Linear Classifiers, Logistic Regression, Multiclass Classification

Overview

Classification: predicting a discrete-valued target

- Binary classification: predicting a binary-valued target
- Multiclass classification: predicting a discrete(>2)-valued target

Binary linear classification

classification: given a D-dimensional input $\mathbf{x} \in \mathbb{R}^D$ 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.
- $t \in \{0, 1\}$ or $t \in \{-1, +1\}$ is for computational convenience.

linear: model prediction y is a linear function of $\mathbf{x}$, followed by a threshold r:

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

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

Simplifications

我们对上面的 binary linear classification 进行简化。

- Eliminating the threshold（消除阈值）

  我们假设 threshold r = 0（该假设“不失一般性”，without loss of generality or WLOG，即该个例能代表普遍情况，而非一种特例）：

  $$\mathbf{w}^\top \mathbf{x} + b \geq r \quad \Longleftrightarrow \quad \mathbf{w}^\top \mathbf{x} + \underbrace{b - r}_{\triangleq w_0} \geq 0$$

  注：$\triangleq$ 为恒等式，即 $b - r = w_0$

- Eliminating the bias

在上式中，b 是 bias。所以，可以和 linear regression 中一样：

在 w 中增加一列 w_0，使 w_0 = b（实际上是 b - r），然后增加一个 dummy feature x_0（x_0 永远为 1）。这样在运算的时候是 $w_0^\top x_0 = b$。

- Simplified model

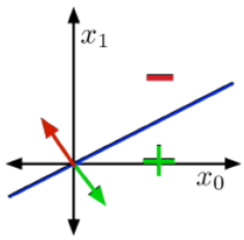
Receive input $\mathbf{x} \in \mathbb{R}^{D+1}$ with $x_0 = 1$:

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

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

The Geometric Picture

Input Space, or Data Space for NOT example



x_0	x_1	t
1	0	1
1	1	0

注：NOT example 是指 t 的结果为 NOT x_1。

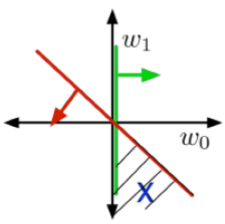
上图中蓝色的线为 decision boundary（在 2D 中，它是一条线；在高维中，它是一个 hyperplane）：$\{\mathbf{x} : \mathbf{w}^\top \mathbf{x} = 0\}$

decision boundary 划分出了两个 half-spaces：

$H_+ = \{\mathbf{x} : \mathbf{w}^\top \mathbf{x} \geq 0\}$，$H_- = \{\mathbf{x} : \mathbf{w}^\top \mathbf{x} < 0\}$。图中红绿色的加减号所在的位置，对应右边表格中 x_0 和 x_1 的值。上方的红色区域是 negative space，下方的绿色区域是 positive space（对于 t）。

如果这个 boundary 可以完美地将 training examples 区分开，我们就说 data is linearly separable。

Weight Space



$w_0 \geq 0$

$w_0 + w_1 < 0$

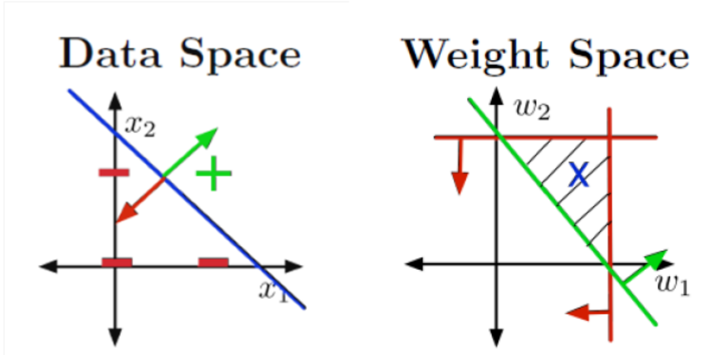
根据上表可得：

- When $x_1 = 0$, need: $z = w_0 x_0 + w_1 x_1 \geq 0 \iff w_0 \geq 0$
- When $x_1 = 1$, need: $z = w_0 x_0 + w_1 x_1 < 0 \iff w_0 + w_1 < 0$

因此，蓝色叉号所在的区域就是 w_0 和 w_1 可以取值的区域，叫做 feasible region。

Visualizations of the **AND** example

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



- Slice for $x_0 = 1$ and
- example sol: $w_0 = -1.5$, $w_1 = 1$, $w_2 = 1$
- decision boundary:

$w_0 x_0 + w_1 x_1 + w_2 x_2 = 0$

$\implies -1.5 + x_1 + x_2 = 0$

- Slice for $w_0 = -1.5$ for the constraints
- $w_0 < 0$
- $w_0 + w_2 < 0$
- $w_0 + w_1 < 0$
- $w_0 + w_1 + w_2 \geq 0$

Towards Logistic Regression

Loss Function

Seemingly obvious loss function: 0-1 loss

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

Usually, the cost $\mathcal{J}$ is the averaged loss over training examples; for 0 - 1 loss, this is the misclassification rate (错分类率):

$$\mathcal{J} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\left[y^{(i)} \neq t^{(i)}\right]$$

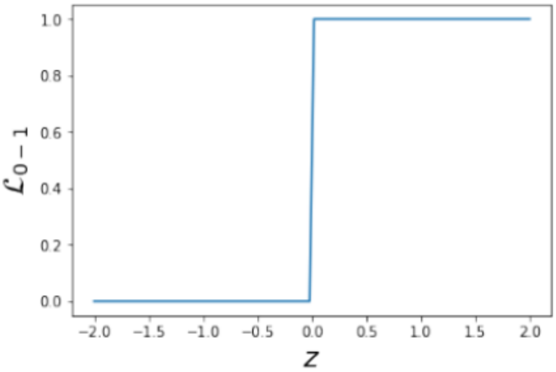
接下来，要对模型进行优化。

Attempt 1：0-1 loss

Minimum of a function will be at its critical points.
Let's try to find the critical point of 0-1 loss
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.
- Almost any point has 0 gradient!

关于 0-1 loss，还有一个问题是：根据最终预测来定义，这本质上是不连续的。所以 0-1 loss 不能用于优化模型。

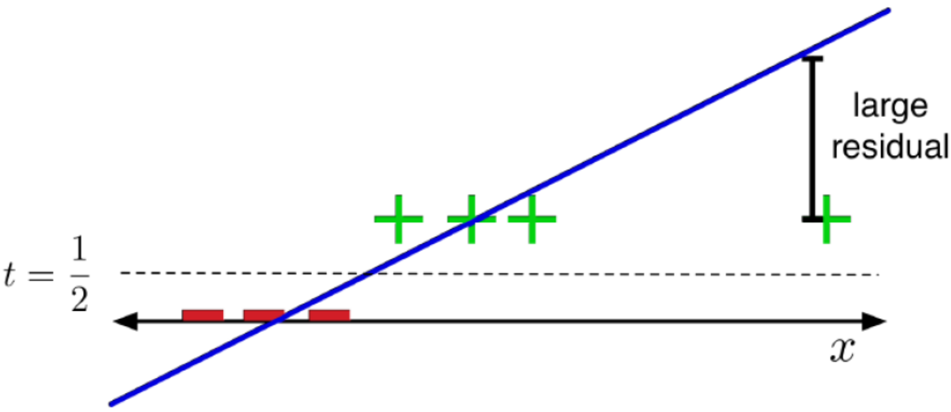
Attempt 2：Linear Regression

有时，我们可以用更容易优化的 loss function 来替换当前的 loss function。这被称为 relaxation with a smooth surrogate loss function（用光滑的替代函数来放松）。

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

$$\mathcal{L}_{\mathrm{SE}}(z,t) = \frac{1}{2}(z-t)^2$$

The problem:



对于该 loss function，z 的阈值 为 $\frac{1}{2}$ 时模型达到最优（看上图，如果阈值为其他数，拟合的线可能是斜的，这样会增大 loss，只有拟合线为 t = $\frac{1}{2}$ 时，loss 最小）。

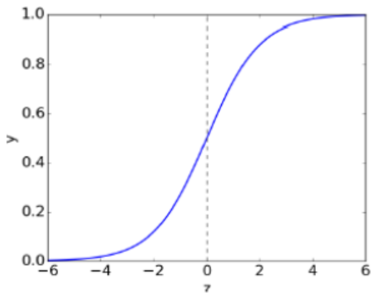
由于上述原因，使用该 loss function 不能很好的进行优化。

Attempt 3：Logistic Activation Function

现在我们将 y 压缩到区间 [0, 1] 之间。

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

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



$\sigma^{-1}(y) = \log(y/(1-y))$ is called the logit.

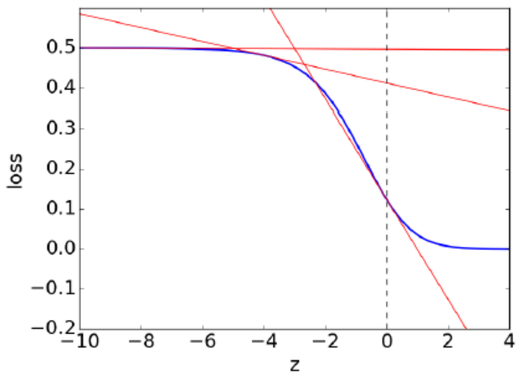
注：logit 和 sigmoid 互为反函数。

具有逻辑非线性 (logistic nonlinearity) 的线性模型称为：log-linear：

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

Used in this way, σ is called an **activation function**.

The problem:
(plot of $\mathcal{L}_{\mathrm{SE}}$ as a function of z, assuming $t = 1$)



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

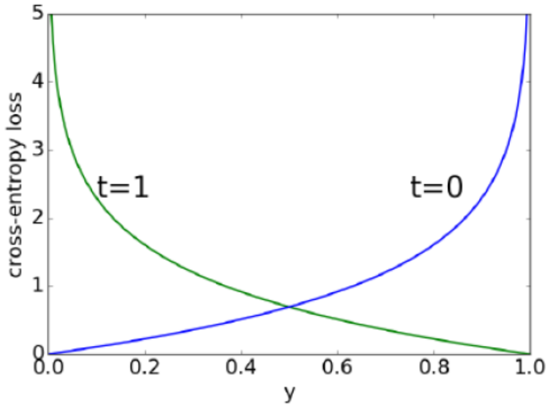
- 由前面 $\sigma(z)$ 的函数图像可知：For $z \ll 0, \sigma(z) \approx 0$.
- 所以 $\frac{\partial \mathcal{L}}{\partial z} \approx 0 \implies \frac{\partial \mathcal{L}}{\partial w_j} \approx 0 \implies$ derivative w.r.t. (with respect to) w_j is small $\implies w_j$ is like a critical point
- 因此，用该 function，可能使模型向负轴方向优化，从而不能到达真正的 critical point

Logistic Regression

Cross-entropy loss

Cross-entropy loss (aka log loss) captures this intuition:

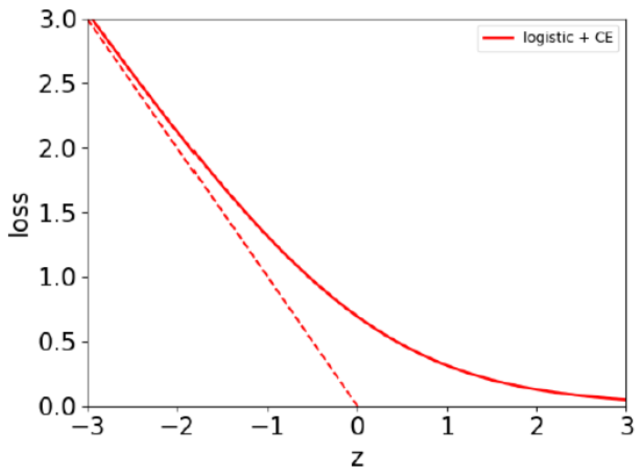
$$\mathcal{L}_{\mathrm{CE}}(y,t) = \begin{cases} -\log y & \text{if } t=1 \\ -\log(1-y) & \text{if } t=0 \end{cases}$$
$$= -t\log y - (1-t)\log(1-y)$$



注：交叉熵 loss 的机制是这样的：如果对于一个 sample，预测的是 1（一般还会包含一个置信度，比如有 90% 的把握认为是 1），但实际结果是 0，这时 cross-entropy loss 会对这个结果进行惩罚（一般，错误预测的置信度越大，惩罚越大），即让 loss 变得很大（见上图蓝线）。

Logistic Regression

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



Plot is for target $t=1$.

Gradient Descent for Logistic Regression

由于 logistic loss 是一个凸函数（**convex function**，如上图，曲线上方空间是凸出来的函数），所以我们可以用 gradient descent 来对其进行优化。

Gradient of Logistic Loss

我们首先将模型的权重进行合理的初始化，这里权重为 w，我们将 w 初始化为 0。

Back to logistic regression:

$$\mathcal{L}_{\mathrm{CE}}(y,t) = -t\log(y) - (1-t)\log(1-y)$$
$$y = 1/(1+e^{-z}) \quad \text{and} \quad z = \mathbf{w}^\top \mathbf{x}$$

Therefore

$$\frac{\partial \mathcal{L}_{\mathrm{CE}}}{\partial w_j} = \frac{\partial \mathcal{L}_{\mathrm{CE}}}{\partial y} \cdot \frac{\partial y}{\partial z} \cdot \frac{\partial z}{\partial w_j} = \left(-\frac{t}{y} + \frac{1-t}{1-y}\right) \cdot y(1-y) \cdot x_j$$
$$= (y-t)x_j$$

(verify this)

Gradient descent (coordinatewise) update to find the weights of logistic regression:

$$w_j \leftarrow w_j - \alpha \frac{\partial \mathcal{J}}{\partial w_j}$$
$$= w_j - \frac{\alpha}{N}\sum_{i=1}^{N}(y^{(i)} - t^{(i)})\, x_j^{(i)}$$

Multiclass Classification and Softmax Regression

One-hot Encoding

Targets form a discrete set $\{1, \ldots, K\}$.

It's often more convenient to represent them as one-hot vectors, or a one-of-K encoding:

$$\mathbf{t} = \underbrace{(0, \ldots, 0, 1, 0, \ldots, 0)}_{\text{entry } k \text{ is } 1} \in \mathbb{R}^K$$

Multiclass Linear Classification

分类任务经常不止一类，比如手写数字分类有 10 类。

We can start with a linear function of the inputs.

Now there are D input dimensions and K output dimensions, so we need $K \times D$ weights, which we arrange as a weight matrix $\mathbf{W}$.

Also, we have a K-dimensional vector $\mathbf{b}$ of biases.

注：X 为 D 维矩阵，W 为 K × D 维矩阵，b 为 K 维矩阵，这样 WX + b 的结果为 K 维矩阵。

A linear function of the inputs:

$$z_k = \sum_{j=1}^{D} w_{kj} x_j + b_k \quad \text{for} \quad k = 1, 2, ..., K$$

$$y_i = \begin{cases} 1 & i = \arg\max_k z_k \\ 0 & \text{otherwise} \end{cases}$$

注：上式和 logistic classification 不一样。logistic classification 是之间算出 Wx 然后进行分类。而这里是先对每一类进行预测，z_k 的大小可以解释为模型倾向于将 k 预测结果的程度。

举个例子，假如做手写数字，一共有 10 类，输入 1 个有 D 个 feature 的 input。那么 w 就是 10 × D，对于每一类都有特殊的 D 个权重。然后模型要对 input 进行 10 次预测，取其中最大的作为结果（在 one-hot encoding 中将相应的位置设为 1，即上面第二个式子所做的）。

We can eliminate the bias $\mathbf{b}$ by taking $\mathbf{W} \in \mathbb{R}^{K \times (D+1)}$ and adding a dummy variable $x_0 = 1$. So, vectorized:

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b} \quad \text{or with dummy } x_0 = 1 \quad \mathbf{z} = \mathbf{W}\mathbf{x}$$

Softmax Regression

We want soft predictions that are like probabilities, i.e., $0 \leq y_k \leq 1$ and $\sum_k y_k = 1$.

A natural activation function to use is the softmax function, a multivariable generalization of the logistic function:

$$y_k = \text{softmax}(z_1, \ldots, z_K)_k = \frac{e^{z_k}}{\sum_{k'} e^{z_{k'}}}$$

- Outputs can be interpreted as probabilities (positive and sum to 1)
- If z_k is much larger than the others, then $\text{softmax}(\mathbf{z})_k \approx 1$ and it behaves like argmax.

注：上式中，z_k 代表 z 中第 k 个元素，$\sum_{k'} e^{z_{k'}}$ 代表对 z 中每一个元素进行 $e^{z_{k'}}$ 后的总和。

If a model outputs a vector of class probabilities, we can use cross-entropy as the loss function:

$$\begin{aligned}\mathcal{L}_{\text{CE}}(\mathbf{y}, \mathbf{t}) &= -\sum_{k=1}^{K} t_k \log y_k \\ &= -\mathbf{t}^\top (\log \mathbf{y}),\end{aligned}$$

where the log is applied elementwise.

上式叫 softmax-cross-entropy function。

Gradient Descent

Softmax regression (with dummy $x_0 = 1$):

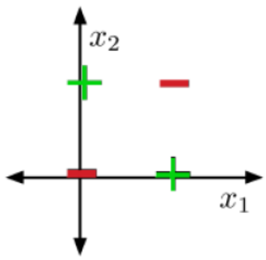
$$\begin{aligned}\mathbf{z} &= \mathbf{W}\mathbf{x} \\ \mathbf{y} &= \text{softmax}(\mathbf{z}) \\ \mathcal{L}_{\text{CE}} &= -\mathbf{t}^\top (\log \mathbf{y})\end{aligned}$$

Gradient descent updates can be derived for each row of $\mathbf{W}$:

$$\begin{aligned}\frac{\partial \mathcal{L}_{\text{CE}}}{\partial \mathbf{w}_k} &= \frac{\partial \mathcal{L}_{\text{CE}}}{\partial z_k} \cdot \frac{\partial z_k}{\partial \mathbf{w}_k} = (y_k - t_k) \cdot \mathbf{x} \\ \mathbf{w}_k &\leftarrow \mathbf{w}_k - \alpha \frac{1}{N} \sum_{i=1}^{N} (y_k^{(i)} - t_k^{(i)}) \mathbf{x}^{(i)}\end{aligned}$$

Limits of Linear Classification

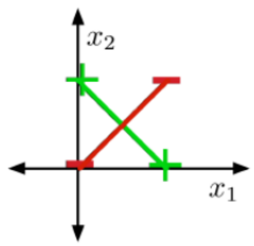
Some datasets are not linearly separable, e.g. **XOR**



Showing that XOR is not linearly separable

通过 contradiction（反证法）来证明：

- If two points lie in a half-space, line segment connecting them also lie in the same halfspace.
- Suppose there were some feasible weights (hypothesis). If the positive examples are in the positive half-space, then the green line segment must be as well.
- Similarly, the red line segment must line within the negative half-space.



- But the intersection can't lie in both half-spaces. Contradiction!

Overcome

- Sometimes we can overcome this limitation using feature maps, just like for linear regression. E.g., for **XOR**:

$$\boldsymbol{\psi}(\mathbf{x}) = \begin{pmatrix} x_1 \\ x_2 \\ x_1 x_2 \end{pmatrix}$$

x_1	x_2	$\psi_1(\mathbf{x})$	$\psi_2(\mathbf{x})$	$\psi_3(\mathbf{x})$	t
0	0	0	0	0	0
0	1	0	1	0	1
1	0	1	0	0	1
1	1	1	1	1	0

- This is linearly separable. (Try it!)