Gradient descent on Logistic model

Chris Cornwell

Mar 27, 2025

Outline

Gradient Descent in Logistic model

Setup

As usual, sample data $\mathcal{S}$; write (x_i, y_i) for point in $\mathcal{S}$, with $y_i \in \{1, -1\}$. Parameter space: $\Omega = \mathbb{R}^{d+1} = \{(\mathrm{w}, b) \mid \mathrm{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$. For each $\omega = (\mathrm{w}, b) \in \Omega$, have function $f_\omega : \mathbb{R}^d \to (0, 1)$ defined as

$$f_\omega(\mathrm{x}) = \sigma(\mathrm{w} \cdot \mathrm{x} + b),$$

where σ is the logistic function. (So, our function output is our probability that the correct label for x is $+1$.)

Setup

As usual, sample data $\mathcal{S}$; write (x_i, y_i) for point in $\mathcal{S}$, with $y_i \in \{1, -1\}$. Parameter space: $\Omega = \mathbb{R}^{d+1} = \{(\mathrm{w}, b) \mid \mathrm{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$. For each $\omega = (\mathrm{w}, b) \in \Omega$, have function $f_\omega : \mathbb{R}^d \to (0, 1)$ defined as

$$f_\omega(\mathrm{x}) = \sigma(\mathrm{w} \cdot \mathrm{x} + b),$$

where σ is the logistic function. (So, our function output is our probability that the correct label for x is $+1$.)

What do we use for the empirical loss function?

- *Considering how model should work*: when correct y is $+1$ (and x is far from the hyperplane), good situation is when $f_\omega(\mathrm{x})$ very close to 1. Alternatively, if the correct y is -1, then we would like $f_\omega(\mathrm{x})$ to be close to 0.

Setup

As usual, sample data $\mathcal{S}$; write (x_i, y_i) for point in $\mathcal{S}$, with $y_i \in \{1, -1\}$. Parameter space: $\Omega = \mathbb{R}^{d+1} = \{(\mathrm{w}, b) \mid \mathrm{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$. For each $\omega = (\mathrm{w}, b) \in \Omega$, have function $f_\omega : \mathbb{R}^d \to (0, 1)$ defined as

$$f_\omega(\mathrm{x}) = \sigma(\mathrm{w} \cdot \mathrm{x} + b),$$

where σ is the logistic function. (So, our function output is our probability that the correct label for x is $+1$.)

What do we use for the empirical loss function?

- *From considerations about model*: It helps to alter numerical values of our labels: for i with $y_i = 1$, define $\tilde{y}_i = 1$; for i with $y_i = -1$, define $\tilde{y}_i = 0$.[1]

[1] Functionally, this can be achieved by the definition $\tilde{y}_i = (y_i + 1)/2$.

Setup

As usual, sample data $\mathcal{S}$; write (x_i, y_i) for point in $\mathcal{S}$, with $y_i \in \{1, -1\}$. Parameter space: $\Omega = \mathbb{R}^{d+1} = \{(\mathrm{w}, b) \mid \mathrm{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$. For each $\omega = (\mathrm{w}, b) \in \Omega$, have function $f_\omega : \mathbb{R}^d \to (0, 1)$ defined as

$$f_\omega(\mathrm{x}) = \sigma(\mathrm{w} \cdot \mathrm{x} + b),$$

where σ is the logistic function. (So, our function output is our probability that the correct label for x is $+1$.)

What do we use for the empirical loss function?

- *From considerations about model*: It helps to alter numerical values of our labels: for i with $y_i = 1$, define $\tilde{y}_i = 1$; for i with $y_i = -1$, define $\tilde{y}_i = 0$.[1]

We then use the log-loss function (a.k.a. binary cross-entropy). It is defined by

$$\mathcal{L}_{\mathcal{S}}(\omega) = \frac{1}{n} \sum_{i=1}^{n} -\tilde{y}_i \log(f_\omega(\mathrm{x}_i)) - (1 - \tilde{y}_i) \log(1 - f_\omega(\mathrm{x}_i)).$$

[1] Functionally, this can be achieved by the definition $\tilde{y}_i = (y_i + 1)/2$.

Gradient descent with logistic regression

Let's compute the gradient of the log-loss. Pick $1 \leq i \leq n$, and write the coordinates (also called *features*) of the vector $\mathrm{x}_i \in \mathbb{R}^d$ as $(x_{i,1}, x_{i,2}, \ldots, x_{i,d})$.

Gradient descent with logistic regression

Let's compute the gradient of the log-loss. Pick $1 \leq i \leq n$, and write the coordinates (also called *features*) of the vector $\mathrm{x}_i \in \mathbb{R}^d$ as $(x_{i,1}, x_{i,2}, \ldots, x_{i,d})$.
The per-example log-loss at x_i is

$$-\tilde{y}_i \log(f_\omega(\mathrm{x}_i)) - (1 - \tilde{y}_i) \log(1 - f_\omega(\mathrm{x}_i)).$$

Gradient descent with logistic regression

Let's compute the gradient of the log-loss. Pick $1 \leq i \leq n$, and write the coordinates (also called *features*) of the vector $\mathrm{x}_i \in \mathbb{R}^d$ as $(x_{i,1}, x_{i,2}, \ldots, x_{i,d})$.
The per-example log-loss at x_i is

$$-\tilde{y}_i \log(f_\omega(\mathrm{x}_i)) - (1 - \tilde{y}_i) \log(1 - f_\omega(\mathrm{x}_i)).$$

If $\tilde{y}_i = 1$ then this is simply $-\log(f_\omega(\mathrm{x}_i))$. Recall that $\sigma(z)$ has a derivative satisfying $\frac{d\sigma}{dz} = \sigma(z) * (1 - \sigma(z))$.

Gradient descent with logistic regression

Let's compute the gradient of the log-loss. Pick $1 \le i \le n$, and write the coordinates (also called *features*) of the vector $\mathrm{x}_i \in \mathbb{R}^d$ as $(x_{i,1}, x_{i,2}, \ldots, x_{i,d})$.
The per-example log-loss at x_i is

$$-\tilde{y}_i \log(f_\omega(\mathrm{x}_i)) - (1 - \tilde{y}_i) \log(1 - f_\omega(\mathrm{x}_i)).$$

If $\tilde{y}_i = 1$ then this is simply $-\log(f_\omega(\mathrm{x}_i))$. Recall that $\sigma(z)$ has a derivative satisfying $\frac{d\sigma}{dz} = \sigma(z) * (1 - \sigma(z))$. Using this and the fact that $\frac{\partial}{\partial w_j}(\mathrm{w} \cdot \mathrm{x}_i + b) = x_{i,j}$, we get

$$\frac{\partial}{\partial w_j}(-\log(f_\omega(\mathrm{x}_i))) = -\frac{x_{i,j} * f_\omega(\mathrm{x}_i) * (1 - f_\omega(\mathrm{x}_i))}{f_\omega(\mathrm{x}_i)} = -x_{i,j} * (1 - f_\omega(\mathrm{x}_i)).$$

Gradient descent with logistic regression

Let's compute the gradient of the log-loss. Pick $1 \leq i \leq n$, and write the coordinates (also called *features*) of the vector $\mathrm{x}_i \in \mathbb{R}^d$ as $(x_{i,1}, x_{i,2}, \ldots, x_{i,d})$.
The per-example log-loss at x_i is

$$-\tilde{y}_i \log(f_\omega(\mathrm{x}_i)) - (1-\tilde{y}_i)\log(1-f_\omega(\mathrm{x}_i)).$$

If $\tilde{y}_i = 1$ then this is simply $-\log(f_\omega(\mathrm{x}_i))$. Recall that $\sigma(z)$ has a derivative satisfying $\frac{d\sigma}{dz} = \sigma(z) * (1-\sigma(z))$. Using this and the fact that $\frac{\partial}{\partial w_j}(\mathrm{w} \cdot \mathrm{x}_i + b) = x_{i,j}$, we get

$$\frac{\partial}{\partial w_j}(-\log(f_\omega(\mathrm{x}_i))) = -\frac{x_{i,j} * f_\omega(\mathrm{x}_i) * (1-f_\omega(\mathrm{x}_i))}{f_\omega(\mathrm{x}_i)} = -x_{i,j} * (1-f_\omega(\mathrm{x}_i)).$$

The partial with respect to b is simply $-(1-f_\omega(\mathrm{x}_i))$.

Gradient descent with logistic regression

On the other hand, if $\tilde{y}_i = 0$ then the per-example log-loss at x_i is $-\log(1 - f_\omega(\mathrm{x}_i))$.

Gradient descent with logistic regression

On the other hand, if $\tilde{y}_i = 0$ then the per-example log-loss at x_i is $-\log(1 - f_\omega(\mathrm{x}_i))$. Consequently, we compute

$$\frac{\partial}{\partial w_j}\left(-\log(1 - f_\omega(\mathrm{x}_i))\right) = \frac{x_{i,j} * f_\omega(\mathrm{x}_i) * (1 - f_\omega(\mathrm{x}_i))}{1 - f_\omega(\mathrm{x}_i)} = x_{i,j} * f_\omega(\mathrm{x}_i),$$

and the partial with respect to b is $f_\omega(\mathrm{x}_i)$.

Gradient descent with logistic regression

On the other hand, if $\tilde{y}_i = 0$ then the per-example log-loss at x_i is $-\log(1 - f_\omega(\mathrm{x}_i))$. Consequently, we compute

$$\frac{\partial}{\partial w_j}\left(-\log(1 - f_\omega(\mathrm{x}_i))\right) = \frac{x_{i,j} * f_\omega(\mathrm{x}_i) * (1 - f_\omega(\mathrm{x}_i))}{1 - f_\omega(\mathrm{x}_i)} = x_{i,j} * f_\omega(\mathrm{x}_i),$$

and the partial with respect to b is $f_\omega(\mathrm{x}_i)$.
Hence, we have, for each $1 \leq j \leq d$,

$$\frac{\partial}{\partial w_j}\mathcal{L}_{\mathcal{S}} = \frac{1}{n}\sum_{i=1}^{n} -\tilde{y}_i x_{i,j} * (1 - f_\omega(\mathrm{x}_i)) + (1 - \tilde{y}_i) x_{i,j} f_\omega(\mathrm{x}_i),$$

and

$$\frac{\partial}{\partial b}\mathcal{L}_{\mathcal{S}} = \frac{1}{n}\sum_{i=1}^{n} -\tilde{y}_i * (1 - f_\omega(\mathrm{x}_i)) + (1 - \tilde{y}_i) f_\omega(\mathrm{x}_i).$$

Similarity to Perceptron algorithm updates

Consider the per-example loss: given one $(\mathrm{x}_i, y_i) \in \mathcal{S}$, have the term $-\tilde{y}_i \log(f_\omega(\mathrm{x}_i)) - (1 - \tilde{y}_i) \log(1 - f_\omega(\mathrm{x}))$.
Its partial w.r.t. w_j is

$$-\tilde{y}_i x_{i,j} * (1 - f_\omega(\mathrm{x}_i)) + (1 - \tilde{y}_i) x_{i,j} f_\omega(\mathrm{x}_i).$$

Similarity to Perceptron algorithm updates

Consider the per-example loss: given one $(\mathrm{x}_i, y_i) \in \mathcal{S}$, have the term $-\tilde{y}_i \log(f_\omega(\mathrm{x}_i)) - (1 - \tilde{y}_i) \log(1 - f_\omega(\mathrm{x}))$.
Its partial w.r.t. w_j is

$$-\tilde{y}_i x_{i,j} * (1 - f_\omega(\mathrm{x}_i)) + (1 - \tilde{y}_i) x_{i,j} f_\omega(\mathrm{x}_i).$$

If $\tilde{y}_i = 1$, but $f_\omega(\mathrm{x}_i)$ is close to 0, then this is *almost* $-x_{i,j} = -y_i x_{i,j}$. Were we to do an update with $\eta = 1$ then we would have $w_j^{(t)} \approx w_j^{(t-1)} + y_i x_{i,j}$. (The Perceptron update in the j^{th} coordinate.)

Similarity to Perceptron algorithm updates

Consider the per-example loss: given one $(\mathrm{x}_i, y_i) \in \mathcal{S}$, have the term $-\tilde{y}_i \log(f_\omega(\mathrm{x}_i)) - (1 - \tilde{y}_i) \log(1 - f_\omega(\mathrm{x}))$.
Its partial w.r.t. w_j is

$$-\tilde{y}_i x_{i,j} * (1 - f_\omega(\mathrm{x}_i)) + (1 - \tilde{y}_i) x_{i,j} f_\omega(\mathrm{x}_i).$$

If $\tilde{y}_i = 1$, but $f_\omega(\mathrm{x}_i)$ is close to 0, then this is *almost* $-x_{i,j} = -y_i x_{i,j}$. Were we to do an update with $\eta = 1$ then we would have $w_j^{(t)} \approx w_j^{(t-1)} + y_i x_{i,j}$. (The Perceptron update in the j^{th} coordinate.)
Likewise, if $\tilde{y}_i = 0$, but $f_\omega(\mathrm{x}_i)$ is close to 1, then the partial is approximately $x_{i,j}$; so, using $\eta = 1$, we would get $w_j^{(t)} \approx w_j^{(t-1)} - x_{i,j} = w_j^{(t-1)} + y_i x_{i,j}$. (Like the Perceptron algorithm update again.)

Example with Logistic model

Given ± 1-labeled sample data $\mathcal{S} = \{(\mathrm{x}_i, y_i)\}_{i=1}^n$, we found an expression for each partial derivative of $\mathcal{L}_{\mathcal{S}}$. In fact, if we call $w_{d+1} = b$ and $x_{i,d+1} = 1$ for every $1 \leq i \leq n$, then

$$\frac{\partial}{\partial w_j}\mathcal{L}_{\mathcal{S}} = \frac{1}{n}\sum_{i=1}^{n} -\tilde{y}_i x_{i,j}(1 - f_\omega(\mathrm{x}_i)) + (1 - \tilde{y}_i)x_{i,j} f_\omega(\mathrm{x}_i)$$

for every $1 \leq j \leq d+1$.

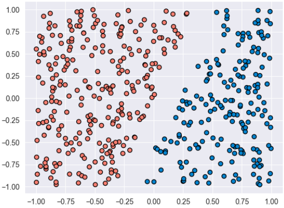
Example with Logistic model

Given ±1-labeled sample data $\mathcal{S} = \{(\mathrm{x}_i, y_i)\}_{i=1}^n$, we found an expression for each partial derivative of $\mathcal{L}_{\mathcal{S}}$. In fact, if we call $w_{d+1} = b$ and $x_{i,d+1} = 1$ for every $1 \le i \le n$, then

$$\frac{\partial}{\partial w_j}\mathcal{L}_{\mathcal{S}} = \frac{1}{n}\sum_{i=1}^{n} -\tilde{y}_i x_{i,j}(1 - f_\omega(\mathrm{x}_i)) + (1 - \tilde{y}_i)x_{i,j}f_\omega(\mathrm{x}_i)$$

for every $1 \le j \le d+1$.

Example: consider simulated sample data below, drawn uniformly from a subset of $[-1,1]^2$, consisting of points at least a fixed distance from $H = \{(x_1, x_2) \in \mathbb{R}^2 \mid 2x_1 - \frac{2}{3}x_2 - \frac{1}{5} = 0\}$. Positively labeled points are in blue.



Example with Logistic model

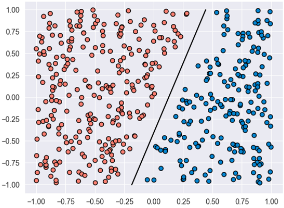
Given ± 1-labeled sample data $\mathcal{S} = \{(\mathrm{x}_i, y_i)\}_{i=1}^n$, we found an expression for each partial derivative of $\mathcal{L}_\mathcal{S}$. In fact, if we call $w_{d+1} = b$ and $x_{i,d+1} = 1$ for every $1 \leq i \leq n$, then

$$\frac{\partial}{\partial w_j}\mathcal{L}_\mathcal{S} = \frac{1}{n}\sum_{i=1}^{n} -\tilde{y}_i x_{i,j}(1 - f_\omega(\mathrm{x}_i)) + (1 - \tilde{y}_i)x_{i,j} f_\omega(\mathrm{x}_i)$$

for every $1 \leq j \leq d+1$.

Example (cont'd): Batch gradient descent, with learning rate 0.5, stopping with threshhold 0.0005, gives (approximately) parameters for:

$$\hat{H} = \{(x_1, x_2) \in \mathbb{R}^2 \mid 2.01x_1 - 0.63x_2 - 0.26 = 0\}.$$



Logistic regression on the Iris data

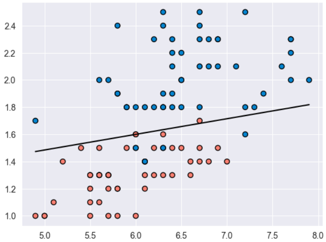
Recall the Iris data set: 150 points, 50 from each of three species of Iris flower. Two of the species in the data set, Iris versicolor and Iris virginica, are not linearly separable.

Logistic regression on the Iris data

Recall the Iris data set: 150 points, 50 from each of three species of Iris flower. Two of the species in the data set, Iris versicolor and Iris virginica, are not linearly separable.
We can use gradient descent on the logistic model to find a hyperplane that does *well* in classifying versicolor vs. virginica species.

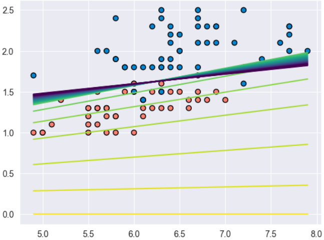
Logistic regression on the Iris data

Recall the Iris data set: 150 points, 50 from each of three species of Iris flower. Two of the species in the data set, Iris versicolor and Iris virginica, are not linearly separable.
We can use gradient descent on the logistic model to find a hyperplane that does *well* in classifying versicolor vs. virginica species.
For visualization, I first show just the first and fourth coordinates, and the results for logistic model in 2D.



Logistic regression on the Iris data

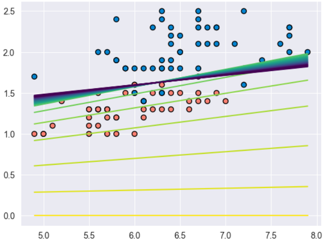
Pictured below are selected lines found during the batch gradient descent.



[2]Recall, the model labels the point with $+1$ when $f_\omega(\mathrm{x}) \geq 0.5$.

Logistic regression on the Iris data

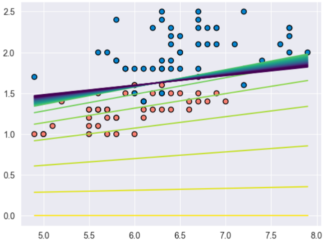
Pictured below are selected lines found during the batch gradient descent.
As before, yellow-to-purple is progression through the procedure.
Consecutive lines that are shown have 200 updates between them; ≈ 4000 updates in total.



[2]Recall, the model labels the point with $+1$ when $f_\omega(\mathrm{x}) \geq 0.5$.

Logistic regression on the Iris data

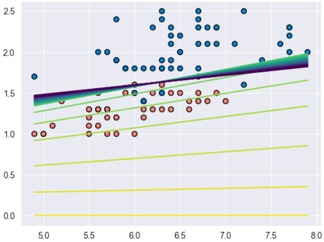
Pictured below are selected lines found during the batch gradient descent.
As before, yellow-to-purple is progression through the procedure.
Consecutive lines that are shown have 200 updates between them; ≈ 4000 updates in total.
The accuracy in this 2D projection is 92% (the final hyperplane correctly labeled 92 out of 100).[2]



[2]Recall, the model labels the point with $+1$ when $f_\omega(\mathrm{x}) \geq 0.5$.

Logistic regression on the Iris data

Pictured below are selected lines found during the batch gradient descent.
As before, yellow-to-purple is progression through the procedure.
Consecutive lines that are shown have 200 updates between them; ≈ 4000 updates in total.
The accuracy in this 2D projection is 92% (the final hyperplane correctly labeled 92 out of 100).[2]
The model on the points in $\mathbb{R}^4$ took less updates (just under 3000). It had 97% accuracy on the data.



[2] Recall, the model labels the point with $+1$ when $f_\omega(\mathrm{x}) \geq 0.5$.