{y}$; error term ε, so that $y = \hat{y} + \varepsilon$.
An observation y for each data point $\mathbf{x} = (x_0, x_1, \ldots, x_{d-1}) \in \mathbb{R}^d$. The **linear model** $\mathbf{x} \mapsto \hat{y}$ approximates $\mathbf{x} \mapsto y$.[1]
We expect that $|y - \hat{y}|$ is <u>almost never</u> exactly 0; a good model: one where $|y - \hat{y}|$ is small on average (but, still positive).

“<u>Regression</u>”

“Classification” tasks: the value y is a <u>label</u> and might not even be a number. The prediction $\hat{y}$ is simply wrong, or not; close doesn’t count. Good model: when $\hat{y} = y$ as often as possible.

---

[1]Should consider the output y here to be a random variable, with distribution that depends on $\mathbf{x}$. In simple linear regression, ε is a random variable and $y = p_0 x_0 + p_1 + \varepsilon$.

Outline

Classification tasks

Half-space model

Perceptron algorithm

A linear model for classification

Assume data is from $\mathbb{R}^d$ for some $d > 0$ and we only have two labels (e.g., this is Spam ($\mathtt{S}$) or it is Not spam ($\mathtt{N}$)).

A linear model for classification

Assume data is from $\mathbb{R}^d$ for some $d > 0$ and we only have two labels (e.g., this is Spam ($\mathtt{S}$) or it is Not spam ($\mathtt{N}$)).

A hyperplane in $\mathbb{R}^d$ is an (affine) linear subspace that separates $\mathbb{R}^d$ in two. Perhaps we get lucky and can find a hyperplane H so that data points with label $\mathtt{S}$ are on one side of H and data with label $\mathtt{N}$ are on the other side.

A linear model for classification

Assume data is from $\mathbb{R}^d$ for some $d > 0$ and we only have two labels (e.g., this is Spam ($\mathtt{S}$) or it is Not spam ($\mathtt{N}$)).

A hyperplane in $\mathbb{R}^d$ is an (affine) linear subspace that separates $\mathbb{R}^d$ in two. Perhaps we get lucky and can find a hyperplane H so that data points with label $\mathtt{S}$ are on one side of H and data with label $\mathtt{N}$ are on the other side. Using coordinates $(x_1, x_2, \ldots, x_d)$ in $\mathbb{R}^d$, a hyperplane H may be determined from $d + 1$ numbers $w_1, w_2, \ldots, w_d$, and b. It consists of solutions to

$$w_1x_1 + w_2x_2 \ldots + w_dx_d + b = 0.$$

- Rewriting in vector form: $\mathbf{w} = (w_1, w_2, \ldots, w_d)$, look for solutions $\mathbf{x} \in \mathbb{R}^d$ to the equation $\mathbf{w} \cdot \mathbf{x} + b = 0$.
- $\mathbf{w}$ is a vector that is orthogonal to a $(d-1)$-dimensional subspace of $\mathbb{R}^d$; $|b|$ corresponds to a translation away from the origin.

Half-space model, continued

Using the notation from last slide:
a half-space model in $\mathbb{R}^d$ is determined by $d+1$ parameters $w_1, w_2, \ldots, w_d, b$; the first d parameters grouped into a vector: $\mathbf{w} = (w_1, w_2, \ldots, w_d)$.

Half-space model, continued

Using the notation from last slide:
a half-space model in $\mathbb{R}^d$ is determined by $d+1$ parameters $w_1, w_2, \ldots, w_d, b$; the first d parameters grouped into a vector: $\mathbf{w} = (w_1, w_2, \ldots, w_d)$.
Given $\mathbf{x} \in \mathbb{R}^d$, the side of the hyperplane it is on is determined by the sign of $\mathbf{w} \cdot \mathbf{x} + b$.

- (Positive side) Say that $h(\mathbf{x}) = 1$ if $\mathbf{w} \cdot \mathbf{x} + b > 0$.
- (Negative side) Say that $h(\mathbf{x}) = -1$ if $\mathbf{x} \cdot \mathbf{x} + b < 0$.

Half-space model, continued

Using the notation from last slide:
a half-space model in $\mathbb{R}^d$ is determined by $d + 1$ parameters $w_1, w_2, \ldots, w_d, b$; the first d parameters grouped into a vector: $\mathbf{w} = (w_1, w_2, \ldots, w_d)$.
Given $\mathbf{x} \in \mathbb{R}^d$, the side of the hyperplane it is on is determined by the sign of $\mathbf{w} \cdot \mathbf{x} + b$.

- (Positive side) Say that $h(\mathbf{x}) = 1$ if $\mathbf{w} \cdot \mathbf{x} + b > 0$.
- (Negative side) Say that $h(\mathbf{x}) = -1$ if $\mathbf{x} \cdot \mathbf{x} + b < 0$.

If there exists a hyperplane, given by some $\mathbf{w}$, b, so that $\mathbf{x}$ has one of the labels if and only if it is on the positive side, the labeled data are called **linearly separable**.

Linearly separable

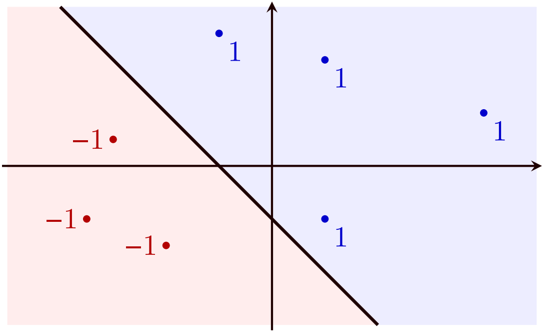


Figure: The hyperplane $H = \{(x, y) \in \mathbb{R}^2 : x + y + 1 = 0\}$, corresponding positive and negative regions, $\mathbf{w} = (1, 1)$, $b = 1$

Not linearly separable

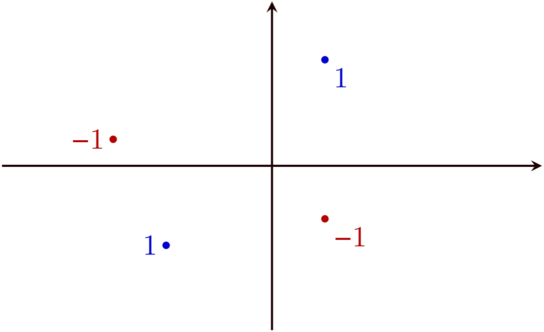


Figure: A data set in $\mathbb{R}^2$ that is not linearly separable.

- A criterion (checkable, in theory) that is equivalent to “not linearly separable”?

Outline

Classification tasks

Half-space model

Perceptron algorithm

Setup for Perceptron algorithm

Assuming that $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ is linearly separable, the Perceptron algorithm is a procedure that is guaranteed to find a hyperplane that separates the data.

Setup for Perceptron algorithm

Assuming that $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ is linearly separable, the Perceptron algorithm is a procedure that is guaranteed to find a hyperplane that separates the data.

To describe it: for each $\mathbf{x}_i$, use X_i to denote the $(d+1)$-vector consisting of $\mathbf{x}_i$ with 1 appended at the end;

Additionally, use W to denote the vector $\mathbf{w}$ with b appended at the end.

Setup for Perceptron algorithm

Assuming that $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ is linearly separable, the Perceptron algorithm is a procedure that is guaranteed to find a hyperplane that separates the data.

To describe it: for each $\mathbf{x}_i$, use X_i to denote the $(d+1)$-vector consisting of $\mathbf{x}_i$ with 1 appended at the end;

Additionally, use W to denote the vector $\mathbf{w}$ with b appended at the end.

Note that $W \cdot X_i = \mathbf{w} \cdot \mathbf{x}_i + b$.

For linearly separable data, our goal is to find $W \in \mathbb{R}^{d+1}$ so that, for all $1 \leq i \leq n$, $W \cdot X_i$ and y_i have the same sign (are both positive or both negative).

- Equivalently, we need $y_i W \cdot X_i > 0$ for all $1 \leq i \leq n$.

Perceptron algorithm

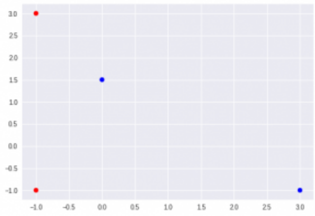
Suppose the data is linearly separable. Also, `x` is an $n \times d$ array of points, with ith row equal to $\mathtt{x}_i$, and `y` is array of the labels. The Perceptron algorithm finds `W` iteratively as follows.

```
input: x, y  ## x is n by d, y is 1d array
X ← append 1 to each row of x
W ← (0,0,...,0)  ## Initial W
while (exists i with y[i]*dot(W, X[i]) ≤ 0){
    W ← W + y[i]*X[i]
}
return W
```

Example

A simple example in $\mathbb{R}^2$, with $n = 4$ points.

$$\text{x: } \begin{bmatrix} -1 & 3 \\ -1 & -1 \\ 3 & -1 \\ 0 & 1.5 \end{bmatrix} \qquad\qquad \text{y: } \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \end{bmatrix}$$



Example, continued

A simple example in $\mathbb{R}^2$, with $n = 4$ points.

$$\text{x:} \begin{bmatrix} -1 & 3 \\ -1 & -1 \\ 3 & -1 \\ 0 & 1.5 \end{bmatrix} \qquad \text{y:} \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \end{bmatrix}$$

Begin with $W^{(1)} = (0, 0, 0)$. On next step:
$W^{(2)} = \vec{0} + y_1 X_1 = -1 * (-1, 3, 1) = (1, -3, -1)$.
Next: since $y_1 W^{(2)} \cdot X_1 > 0$, check
$y_2 W^{(2)} \cdot X_2 = -1 * (-1 + 3 - 1) = -1$. So,

$$W^{(3)} = W^{(2)} + y_2 X_2 = (2, -2, -2).$$

Continue in this way – on each step check dot products (in order) with X_1, X_2, X_3, X_4. Eventually you get,

$$W^{(10)} = (4, -0.5, 1)$$

and this vector is returned, meaning that
$H = \{(x_1, x_2) \in \mathbb{R}^2 : \; 4x_1 - 0.5x_2 + 1 = 0\}$ separates the points.

Perceptron algorithm, stopping time

Under our assumptions for Perceptron algorithm, a guarantee on eventually stopping.

Theorem

Define $R = \max_i |X_i|$ *and* $B = \min_i\{|V| : \forall i, y_i V \cdot X_i \geq 1\}$. *Then, the Perceptron algorithm stops after at most* $(RB)^2$ *iterations and, when it stops with output* W, *then* $y_i W \cdot X_i > 0$ *for all* $1 \leq i \leq n$.

Perceptron algorithm, stopping time

Under our assumptions for Perceptron algorithm, a guarantee on eventually stopping.

Theorem

Define $R = \max_i |X_i|$ and $B = \min_i\{|V| : \forall i, y_i V \cdot X_i \geq 1\}$. Then, the Perceptron algorithm stops after at most $(RB)^2$ iterations and, when it stops with output W, then $y_i W \cdot X_i > 0$ for all $1 \leq i \leq n$.

Idea of proof: Write W^* for vector that realizes the minimum B. Also, write $W^{(t)}$ for the vector W on the t^{th} step, with $W^{(1)} = (0, 0, \ldots, 0)$.

Perceptron algorithm, stopping time

Under our assumptions for Perceptron algorithm, a guarantee on eventually stopping.

Theorem

Define $R = \max_i |X_i|$ and $B = \min_i\{|V| : \forall i, y_i V \cdot X_i \geq 1\}$. Then, the Perceptron algorithm stops after at most $(RB)^2$ iterations and, when it stops with output W, then $y_i W \cdot X_i > 0$ for all $1 \leq i \leq n$.

Idea of proof: Write W^* for vector that realizes the minimum B. Also, write $W^{(t)}$ for the vector W on the t^{th} step, with $W^{(1)} = (0, 0, \ldots, 0)$. Using how $W^{(t+1)}$ is obtained from $W^{(t)}$, can show that $W^* \cdot W^{(T+1)} \geq T$ after $T + 1$ iterations. Also, using the condition on $W^{(T)}$ that necessitates an update, can show that $|W^{(T+1)}| \leq \sqrt{T}R$. (Both statements, use induction.)

Perceptron algorithm, stopping time

Under our assumptions for Perceptron algorithm, a guarantee on eventually stopping.

Theorem

Define $R = \max_i |X_i|$ and $B = \min_i\{|V| : \forall i, y_i V \cdot X_i \geq 1\}$. Then, the Perceptron algorithm stops after at most $(RB)^2$ iterations and, when it stops with output W, then $y_i W \cdot X_i > 0$ for all $1 \leq i \leq n$.

Idea of proof: Write W^* for vector that realizes the minimum B. Also, write $W^{(t)}$ for the vector W on the t^{th} step, with $W^{(1)} = (0, 0, \ldots, 0)$. Using how $W^{(t+1)}$ is obtained from $W^{(t)}$, can show that $W^* \cdot W^{(T+1)} \geq T$ after $T + 1$ iterations. Also, using the condition on $W^{(T)}$ that necessitates an update, can show that $|W^{(T+1)}| \leq \sqrt{T}R$. (Both statements, use induction.)
Now, by Cauchy-Schwarz inequality, $T \leq BR\sqrt{T}$, which we can rearrange to $T \leq (BR)^2$.