CSC 411 Lecture 7: Linear Classification

Roger Grosse, Amir-massoud Farahmand, and Juan Carrasquilla

University of Toronto

Overview

- Classification: predicting a discrete-valued target
  - Binary classification: predicting a binary-valued target
- Examples
  - predict whether a patient has a disease, given the presence or absence of various symptoms
  - classify e-mails as spam or non-spam
  - predict whether a financial transaction is fraudulent

Overview

Binary linear classification

- **classification:** predict a discrete-valued target
- **binary:** predict a binary target $t \in \{0, 1\}$
  - Training examples with $t = 1$ are called positive examples, and training examples with $t = 0$ are called negative examples. Sorry.
- **linear:** model is a linear function of $\mathbf{x}$, followed by a threshold:

$$z = \mathbf{w}^T \mathbf{x} + b$$

$$y = \begin{cases} 1 & \text{if } z \geq r \\ 0 & \text{if } z < r \end{cases}$$

Some simplifications

Eliminating the threshold

- We can assume WLOG that the threshold $r = 0$:

$$\mathbf{w}^T\mathbf{x} + b \geq r \quad \Longleftrightarrow \quad \mathbf{w}^T\mathbf{x} + \underbrace{b - r}_{\triangleq b'} \geq 0.$$

Eliminating the bias

- Add a dummy feature x_0 which always takes the value 1. The weight w_0 is equivalent to a bias.

Simplified model

$$z = \mathbf{w}^T\mathbf{x}$$

$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

Examples

NOT

x_0	x_1	t
1	0	1
1	1	0

$$b > 0$$
$$b + w < 0$$

$$b = 1,\ w = -2$$

Examples

AND

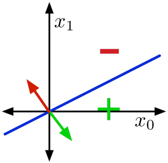
x_0	x_1	x_2	t
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

$$
\begin{aligned}
b &< 0 \\
b + w_2 &< 0 \\
b + w_1 &< 0 \\
b + w_1 + w_2 &> 0
\end{aligned}
$$

$$b = -1.5, \; w_1 = 1, \; w_2 = 1$$

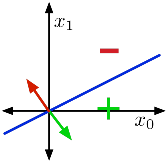
The Geometric Picture

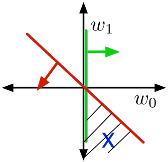
Input Space, or Data Space



- Here we're visualizing the **NOT** example
- Training examples are points
- Hypotheses are half-spaces whose boundaries pass through the origin
- The boundary is the decision boundary
  - In 2-D, it's a line, but think of it as a hyperplane
- If the training examples can be separated by a linear decision rule, they are linearly separable.

Weight Space





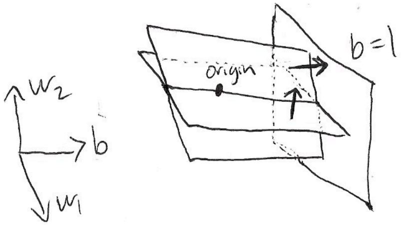
$w_0 > 0$

$w_0 + w_1 < 0$

- Hypotheses are points
- Training examples are half-spaces whose boundaries pass through the origin
- The region satisfying all the constraints is the feasible region; if this region is nonempty, the problem is feasible

The Geometric Picture

- The **AND** example requires three dimensions, including the dummy one.
- To visualize data space and weight space for a 3-D example, we can look at a 2-D slice:

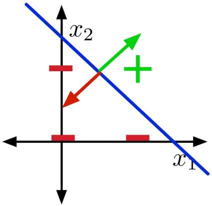


- The visualizations are similar, except that the decision boundaries and the constraints need not pass through the origin.

The Geometric Picture

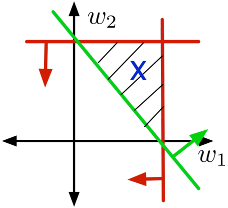
Visualizations of the **AND** example

Data Space



Slice for $x_0 = 1$

Weight Space

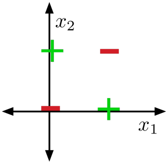


Slice for $w_0 = -1$

What happened to the fourth constraint?

The Geometric Picture

Some datasets are not linearly separable, e.g. **XOR**



Proof coming next lecture...

Overview

- **Recall: binary linear classifiers.** Targets $t \in \{0, 1\}$

$$z = \mathbf{w}^T \mathbf{x} + b$$
$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

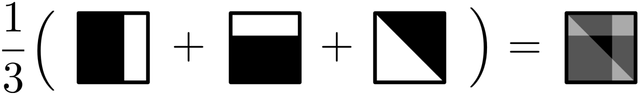
- What if we can't classify all the training examples correctly?
- Seemingly obvious loss function: 0-1 loss

$$\mathcal{L}_{0-1}(y, t) = \begin{cases} 0 & \text{if } y = t \\ 1 & \text{if } y \neq t \end{cases}$$
$$= \mathbb{1}_{y \neq t}.$$

Attempt 1: 0-1 loss

- As always, the cost $\mathcal{J}$ is the average loss over training examples; for 0-1 loss, this is the error rate:

$$\mathcal{J} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{y^{(i)} \neq t^{(i)}}$$



Attempt 1: 0-1 loss

- Problem: how to optimize?
- Chain rule:

$$\frac{\partial \mathcal{L}_{0-1}}{\partial w_j} = \frac{\partial \mathcal{L}_{0-1}}{\partial z} \frac{\partial z}{\partial w_j}$$

- But $\partial \mathcal{L}_{0-1} / \partial z$ is zero everywhere it's defined!
  - $\partial \mathcal{L}_{0-1} / \partial w_j = 0$ means that changing the weights by a very small amount probably has no effect on the loss.
  - The gradient descent update is a no-op.

Attempt 2: Linear Regression

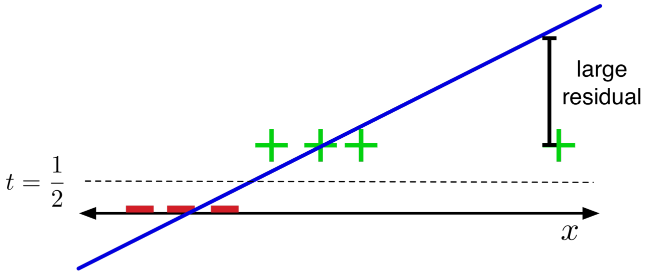
- Sometimes we can replace the loss function we care about with one which is easier to optimize. This is known as a surrogate loss function.
- We already know how to fit a linear regression model. Can we use this instead?

$$y = \mathbf{w}^\top \mathbf{x} + b$$
$$\mathcal{L}_{\mathrm{SE}}(y, t) = \frac{1}{2}(y - t)^2$$

- Doesn't matter that the targets are actually binary.
- Threshold predictions at $y = 1/2$.

Attempt 2: Linear Regression

The problem:



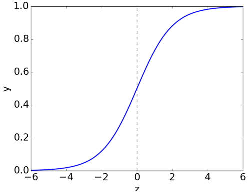
- The loss function hates when you make correct predictions with high confidence!
- If $t = 1$, it's more unhappy about $y = 10$ than $y = 0$.

Attempt 3: Logistic Activation Function

- There's obviously no reason to predict values outside [0, 1]. Let's squash y into this interval.

- The logistic function is a kind of sigmoidal, or S-shaped, function:

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



- A linear model with a logistic nonlinearity is known as log-linear:

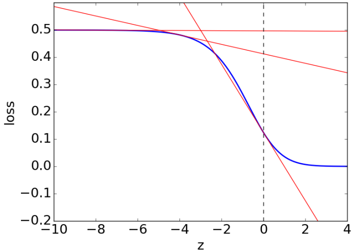
$$z = \mathbf{w}^\top \mathbf{x} + b$$
$$y = \sigma(z)$$
$$\mathcal{L}_{\mathrm{SE}}(y, t) = \frac{1}{2}(y - t)^2.$$

- Used in this way, σ is called an activation function, and z is called the logit.

Attempt 3: Logistic Activation Function

The problem:

(plot of $\mathcal{L}_{\mathrm{SE}}$ as a function of z)



$$\frac{\partial \mathcal{L}}{\partial w_j} = \frac{\partial \mathcal{L}}{\partial z} \frac{\partial z}{\partial w_j}$$

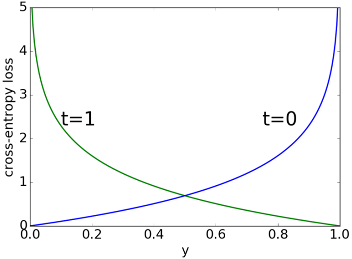
$$w_j \leftarrow w_j - \alpha \frac{\partial \mathcal{L}}{\partial w_j}$$

- In gradient descent, a small gradient (in magnitude) implies a small step.
- If the prediction is really wrong, shouldn't you take a large step?

Logistic Regression

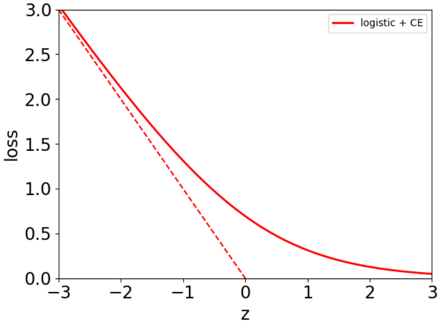
- Because $y \in [0, 1]$, we can interpret it as the estimated probability that $t = 1$.
- The pundits who were 99% confident Clinton would win were much more wrong than the ones who were only 90% confident.
- Cross-entropy loss captures this intuition:

$$\mathcal{L}_{\mathrm{CE}}(y, t) = \begin{cases} -\log y & \text{if } t = 1 \\ -\log(1 - y) & \text{if } t = 0 \end{cases}$$
$$= -t \log y - (1 - t) \log(1 - y)$$



Logistic Regression:

$$
\begin{aligned}
z &= \mathbf{w}^\top \mathbf{x} + b \\
y &= \sigma(z) \\
&= \frac{1}{1 + e^{-z}} \\
\mathcal{L}_{\mathrm{CE}} &= -t \log y - (1 - t) \log(1 - y)
\end{aligned}
$$



[[gradient derivation in the notes]]

Logistic Regression

- Problem: what if $t = 1$ but you're really confident it's a negative example ($z \ll 0$)?
- If y is small enough, it may be numerically zero. This can cause very subtle and hard-to-find bugs.

$$
\begin{aligned}
y &= \sigma(z) & \Rightarrow y \approx 0 \\
\mathcal{L}_{\mathrm{CE}} &= -t \log y - (1-t)\log(1-y) & \Rightarrow \text{ computes } \log 0
\end{aligned}
$$

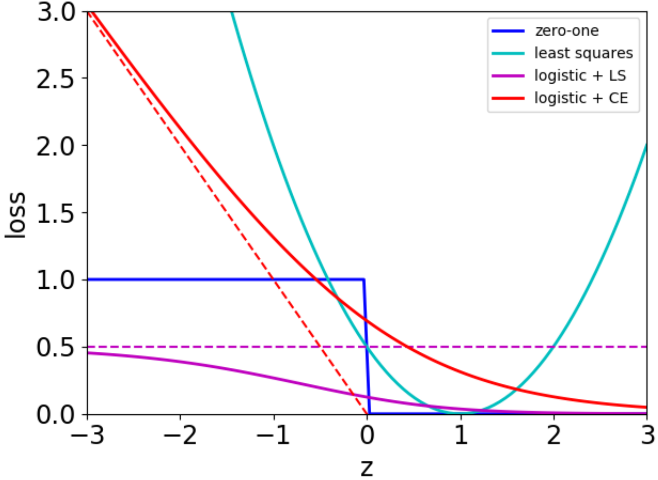
- Instead, we combine the activation function and the loss into a single logistic-cross-entropy function.

$$
\mathcal{L}_{\mathrm{LCE}}(z, t) = \mathcal{L}_{\mathrm{CE}}(\sigma(z), t) = t \log(1 + e^{-z}) + (1-t)\log(1 + e^{z})
$$

- Numerically stable computation:

```
E = t * np.logaddexp(0, -z) + (1-t) * np.logaddexp(0, z)
```

Comparison of loss functions:



Logistic Regression

Comparison of gradient descent updates:

- Linear regression:

$$\mathbf{w} \leftarrow \mathbf{w} - \frac{\alpha}{N} \sum_{i=1}^{N} (y^{(i)} - t^{(i)})\, \mathbf{x}^{(i)}$$

- Logistic regression:

$$\mathbf{w} \leftarrow \mathbf{w} - \frac{\alpha}{N} \sum_{i=1}^{N} (y^{(i)} - t^{(i)})\, \mathbf{x}^{(i)}$$

- Not a coincidence! These are both examples of matching loss functions, but that's beyond the scope of this course.