Logistic Regression

Chris Cornwell

Feb 27, 2025

Outline

Outline

Decision boundaries

For model h, made for classification task (with data points $\mathbf{x} \in \mathbb{R}^d$), write $C_y \subset \mathbb{R}^d$ for the set of points with label y, i.e.,

$$C_y = h^{-1}(y) = \{\mathbf{x} \in \mathbb{R}^d \mid h(\mathbf{x}) = y\}.$$

Decision boundaries

For model h, made for classification task (with data points $\mathbf{x} \in \mathbb{R}^d$), write $C_y \subset \mathbb{R}^d$ for the set of points with label y, i.e.,

$$C_y = h^{-1}(y) = \{\mathbf{x} \in \mathbb{R}^d \mid h(\mathbf{x}) = y\}.$$

Say $y \neq y'$ and a point is in the boundary of both C_y and $C_{y'}$. We say that point is on a **decision boundary** of the model. In a half-space model (last lecture), the hyperplane determined by $\mathbf{w}$ and b is the decision boundary.

Decision boundaries

For model h, made for classification task (with data points $\mathbf{x} \in \mathbb{R}^d$), write $C_y \subset \mathbb{R}^d$ for the set of points with label y, i.e.,

$$C_y = h^{-1}(y) = \{\mathbf{x} \in \mathbb{R}^d \mid h(\mathbf{x}) = y\}.$$

Say $y \neq y'$ and a point is in the boundary of both C_y and $C_{y'}$. We say that point is on a **decision boundary** of the model. In a half-space model (last lecture), the hyperplane determined by $\mathbf{w}$ and b is the decision boundary. The Perceptron algorithm might produce a model with many data points that are near the decision boundary.

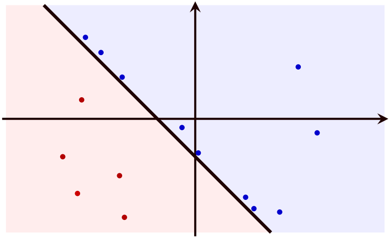


Figure: Many points near the decision boundary

But...data is messy

If many points close to the decision boundary, likely for newly observed data to appear on “wrong side” of decision boundary. Perceptron algorithm gives us nothing to correct for this.

But...data is messy

If many points close to the decision boundary, likely for newly observed data to appear on "wrong side" of decision boundary. Perceptron algorithm gives us nothing to correct for this.

If $\mathbf{x}$ is close to decision boundary, we might feel less confident in giving the label $h(\mathbf{x})$ that we do. And, if $\mathbf{x}$ is farther from the boundary, where only one label is seen nearby, more confidence is warranted.

But...data is messy

If many points close to the decision boundary, likely for newly observed data to appear on “wrong side” of decision boundary. Perceptron algorithm gives us nothing to correct for this.

If $\mathbf{x}$ is close to decision boundary, we might feel less confident in giving the label $h(\mathbf{x})$ that we do. And, if $\mathbf{x}$ is farther from the boundary, where only one label is seen nearby, more confidence is warranted.

Also...the immediate change of label across the boundary (a discontinuity in the model) ...perhaps not “natural”?

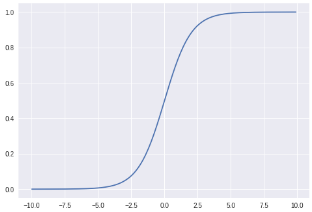
Outline

Incorporating a probability into half-space model

Instead of only capturing the sign of $\mathbf{w} \cdot \mathbf{x} + b$, compose it with the **logistic function**.

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

- $0 < \sigma(z) < 1$ for all $z \in \mathbb{R}$;
- $\lim_{z \to \infty} \sigma(z) = 1$ and $\lim_{z \to -\infty} \sigma(z) = 0$;
- $\sigma(0) = 1/2$.



Logistic model, continued

Define logistic model as follows.
So, for $\mathbf{x} \in \mathbb{R}^d$, find $\sigma(\mathbf{w} \cdot \mathbf{x} + b)$.

Logistic model, continued

Define logistic model as follows.
So, for $\mathbf{x} \in \mathbb{R}^d$, find $\sigma(\mathbf{w} \cdot \mathbf{x} + b)$.
Given $\mathbf{x} \in \mathbb{R}^d$, the side of the hyperplane H it is on is determined by the sign of $\mathbf{w} \cdot \mathbf{x} + b$. Our half-space model: $h : \mathbb{R}^d \setminus H \longrightarrow \{1, -1\}$.

Logistic model, continued

Define logistic model as follows.
So, for $\mathbf{x} \in \mathbb{R}^d$, find $\sigma(\mathbf{w} \cdot \mathbf{x} + b)$.
Given $\mathbf{x} \in \mathbb{R}^d$, the side of the hyperplane H it is on is determined by the sign of $\mathbf{w} \cdot \mathbf{x} + b$. Our half-space model: $h : \mathbb{R}^d \setminus H \longrightarrow \{1, -1\}$.

- (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 $\mathbf{x} \cdot \mathbf{x} + b < 0$.

Logistic model, continued

Define logistic model as follows.
So, for $\mathbf{x} \in \mathbb{R}^d$, find $\sigma(\mathbf{w} \cdot \mathbf{x} + b)$.
Given $\mathbf{x} \in \mathbb{R}^d$, the side of the hyperplane H it is on is determined by the sign of $\mathbf{w} \cdot \mathbf{x} + b$. Our half-space model: $h : \mathbb{R}^d \setminus H \longrightarrow \{1, -1\}$.

- (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 $\mathbf{x} \cdot \mathbf{x} + b < 0$.

Given data with $\{\pm 1\}$ labels, if there exists a hyperplane H so that $\mathbf{x}$ has label 1 if and only if it is on the positive side, the labeled data are called **linearly separable**.

Linearly separable

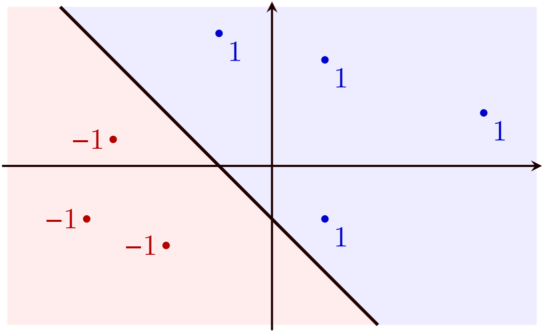


Figure: The hyperplane $H = \{(x_1, x_2) \in \mathbb{R}^2 : x_1 + x_2 + 1 = 0\}$, corresponding positive and negative regions, $\mathbf{w} = (1, 1)$, $b = 1$

Not linearly separable

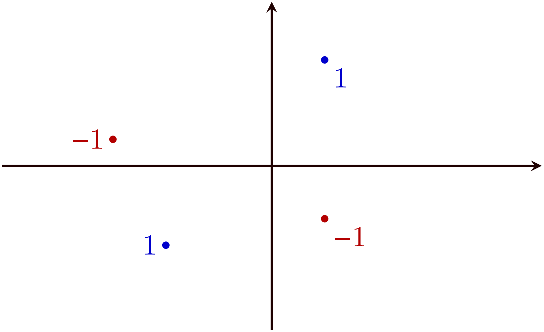


Figure: A data set in $\mathbb{R}^2$ that is not linearly separable.

Not linearly separable

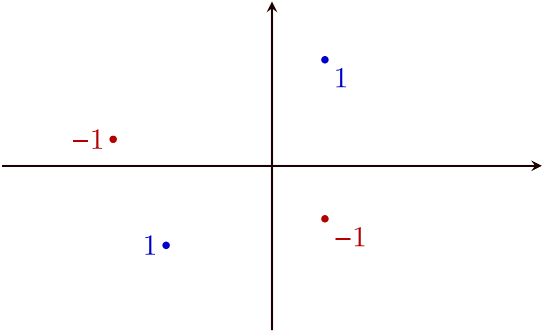


Figure: A data set in $\mathbb{R}^2$ that is not linearly separable.

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

Outline

Setup for Perceptron algorithm

Labeled data: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{\pm 1\}$ for all i.
Assuming labeled data is linearly separable, the Perceptron algorithm is a procedure that is guaranteed to find a hyperplane that separates the data.[1]

[1] Introduced in *The perceptron: A probabilistic model for information storage and organization in the brain*, F. Rosenblatt, Psychological Review **65** (1958), 386–407.

Setup for Perceptron algorithm

Labeled data: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{\pm 1\}$ for all i.
Assuming labeled data is linearly separable, the Perceptron algorithm is a procedure that is guaranteed to find a hyperplane that separates the data.[1]
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.

[1] Introduced in *The perceptron: A probabilistic model for information storage and organization in the brain*, F. Rosenblatt, Psychological Review **65** (1958), 386–407.

Setup for Perceptron algorithm

Labeled data: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{\pm 1\}$ for all i.
Assuming labeled data is linearly separable, the Perceptron algorithm is a procedure that is guaranteed to find a hyperplane that separates the data.[1]
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 $W \cdot X_i$ and y_i have the same sign (both positive or both negative), for all $1 \leq i \leq n$.

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

[1] Introduced in *The perceptron: A probabilistic model for information storage and organization in the brain*, F. Rosenblatt, Psychological Review **65** (1958), 386–407.

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.[2]

[2] Recall, in pseudo-code block, left-facing arrow means *assign* to variable on left.

Perceptron algorithm

Suppose the data is linearly separable. Also, x is an $n \times d$ array of points, with ith row equal to $\mathbf{x}_i$, and y is array of the labels. The Perceptron algorithm finds W iteratively as follows.[2]

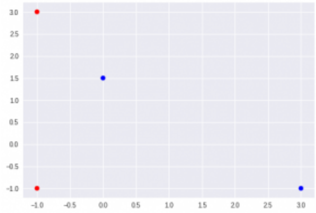
```
input: x, y   ## x is n by d, y is 1d array
X ← append 1 to each row of x
W ← (0,0,...,0)   ## Initial W
while (exists i with y[i]*dot(W, X[i]) ≤ 0){
    W ← W + y[i]*X[i]
}
return W
```

[2]Recall, in pseudo-code block, left-facing arrow means *assign* to variable on left.

Example

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

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



Example, continued

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

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

Use $W^{(t)}$ for value of W on step t. Start: $W^{(1)} = (0, 0, 0)$.
Next step: $W^{(2)} = \vec{0} + y_1 X_1 = -1 * (-1, 3, 1) = (1, -3, -1)$.

Example, continued

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

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

Use $W^{(t)}$ for value of W on step t. Start: $W^{(1)} = (0, 0, 0)$.
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
$y_2 W^{(2)} \cdot X_2 = -1 * (-1 + 3 - 1) = -1$.

Example, continued

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

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

Use $W^{(t)}$ for value of W on step t. Start: $W^{(1)} = (0, 0, 0)$.
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,

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

Example, continued

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

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

Use $W^{(t)}$ for value of W on step t. Start: $W^{(1)} = (0,0,0)$.
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 $y_1 X_1, y_2 X_2, y_3 X_3, y_4 X_4$. Eventually you return the vector $W^{(10)} = (4,-0.5,1)$.

Example, continued

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

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

Use $W^{(t)}$ for value of W on step t. Start: $W^{(1)} = (0, 0, 0)$.
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 $y_1 X_1, y_2 X_2, y_3 X_3, y_4 X_4$. Eventually you return the vector $W^{(10)} = (4, -0.5, 1)$.
i.e., $H = \{(x_1, x_2) \in \mathbb{R}^2 : \; 4x_1 - 0.5x_2 + 1 = 0\}$ 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 $W^{(1)} = (0, 0, \dots, 0)$. 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 $W^{(1)} = (0, 0, \dots, 0)$. 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 $T \leq (BR)^2$.

Another example, the Iris data set

First discussed by R.A. Fisher in a 1936 paper, Iris data set commonly used in explanations. It contains 150 points in $\mathbb{R}^4$, each for an individual iris flower from one of 3 species: Iris setosa, Iris virginica, and Iris versicolor.

Figure: Images by G. Robertson, E. Hunt, Radomil ©CC BY-SA 3.0

Another example, the Iris data set

First discussed by R.A. Fisher in a 1936 paper, Iris data set commonly used in explanations. It contains 150 points in $\mathbb{R}^4$, each for an individual iris flower from one of 3 species: Iris setosa, Iris virginica, and Iris versicolor. The 4 coordinates are measurements of sepal length, sepal width, petal length, and petal width (in cm).

Figure: Images by G. Robertson, E. Hunt, Radomil ©CC BY-SA 3.0

Another example, the Iris data set

First discussed by R.A. Fisher in a 1936 paper, Iris data set commonly used in explanations. It contains 150 points in $\mathbb{R}^4$, each for an individual iris flower from one of 3 species: Iris setosa, Iris virginica, and Iris versicolor. The 4 coordinates are measurements of sepal length, sepal width, petal length, and petal width (in cm).

Iris setosa points are linearly separable from the other two.

Labels: *Iris setosa* $\longleftarrow$ 1; *Other species* $\longleftarrow$ -1.

Figure: Images by G. Robertson, E. Hunt, Radomil ©CC BY-SA 3.0

Another example, the Iris data set

First discussed by R.A. Fisher in a 1936 paper, Iris data set commonly used in explanations. It contains 150 points in $\mathbb{R}^4$, each for an individual iris flower from one of 3 species: Iris setosa, Iris virginica, and Iris versicolor. The 4 coordinates are measurements of sepal length, sepal width, petal length, and petal width (in cm).

Iris setosa points are linearly separable from the other two.

Labels: *Iris setosa* $\longleftarrow$ 1; *Other species* $\longleftarrow$ -1.

Begin by opening the notebook `'perceptron-iris-notebook.ipynb'` ...After completing the algorithm, should get final $W = (\mathbf{w}, b)$, where $\mathbf{w} = (1.3, 4.1, -5.2, -2.2)$ and $b = 1$.

Figure: Images by G. Robertson, E. Hunt, Radomil ©CC BY-SA 3.0