Logistic Regression

Nipun Batra

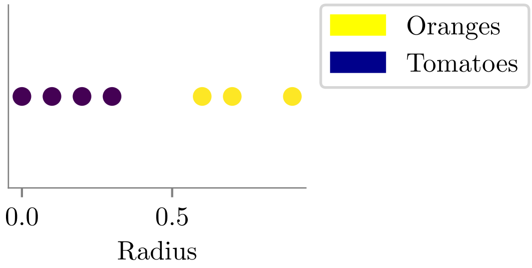
IIT Gandhinagar

September 22, 2025

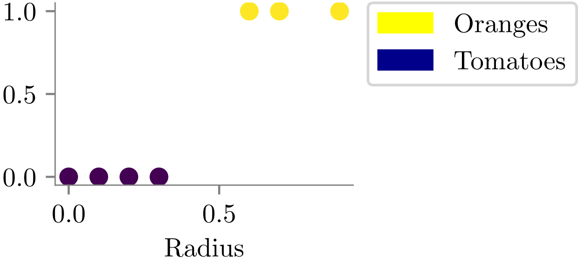
Table of Contents

Problem Setup

Classification Technique



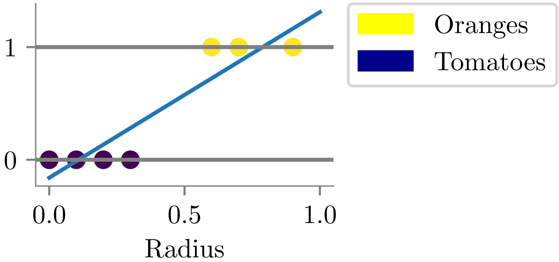
Classification Technique



Aim: Probability(Tomatoes | Radius) ? or

$\boxed{\text{More generally, P}(y = 1|\mathbf{X} = \mathbf{x})\text{?}}$

Idea: Use Linear Regression



$$P(X = Orange|Radius) = \theta_0 + \theta_1 \times Radius$$

Generally,

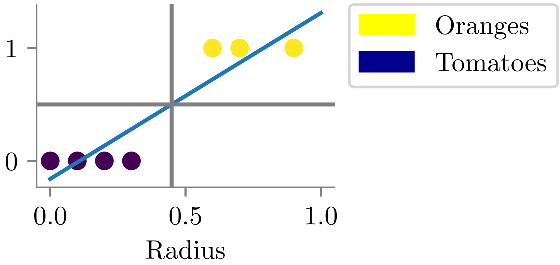
$$P(y = 1|\mathbf{x}) = \mathbf{X}\boldsymbol{\theta}$$

Idea: Use Linear Regression

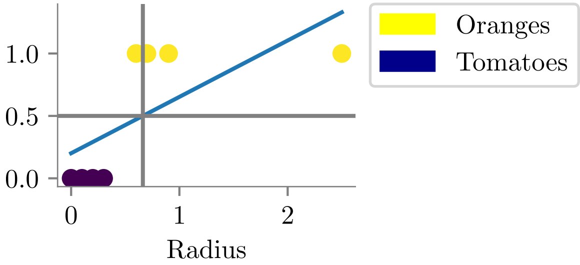
Prediction:
If $\theta_0 + \theta_1 \times \mathit{Radius} > 0.5 \rightarrow$ Orange
Else $\rightarrow$ Tomato

Problem:
Range of $\mathbf{X}\boldsymbol{\theta}$ is $(-\infty, \infty)$
But $P(y = 1|\ldots) \in [0, 1]$

Idea: Use Linear Regression

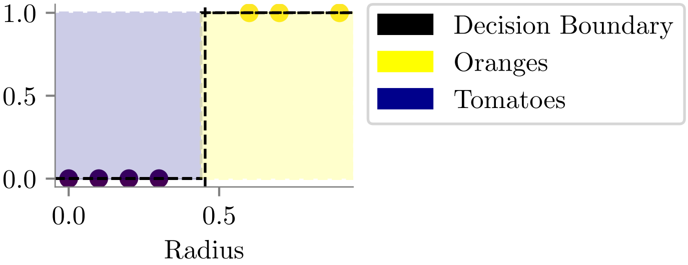


Idea: Use Linear Regression



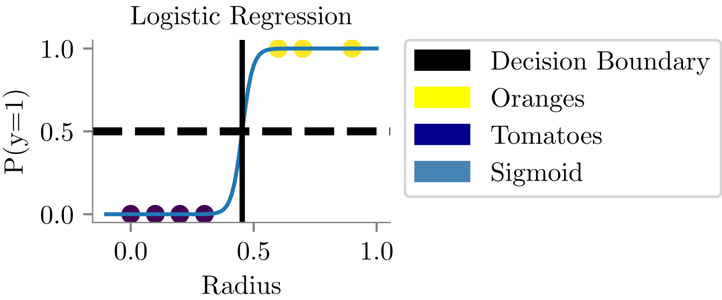
Linear regression for classification gives a poor prediction!

Ideal boundary



- Have a decision function similar to the above (but not so sharp and discontinuous)
- Aim: use linear regression still!

Idea: Use Linear Regression



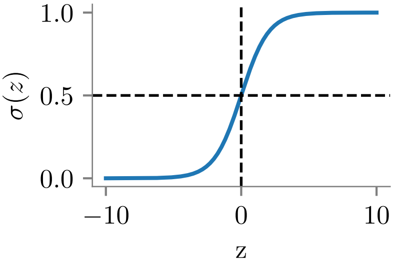
Question. Can we still use Linear Regression?
Answer. Yes! Transform $\hat{y} \rightarrow [0, 1]$

Logistic/Sigmoid function

Logistic / Sigmoid Function

$\hat{y} \in (-\infty, \infty)$
$\phi =$ Sigmoid / Logistic Function (σ)
$\phi(\hat{y}) \in [0, 1]$

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



Logistic / Sigmoid Function

$$z \to \infty$$
$$\sigma(z) \to 1$$
$$z \to -\infty$$
$$\sigma(z) \to 0$$
$$z = 0$$
$$\sigma(z) = 0.5$$

Logistic / Sigmoid Function

Question. Could you use some other transformation (ϕ) of $\hat{y}$ s.t.

$$\phi(\hat{y}) \in [0, 1]$$

Yes! But Logistic Regression works.

Logistic / Sigmoid Function

$$P(y = 1|\mathbf{X}) = \sigma(\mathbf{X}\boldsymbol{\theta}) = \frac{1}{1 + e^{-\mathbf{X}\boldsymbol{\theta}}}$$

Q. Write $\mathbf{X}\boldsymbol{\theta}$ in a more convenient form (as $P(y = 1|X)$, $P(y = 0|X)$)

Logistic / Sigmoid Function

$$P(y=1|\mathbf{X}) = \sigma(\mathbf{X}\boldsymbol{\theta}) = \frac{1}{1+e^{-\mathbf{X}\boldsymbol{\theta}}}$$

Q. Write $\mathbf{X}\boldsymbol{\theta}$ in a more convenient form (as $P(y=1|X)$, $P(y=0|X)$)

$$P(y=0|X) = 1 - P(y=1|X) = 1 - \frac{1}{1+e^{-\mathbf{X}\boldsymbol{\theta}}} = \frac{e^{-\mathbf{X}\boldsymbol{\theta}}}{1+e^{-\mathbf{X}\boldsymbol{\theta}}}$$

$$\therefore \frac{P(y=1|X)}{1-P(y=1|X)} = e^{\mathbf{X}\boldsymbol{\theta}} \implies \mathbf{X}\boldsymbol{\theta} = \log\frac{P(y=1|X)}{1-P(y=1|X)}$$

Odds (Used in betting)

$$\frac{P(win)}{P(loss)}$$

Here,

$$Odds = \frac{P(y=1)}{P(y=0)}$$

$$\boxed{\text{log-odds} = \log \frac{P(y=1)}{P(y=0)} = \mathbf{X}\boldsymbol{\theta}}$$

Logistic Regression

Q. What is decision boundary for Logistic Regression?

Logistic Regression

Q. What is decision boundary for Logistic Regression?

Decision Boundary: $P(y=1|X) = P(y=0|X)$

$$\text{or } \frac{1}{1+e^{-\mathbf{X}\boldsymbol{\theta}}} = \frac{e^{-\mathbf{X}\boldsymbol{\theta}}}{1+e^{-\mathbf{X}\boldsymbol{\theta}}}$$

$$\text{or } e^{\mathbf{X}\boldsymbol{\theta}} = 1$$

$$\text{or } \mathbf{X}\boldsymbol{\theta} = 0$$

Learning Parameters

Could we use cost function as:

$$J(\theta) = \sum (y_i - \hat{y}_i)^2$$

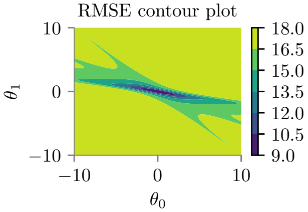
$$\hat{y}_i = \sigma(\mathbf{X}\boldsymbol{\theta})$$

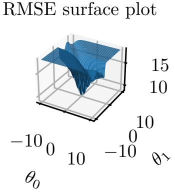
Answer: **No (Non-Convex)**
Why? Squared loss + sigmoid creates non-convex surface:

- Sigmoid $\sigma(z) = \frac{1}{1+e^{-z}}$ is non-linear
- Composition $(\sigma(\mathbf{X}\boldsymbol{\theta}) - y)^2$ has multiple local minima
- No guarantee gradient descent finds global optimum
- This is why we need cross-entropy loss instead!

Deriving Cost Function via Maximum Likelihood Estimation

Cost function convexity





Learning Parameters

Likelihood = $P(D|\theta)$

$P(y|X,\theta) = \prod_{i=1}^{n} P(y_i|x_i,\theta)$
where y = 0 or 1

Learning Parameters

Likelihood $= P(D|\theta)$

$$P(y|X,\theta) = \prod_{i=1}^{n} P(y_i|x_i,\theta) = \prod_{i=1}^{n} \left\{ \frac{1}{1+e^{-x_i^T\theta}} \right\}^{y_i} \left\{ 1 - \frac{1}{1+e^{-x_i^T\theta}} \right\}^{1-y_i}$$

[Above: Similar to $P(D|\theta)$ for Linear Regression;

Difference Bernoulli instead of Gaussian]

$-\log P(y|\mathbf{X},\boldsymbol{\theta})$ = Negative Log Likelihood = Cost function will be minir

Aside on Bernoulli Likelihood

- Assume you have a coin and flip it ten times and get (H, H, T, T, T, H, H, T, T, T).
- What is p(H)?
- We might think it to be: $4/10 = 0.4$. But why?
- Answer 1: Probability defined as a measure of long running frequencies
- Answer 2: What is likelihood of seeing the above sequence when the p(Head)$=\theta$?
- Idea find MLE estimate for θ

Aside on Bernoulli Likelihood

- $p(H) = \theta$ and $p(T) = 1 - \theta$
- What is the PMF for first observation $P(D_1 = x|\theta)$, where x = 0 for Tails and x = 1 for Heads?
- $P(D_1 = x|\theta) = \theta^x(1-\theta)^{(1-x)}$
- Verify the above: if x = 0 (Tails), $P(D_1 = x|\theta) = 1 - \theta$ and if x = 1 (Heads), $P(D_1 = x|\theta) = \theta$
- What is $P(D_1, D_2, ..., D_n|\theta)$?
- $P(D_1, D_2, ..., D_n|\theta) = P(D_1|\theta)P(D_2|\theta)...P(D_n|\theta)$
- $P(D_1, D_2, ..., D_n|\theta) = \theta^{n_h}(1-\theta)^{n_t}$
- Log-likelihood $= \mathcal{LL}(\theta) = n_h \log(\theta) + n_t \log(1-\theta)$
- $\frac{\partial \mathcal{LL}(\theta)}{\partial \theta} = 0 \implies \frac{n_h}{\theta} + \frac{n_t}{1-\theta} = 0 \implies \theta_{MLE} = \frac{n_h}{n_h + n_t}$

Cross Entropy Cost Function

Learning Parameters

$$J(\theta) = -\log \left\{ \prod_{i=1}^{n} \left\{ \frac{1}{1 + e^{-x_i^T \theta}} \right\}^{y_i} \left\{ 1 - \frac{1}{1 + e^{-x_i^T \theta}} \right\}^{1-y_i} \right\}$$

$$J(\theta) = -\left\{ \sum_{i=1}^{N} y_i \log(\sigma_\theta(x_i)) + (1 - y_i) \log(1 - \sigma_\theta(x_i)) \right\}$$

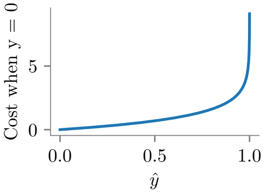
This cost function is called cross-entropy.
Why?

Interpretation of Cross-Entropy Cost Function

What is the interpretation of the cost function?
Let us try to write the cost function for a single example:

$$J(\theta) = -y_i \log \hat{y}_i - (1 - y_i) \log(1 - \hat{y}_i)$$

First, assume y_i is 0, then if $\hat{y}_i$ is 0, the loss is 0; but, if $\hat{y}_i$ is 1, the loss tends towards infinity!



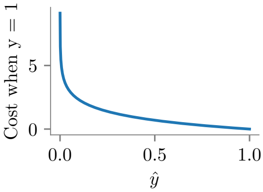
Notebook: logits-usage

Interpretation of Cross-Entropy Cost Function

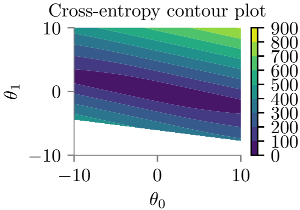
What is the interpretation of the cost function?

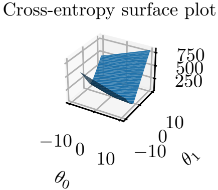
$$J(\theta) = -y_i \log \hat{y}_i - (1 - y_i) \log(1 - \hat{y}_i)$$

Now, assume y_i is 1, then if $\hat{y}_i$ is 0, the loss is huge; but, if $\hat{y}_i$ is 1, the loss is zero!



Cost function convexity





Learning Parameters

$$\frac{\partial J(\theta)}{\partial \theta_j} = -\frac{\partial}{\partial \theta_j}\left\{ \sum_{i=1}^{N} y_i log(\sigma_\theta(x_i)) + (1 - y_i) log(1 - \sigma_\theta(x_i)) \right\}$$

$$= -\sum_{i=1}^{N} \left[y_i \frac{\partial}{\partial \theta_j} \log(\sigma_\theta(x_i)) + (1 - y_i) \frac{\partial}{\partial \theta_j} log(1 - \sigma_\theta(x_i)) \right]$$

Learning Parameters

$$\frac{\partial J(\theta)}{\partial \theta_j} = -\sum_{i=1}^{N}\left[y_i \frac{\partial}{\partial \theta_j}\log(\sigma_\theta(x_i)) + (1-y_i)\frac{\partial}{\partial \theta_j} log(1-\sigma_\theta(x_i))\right]$$

$$= -\sum_{i=1}^{N}\left[\frac{y_i}{\sigma_\theta(x_i)}\frac{\partial}{\partial \theta_j}\sigma_\theta(x_i) + \frac{1-y_i}{1-\sigma_\theta(x_i)}\frac{\partial}{\partial \theta_j}(1-\sigma_\theta(x_i))\right]$$

Aside:

$$\frac{\partial}{\partial z}\sigma(z) = \frac{\partial}{\partial z}\frac{1}{1+e^{-z}} = -(1+e^{-z})^{-2}\frac{\partial}{\partial z}(1+e^{-z})$$

$$= \frac{e^{-z}}{(1+e^{-z})^2} = \left(\frac{1}{1+e^{-z}}\right)\left(\frac{e^{-z}}{1+e^{-z}}\right) = \sigma(z)\left\{\frac{1+e^{-z}}{1+e^{-z}} - \frac{1}{1+e^{-z}}\right\}$$

$$= \sigma(z)(1-\sigma(z))$$

Learning Parameters

Resuming from (1)

$$\frac{\partial J(\theta)}{\partial \theta_j} = -\sum_{i=1}^{N}\left[\frac{y_i}{\sigma_\theta(x_i)}\frac{\partial}{\partial \theta_j}\sigma_\theta(x_i) + \frac{1-y_i}{1-\sigma_\theta(x_i)}\frac{\partial}{\partial \theta_j}(1-\sigma_\theta(x_i))\right]$$

$$= -\sum_{i=1}^{N}\left[\frac{y_i\sigma_\theta(x_i)}{\sigma_\theta(x_i)}(1-\sigma_\theta(x_i))\frac{\partial}{\partial \theta_j}(x_i\theta) + \frac{1-y_i}{1-\sigma_\theta(x_i)}(1-\sigma_\theta(x_i))\frac{\partial}{\partial \theta_j}(1-\sigma_\theta(x_i))\right]$$

$$= -\sum_{i=1}^{N}\left[y_i(1-\sigma_\theta(x_i))x_i^j - (1-y_i)\sigma_\theta(x_i)x_i^j\right]$$

$$= -\sum_{i=1}^{N}\left[(y_i - y_i\sigma_\theta(x_i) - \sigma_\theta(x_i) + y_i\sigma_\theta(x_i))x_i^j\right]$$

$$= \sum_{i=1}^{N}\left[\sigma_\theta(x_i) - y_i\right]x_i^j$$

Learning Parameters

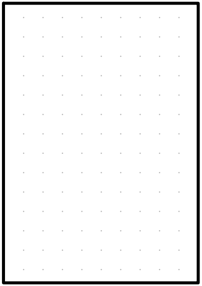
$$\boxed{\frac{\partial J(\theta)}{\theta_j} = \sum_{i=1}^{N} \left[\sigma_\theta(x_i) - y_i\right] x_i^j}$$

Now, just use Gradient Descent!

$$\frac{\partial J(\theta)}{\partial \theta_j} = \sum_{i=1}^{N} (\hat{y}_i - y_i) x_i^j$$

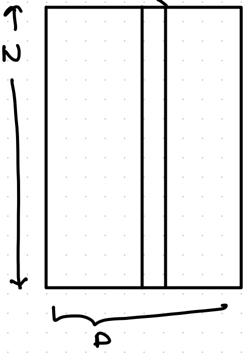
$$\frac{\partial J(\theta)}{\partial \theta_j} = \sum_{i=1}^{N} (\hat{y}_i - y_i) x_i^j$$

MATRIX X



$$\frac{\partial J(\theta)}{\partial \theta_j} = \sum_{i=1}^{N} (\hat{y}_i - y_i) x_i^j$$

MATRIX X

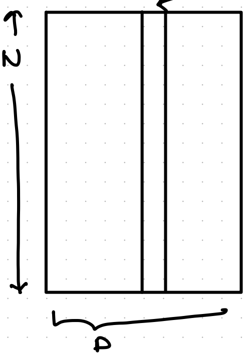


j^{th} column of X

$= X^j$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \sum_{i=1}^{N} (\hat{y}_i - y_i)\, x_i^j$$

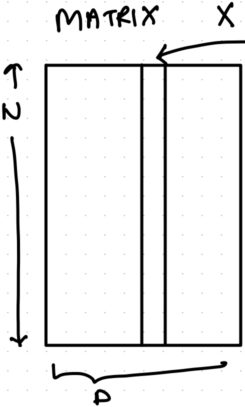
MATRIX X



j^{th} column of X

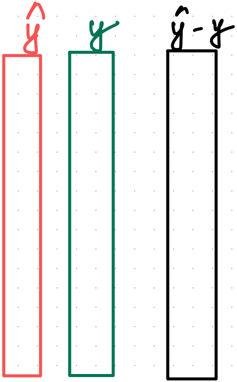
$= X^j$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \sum_{i=1}^{N} (\hat{y}_i - y_i) x_i^j$$

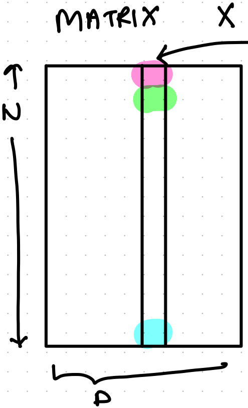


j^{th} column of X

$= X^j$

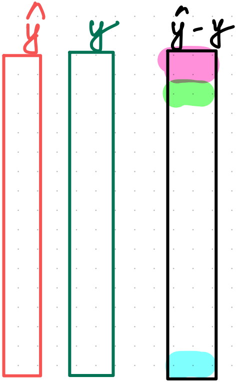


$$\frac{\partial J(\theta)}{\partial \theta_j} = \sum_{i=1}^{N} (\hat{y}_i - y_i)\, x_i^j$$



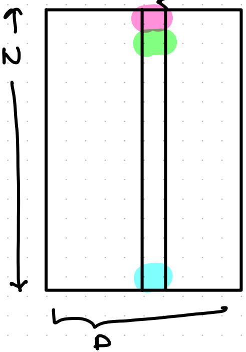
j^{th} column of X

$= X^j$



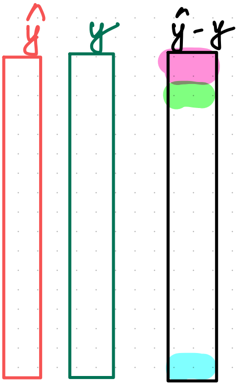
$$\frac{\partial J(\theta)}{\partial \theta_j} = \sum_{i=1}^{N} (\hat{y}_i - y_i)\, x_i^j = \underset{1\times N}{{x^j}^T} (\hat{y} - y)$$

MATRIX X



j^{th} column of X

$= X^j$



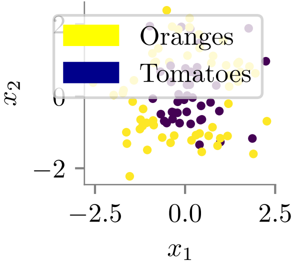
$$\frac{\partial J(\theta)}{\partial \theta_j} = \sum_{i=1}^{N} (\hat{y}_i - y_i) x_i^j = \underset{1\times N}{x^{j^T}} (\hat{y} - y)$$

$$\begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta_1} \\ \frac{\partial J(\theta)}{\partial \theta_2} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_D} \end{bmatrix} = \begin{pmatrix} x^{1^T} (\hat{y} - y) \\ x^{2^T} (\hat{y} - y) \\ \vdots \\ x^{D^T} (\hat{y} - y) \end{pmatrix}$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \sum_{i=1}^{N} (\hat{y}_i - y_i) x_i^j = \underset{1\times N}{x^{j^T}} (\hat{y} - y)$$

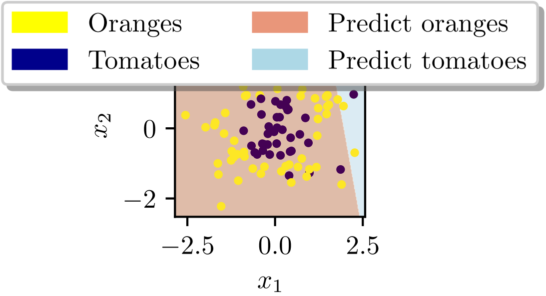
$$\begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta_1} \\ \frac{\partial J(\theta)}{\partial \theta_2} \\ \vdots \\ \\ \frac{\partial J(\theta)}{\partial \theta_D} \end{bmatrix} = \begin{pmatrix} x^{1^T} (\hat{y} - y) \\ x^{2^T} (\hat{y} - y) \\ \vdots \\ \\ x^{D^T} (\hat{y} - y) \end{pmatrix} = X^T (\hat{y} - y)$$

Logistic Regression with feature transformation



What happens if you apply logistic regression on the above data?

Logistic Regression with feature transformation

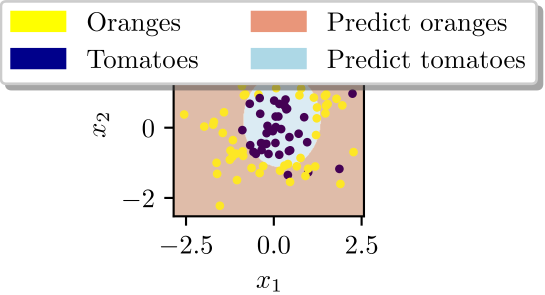


Linear boundary will not be accurate here. What is the technical name of the problem? Bias!

Logistic Regression with feature transformation

$$\phi(x) = \begin{bmatrix} \phi_0(x) \\ \phi_1(x) \\ \vdots \\ \phi_{K-1}(x) \end{bmatrix} = \begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \\ \vdots \\ x^{K-1} \end{bmatrix} \in \mathbb{R}^K$$

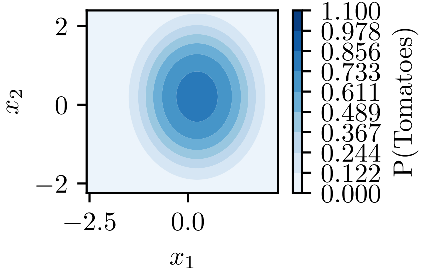
Logistic Regression with feature transformation



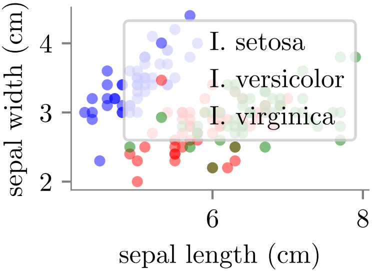
Using x_1^2, x_2^2 as additional features, we are able to learn a more accurate classifier.

Logistic Regression with feature transformation

How would you expect the probability contours look like?

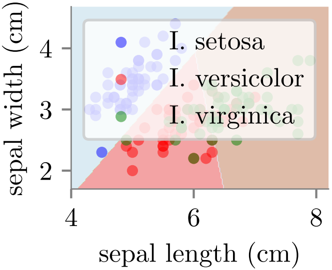


Multi-Class Prediction



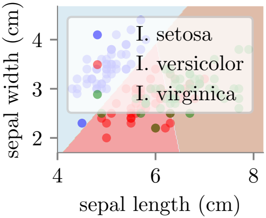
How would you learn a classifier? Or, how would you expect the classifier to learn decision boundaries?

Multi-Class Prediction



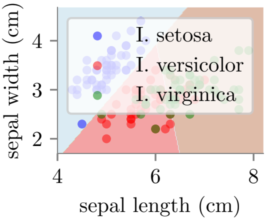
1. Use one-vs.-all on Binary Logistic Regression
2. Use one-vs.-one on Binary Logistic Regression
3. Extend Binary Logistic Regression to Multi-Class Logistic Regression

Multi-Class Prediction



1. Learn P(setosa (class 1)) $= \mathcal{F}(\mathbf{X}\boldsymbol{\theta}_1)$
2. P(versicolor (class 2)) $= \mathcal{F}(\mathbf{X}\boldsymbol{\theta}_2)$
3. P(virginica (class 3)) $= \mathcal{F}(\mathbf{X}\boldsymbol{\theta}_3)$
4. Goal: Learn $\theta_i \forall i \in \{1, 2, 3\}$
5. Question: What could be an $\mathcal{F}$?

Multi-Class Prediction



1. Question: What could be an $\mathcal{F}$?
2. Property: $\sum_{i=1}^{3} \mathcal{F}(\mathbf{X}\boldsymbol{\theta}_i) = 1$
3. Also $\mathcal{F}(z) \in [0, 1]$
4. Also, $\mathcal{F}(z)$ has squashing proprties: $R \mapsto [0, 1]$

Softmax

$$Z \in \mathbb{R}^d$$

$$\mathcal{F}(z_i) = \frac{e^{z_i}}{\sum_{i=1}^{d} e^{z_i}}$$

$$\therefore \sum \mathcal{F}(z_i) = 1$$

$\mathcal{F}(z_i)$ refers to <u>probability</u> of class <u>i</u>

Softmax for Multi-Class Logistic Regression

$$k = \{1, \ldots, k\} \text{classes}$$

$$\theta = \begin{bmatrix} \vdots\vdots\vdots\vdots \\ \theta_1 \theta_2 \cdots \theta_k \\ \vdots\vdots\vdots\vdots \end{bmatrix}$$

$$\boxed{P(y = k | X, \theta) = \frac{e^{\mathbf{X}\boldsymbol{\theta}_k}}{\sum_{k=1}^{K} e^{\mathbf{X}\boldsymbol{\theta}_k}}}$$

Softmax for Multi-Class Logistic Regression

For K = 2 classes,

$$P(y = k|X, \theta) = \frac{e^{\mathbf{X}\boldsymbol{\theta}_k}}{\sum_{k=1}^{K} e^{\mathbf{X}\boldsymbol{\theta}_k}}$$

$$P(y = 0|X, \theta) = \frac{e^{\mathbf{X}\boldsymbol{\theta}_0}}{e^{\mathbf{X}\boldsymbol{\theta}_0} + e^{\mathbf{X}\boldsymbol{\theta}_1}}$$

$$P(y = 1|X, \theta) = \frac{e^{\mathbf{X}\boldsymbol{\theta}_1}}{e^{\mathbf{X}\boldsymbol{\theta}_0} + e^{\mathbf{X}\boldsymbol{\theta}_1}} = \frac{e^{\mathbf{X}\boldsymbol{\theta}_1}}{e^{\mathbf{X}\boldsymbol{\theta}_1}\{1 + e^{X(\theta_0 - \theta_1)}\}}$$

$$= \frac{1}{1 + e^{-\mathbf{X}\boldsymbol{\theta}'}}$$

$$= \text{Sigmoid!}$$

Multi-Class Logistic Regression Cost

Assume our prediction and ground truth for the three classes for i^{th} point is:

$$\hat{y}_i = \begin{bmatrix} 0.1 \\ 0.8 \\ 0.1 \end{bmatrix} = \begin{bmatrix} \hat{y}_i^1 \\ \hat{y}_i^2 \\ \hat{y}_i^3 \end{bmatrix}$$

$$y_i = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} y_i^1 \\ y_i^2 \\ y_i^3 \end{bmatrix}$$

meaning the true class is Class $\#2$
Let us calculate $-\sum_{k=1}^{3} y_i^k \log \hat{y}_i^k$
$= -(0 \times \log(0.1) + 1 \times \log(0.8) + 0 \times \log(0.1))$
Tends to zero

Multi-Class Logistic Regression Cost

Assume our prediction and ground truth for the three classes for i^{th} point is:

$$\hat{y}_i = \begin{bmatrix} 0.3 \\ 0.4 \\ 0.3 \end{bmatrix} = \begin{bmatrix} \hat{y}_i^1 \\ \hat{y}_i^2 \\ \hat{y}_i^3 \end{bmatrix}$$

$$y_i = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} y_i^1 \\ y_i^2 \\ y_i^3 \end{bmatrix}$$

meaning the true class is Class $\#2$
Let us calculate $-\sum_{k=1}^{3} y_i^k \log \hat{y}_i^k$
$= -(0 \times \log(0.1) + 1 \times \log(0.4) + 0 \times \log(0.1))$
High number! Huge penalty for misclassification!

Multi-Class Logistic Regression Cost

For 2 class we had:

$$J(\theta) = -\left\{ \sum_{i=1}^{N} y_i \log(\sigma_\theta(x_i)) + (1 - y_i) \log(1 - \sigma_\theta(x_i)) \right\}$$

More generally,

$$J(\theta) = -\left\{ \sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right\}$$

$$J(\theta) = -\left\{ \sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right\}$$

Extend to K-class:

$$J(\theta) = -\left\{ \sum_{i=1}^{N} \sum_{k=1}^{K} y_i^k \log(\hat{y}_i^k) \right\}$$

Multi-Class Logistic Regression Cost

Now:

$$\frac{\partial J(\theta)}{\partial \theta_k} = \sum_{i=1}^{N} \left[x_i \left\{ I(y_i = k) - P(y_i = k | x_i, \theta) \right\} \right]$$

Hessian Matrix

The Hessian matrix of f(.) with respect to θ, written $\nabla_\theta^2 f(\theta)$ or simply as $\mathbb{H}$, is the $d \times d$ matrix of partial derivatives,

$$\nabla_\theta^2 f(\theta) = \begin{bmatrix} \frac{\partial^2 f(\theta)}{\partial \theta_1^2} & \frac{\partial^2 f(\theta)}{\partial \theta_1 \partial \theta_2} & \cdots & \frac{\partial^2 f(\theta)}{\partial \theta_1 \partial \theta_n} \\ \frac{\partial^2 f(\theta)}{\partial \theta_2 \partial \theta_1} & \frac{\partial^2 f(\theta)}{\partial \theta_2^2} & \cdots & \frac{\partial^2 f(\theta)}{\partial \theta_2 \partial \theta_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(\theta)}{\partial \theta_n \partial \theta_1} & \frac{\partial^2 f(\theta)}{\partial \theta_n \partial \theta_2} & \cdots & \frac{\partial^2 f(\theta)}{\partial \theta_n^2} \end{bmatrix}$$

Newton's Algorithm

The most basic second-order optimization algorithm is Newton's algorithm, which consists of updates of the form,

$$\theta_{k+1} = \theta_k - \mathbb{H}_k^{-1} g_k$$

where g_k is the gradient at step k. This algorithm is derived by making a second-order Taylor series approximation of $f(\theta)$ around θ_k:

$$f_{quad}(\theta) = f(\theta_k) + g_k^T(\theta - \theta_k) + \frac{1}{2}(\theta - \theta_k)^T \mathbb{H}_k(\theta - \theta_k)$$

differentiating and equating to zero to solve for θ_{k+1}.

Learning Parameters

Now assume:

$$g(\theta) = \sum_{i=1}^{N} \Big[\sigma_\theta(x_i) - y_i\Big] x_i^j = \mathbf{X}^\top(\sigma_\theta(X) - y)$$

$$\pi_i = \sigma_\theta(x_i)$$

Let $\mathbb{H}$ represent the Hessian of $J(\theta)$

$$\mathbb{H} = \frac{\partial}{\partial\theta} g(\theta) = \frac{\partial}{\partial\theta} \sum_{i=1}^{N} \Big[\sigma_\theta(x_i) - y_i\Big] x_i^j$$

$$= \sum_{i=1}^{N} \left[\frac{\partial}{\partial\theta}\sigma_\theta(x_i) x_i^j - \frac{\partial}{\partial\theta} y_i x_i^j\right] = \sum_{i=1}^{N} \sigma_\theta(x_i)(1 - \sigma_\theta(x_i)) x_i x_i^T$$

$$= \mathbf{X}^\top \mathrm{diag}(\sigma_\theta(x_i)(1 - \sigma_\theta(x_i)))\mathbf{X}$$

Iteratively reweighted least squares (IRLS)

For binary logistic regression, recall that the gradient and Hessian of the negative log-likelihood are given by:

$$g(\theta)_k = \mathbf{X}^\top(\pi_k - y)$$

$$\mathbf{H}_k = \mathbf{X}^\top S_k \mathbf{X}$$

$$\mathbf{S}_k = diag(\pi_{1k}(1-\pi_{1k}), \ldots, \pi_{nk}(1-\pi_{nk}))$$

$$\pi_{ik} = sigm(\mathbf{x_i}\theta_\mathbf{k})$$

The Newton update at iteraion $k+1$ for this model is as follows:

$$\theta_{k+1} = \theta_k - \mathbb{H}^{-1} g_k = \theta_k + (X^T S_k X)^{-1} X^T (y - \pi_k)$$

$$= (X^T S_k X)^{-1}[(X^T S_k X)\theta_k + X^T(y - \pi_k)] = (X^T S_k X)^{-1} X^T [S_k \mathbf{X}\boldsymbol{\theta}_k + y - \pi_k]$$

Regularized Logistic Regression

Unregularised:

$$J_1(\theta) = -\left\{ \sum_{i=1}^{N} y_i \log(\sigma_\theta(x_i)) + (1 - y_i) \log(1 - \sigma_\theta(x_i)) \right\}$$

L2 Regularization:

$$J(\theta) = J_1(\theta) + \lambda \theta^T \theta$$

L1 Regularization:

$$J(\theta) = J_1(\theta) + \lambda |\theta|$$

Class Imbalance Handling

The Problem: Imbalanced Data

- **Class Imbalance**: When one class has significantly more samples than others
- **Examples**:
  - Medical diagnosis: 99% healthy, 1% disease
  - Fraud detection: 99.9% legitimate, 0.1% fraud
  - Email spam: 90% legitimate, 10% spam
- **Problem**: Standard logistic regression biased toward majority class
- **Naive approach fails**: Predicting all samples as majority class

Impact on Model Performance

With 99% class 0, 1% class 1:

- **Naive classifier**: Always predict class 0 $\rightarrow$ 99% accuracy!
- **But**: 0% recall for class 1 (complete failure)
- **Standard metrics misleading**:
  - Accuracy $=$ 99% (looks great, but useless)
  - Precision for class 1 $=$ undefined (no predictions)
  - Recall for class 1 $=$ 0% (misses all positive cases)
- **Need**: Better evaluation metrics and techniques

Solution 1: Weighted Loss Function

Modify the cost function to penalize minority class errors more:

$$J(\boldsymbol{\theta}) = -\sum_{i=1}^{N} w_i \left[y_i \log(\sigma(\boldsymbol{\theta}^\top \mathbf{x}_i)) + (1 - y_i) \log(1 - \sigma(\boldsymbol{\theta}^\top \mathbf{x}_i)) \right]$$

- **Class weights**: $w_i = w_0$ if $y_i = 0$, $w_i = w_1$ if $y_i = 1$
- **Common choice**: $w_1 = \frac{N_0}{N_1}$ (inverse frequency)
- **Effect**: Forces model to pay attention to minority class
- **Implementation**: Available in most ML libraries (sklearn: `class_weight='balanced'`)

Solution 2: Threshold Adjustment

- **Standard**: Predict class 1 if $P(y = 1|\mathbf{x}) > 0.5$
- **Imbalanced**: Predict class 1 if $P(y = 1|\mathbf{x}) > \tau$ where $\tau < 0.5$
- **Threshold selection**:
  - Plot precision-recall curve or ROC curve
  - Choose τ that optimizes F1-score or business metric
  - Cross-validation to avoid overfitting
- **Trade-off**: Lower threshold $\rightarrow$ higher recall, lower precision

Solution 3: Resampling Techniques

Modify the training data distribution:

- **Undersampling**: Remove samples from majority class
  - Pro: Faster training, balanced classes
  - Con: Loss of information, smaller dataset
- **Oversampling**: Duplicate samples from minority class
  - Pro: No information loss
  - Con: Risk of overfitting, larger dataset
- **SMOTE**: Generate synthetic minority examples
  - Creates new samples between existing minority samples
  - More sophisticated than simple duplication

Evaluation Metrics for Imbalanced Data

- **Don't use accuracy alone!**
- **Precision**: $\frac{TP}{TP+FP}$ (of predicted positives, how many correct?)
- **Recall/Sensitivity**: $\frac{TP}{TP+FN}$ (of actual positives, how many found?)
- **F1-Score**: $\frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$ (harmonic mean)
- **ROC-AUC**: Area under ROC curve (threshold-independent)
- **PR-AUC**: Area under precision-recall curve (better for imbalanced data)

Practice and Review

Pop Quiz: Logistic Regression

1. Why can't we use linear regression for classification problems?
2. What is the key difference between sigmoid and softmax functions?
3. Why do we use cross-entropy loss instead of squared error?
4. How does regularization help in logistic regression?

Key Takeaways

- **Probabilistic Model**: Outputs probabilities via sigmoid function
- **Linear Decision Boundary**: Creates linear separation in feature space
- **Maximum Likelihood**: Optimized using gradient-based methods
- **Cross-Entropy Loss**: Appropriate for classification problems
- **No Closed Form**: Requires iterative optimization (gradient descent)
- **Regularization**: L1/L2 help prevent overfitting