Introduction to Logistic Regression

CHE 358 Mock Lecture

Tian Tian

22 June 2023

Perspective of Lecture

1. The categorical variables and the classification problem
2. Logistic regression vs linear regression
3. Fitting a logistic regression model
4. Case study: Li-ion battery failure
5. Beyond logistic regression: neural networks

Linear Regression Recap

1. Model Structure

   $y = f(x;\beta) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_d x_d + \epsilon \qquad \epsilon \sim N(0, \sigma^2)$

   x and y are **continuous** variables.

2. Assumptions

   - **Linear** relation between x and y
   - **Independent** observations
   - **Normal** distribution of error ϵ
   - **Constant** variance of error

3. Solving linear regression

   Least square fitting

   $$\hat{\beta} = \underset{\beta}{\operatorname{argmin}}\, S = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - f(x_i;\beta))^2$$

   where n is the number of samples

4. Interpretation

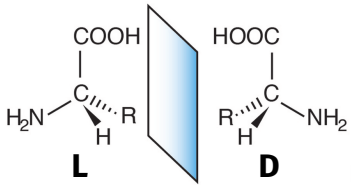
   Coefficients β: how much y change by one unit of x

   R^2: how well the model fits data

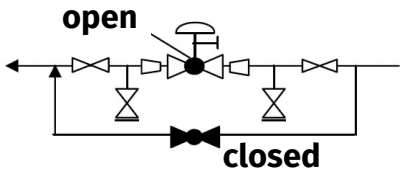
Categorical Response Variables

Many engineering problems involve outputs that can only assume a limited set of values, also known as **categorical** response variables."

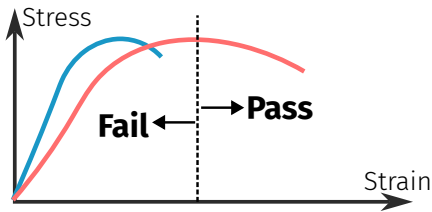
Chirality



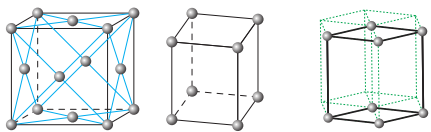
Process control



Material test



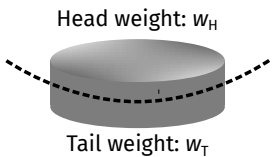
Lattice type



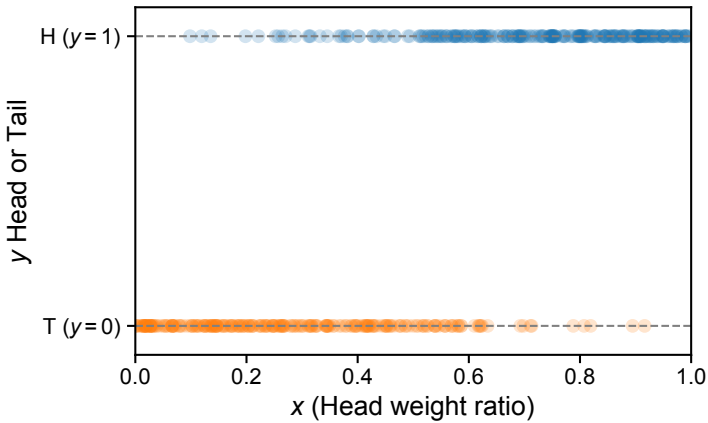
Cubic Triclinic Hexagonal

Thought Experiment: the Cheater's Coin

As a casino owner, if you wish to create a biased coin, how would you construct a model to fit your observations and measurements?



Head weight ratio
w_H / (w_T + w_H)



First Attempt: Linear Regression for Categorical Variables

Let's start by using linear regression to fit the **probability** that our biased coin lands heads up. The probability function to fit is:

$$y = p(x;\beta) = \beta_0 + \beta_1 x$$

Our dataset

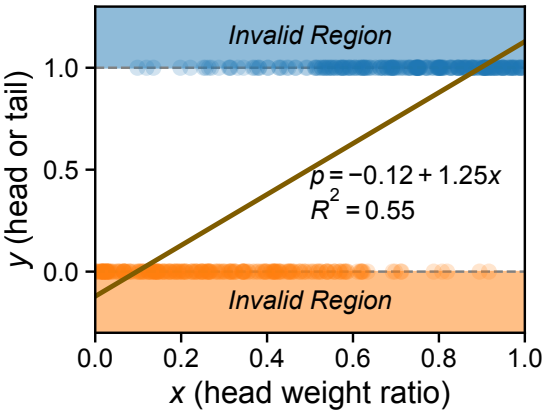
- Predictor (input) variable: $\mathrm{x} = \{x_1, x_2, \cdots, x_n\} \in \mathbb{R}^{(1,n)}$
- Response (predict) variable: $\mathrm{y} = \{y_1, y_2, \cdots, y_n\} \in \mathbb{R}^{(1,n)}$
- Parameter set: $\beta = \{\beta_0, \beta_1\} \in \mathbb{R}^{(1,2)}$

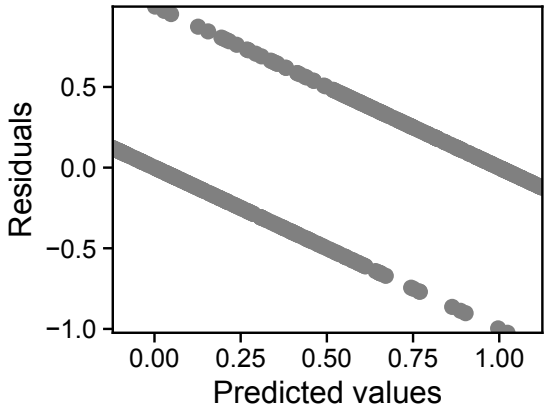
Is linear regression suitable for this problem?

- Range of predicted values?
- Distribution of residues?
- Are the predictions meaningful?
- Data extrapolation?

Linear Regression Results

Let's compare the linear regression and analyze the residuals.





Why Not Linear Regression?

As seen from previous example, linear regression is not suitable for predicting a dichotomous (binary) variable:

1. **Invalid predicted values**
   Probability should fall within the interval $(0, 1)$

2. **Assumptions of linear regression violated**
   Residuals are not independent from the predicted values, violating the assumption of homoscedasticity.

3. **Sensitive to data imbalance**
   If we have more "heavier-headed" coins sampled, the slope of linear regression will differ greatly.

4. **Ambiguous definition of order and distance**
   Categorical variables (usually) do not have a defined ordering or distance. Metrics like least square error may not be appropriate. (even worse for multi-class variables).

Second Attempt: Non-Linear Regression

Recall: for binary outcome $y_i \in \{0, 1\}$, $p(x_i; \beta)$ means the probability of that the outcome $y_i = 1$, given the input x_i and parameters β:

$$p(x_i; \beta) = \text{Prob}(y_i = 1 | x = x_i; \beta)$$

We could infer some properties of a non-linear p:

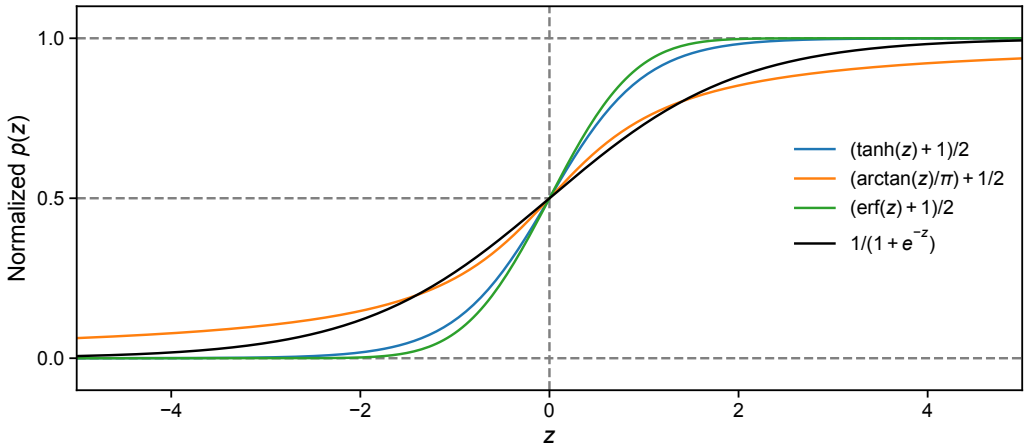
1. **Range**:
   p maps $x \in \mathbb{R}$ to $y \in (0, 1)$.
2. **Linear transformation**:
   We want p to be "linearizable", i.e. we can separate x and β to L.H.S via function inversion (with an analytical expression):

$$p^{-1} = \text{Inversion}(p) = \beta_0 + \beta_1 x_i$$

3. **Continuity**:
   To satisfy linearization, p should be continuous and monotonic in $\mathbb{R}$ (why?)

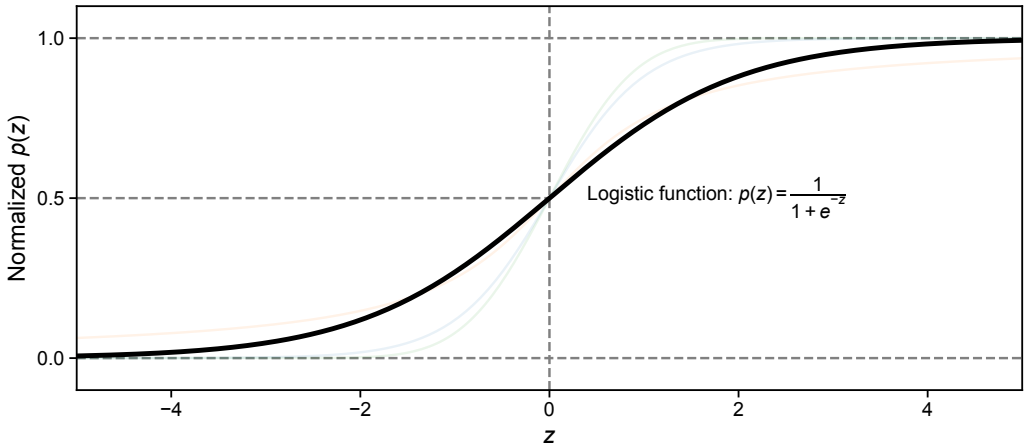
Candidates for Non-Linear Regression

A series of S-shaped (sigmoid) functions could potentially meet our criteria. Let's examine their plots in 1D.



Candidates for Non-Linear Regression

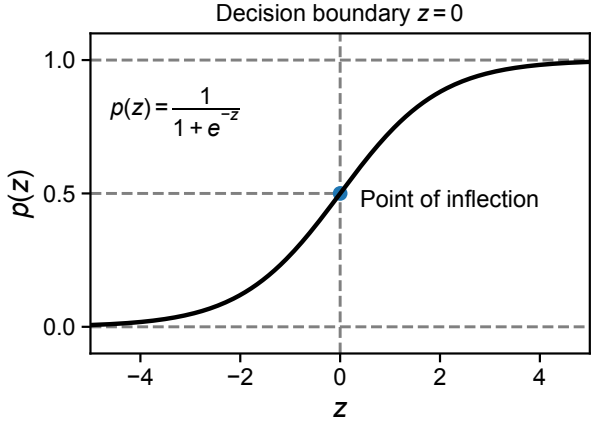
A series of S-shaped (sigmoid) functions could potentially meet our criteria. Let's examine their plots in 1D.



The Logistic Function: Properties

In 1D form, the sigmoid function $p(z) = 1/(1 + \exp(-z))$ is called the **logistic function** .
The process of fitting binary dataset is referred to as **logistic regression**.

1. Asymptotic behavior
   $\lim_{z\to\infty} p(z) = 1$
   $\lim_{z\to-\infty} p(z) = 0$
2. Symmetry
   $p(z) = 1 - p(-z)$
3. Point of inflection
   $p(0) = \frac{1}{2}$
4. Inverse function
   $p^{-1} = \log(\frac{p}{1-p})$
5. 1st and 2nd Derivatives
   $p^{'}(z) = p(z)(1 - p(z))$
   $p^{''}(z) = p'(z)(1 - 2p(z))$



In machine learning, *sigmoid* function explicitly means logistic function.

Linear Regression: Formal Definition (1D)

Problem: for a given dataset $\mathbb{D} = \{(x_1, y_1), \cdots, (x_n, y_n)\}$, estimate the probability that the input x belongs to one of the binary output $y \in \{0, 1\}$, where $x_i \in \mathbb{R}^1$.

Model: we use a linear combination of the input variable x with parameter β to construct the logistic function:

$$p(x; \beta) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}} = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}}$$

Linearization: recall the inverse function of the logistic function

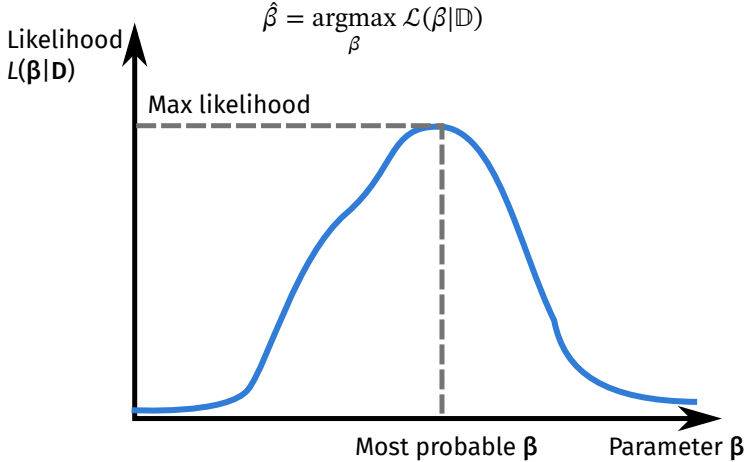
$$p^{-1}(x; \beta) = \text{Logit}(p) = \log(\frac{p}{1-p}) = \beta_0 + \beta_1 x$$

p^{-1} is called the **Logit** function, measuring the log-odds (i.e. probability ratio between a head and tail).

Question: can we directly use the Logit function form to perform a logistic regression?

Fitting by Logistic Function: the Maximum Likelihood Estimation

A common approach to solve logistic regression is to maximize the **likelihood function** $\mathcal{L}(\beta|\mathbb{D})$ of the logistic function by varying the parameter set β.
The logistic regression problem is solved by finding:



The Likelihood of Logistic Function

The likelihood $L(\beta)$ of logistic functions is derived from the assumption that the response variable follows a Bernoulli (binary) distribution. Assume for any datum (x_i, y_i), $p_i = p(x_i; \beta)$, we have:

$$\mathcal{L}(\beta|\mathbb{D}) = \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i}$$

The log-loss likelihood ℓ is used more frequently:

$$\ell(\beta|\mathbb{D}) = -\log\mathcal{L}(\beta|\mathbb{D}) = -\sum_{i=1}^{n}[y_i \log p_i + (1-y_i)\log(1-p_i)]$$

$\ell(\beta|\mathbb{D})$ is the actual **cost function** of logistic regression.

Now we just need to solve the optimal parameter set $\hat{\beta} = \underset{\beta}{\operatorname{argmin}}\ \ell(\beta|\mathbb{D})$.

Montgomery & Runger, Applied Statistics & Probability for Engineers, 6th Ed, Chapter 7

Numerical Optimization of the Cost Function: Gradient Descent

The likelihood function **does not** have a closed-form solution and the minimal value of $l(\beta)$ has to be solved numerically.

Algorithm: for each parameter β_j $(j = 0, 1)$, we pick the initial guess β_j^0, compute $l(\beta|\mathbb{D})$ and the first derivative $\partial l/\partial\beta_j$. We then iteratively update β_j by the gradient descent between steps s and $s+1$:

$$\beta_0^{s+1} = \beta_0^s - \alpha^s \frac{l(\beta^s)}{\partial l/\partial\beta_0}$$
$$\beta_1^{s+1} = \beta_1^s - \alpha^s \frac{l(\beta^s)}{\partial l/\partial\beta_1}$$

The parameter α^s is called the **learning rate** (at step s), an important hyper-parameter in machine learning.

Question: where are the data points (x_i, y_i) in the above equation?

Gradients of Logistic Regression

The logistic function is a perfect candidate for gradient-based optimization due to the neat form of gradients.

We can rewrite the cost function $\ell(\beta|\mathbb{D})$ of logistic function as:

$$\ell(\beta|\mathbb{D}) = \sum_{i=1}^{n} -\log(1-p_i) - y_i(\beta_0 + \beta_1 x_i)$$

Then the first partial derivatives are:

$$\frac{\partial \ell}{\partial \beta_0} = -\sum_{i=1}^{n}\left[-\frac{1}{1+e^{-(\beta_0+\beta_1 x)}} + y_i\right] = \sum_{i=1}^{n}(p_i - y_i)$$

$$\frac{\partial \ell}{\partial \beta_1} = -\sum_{i=1}^{n}\left[-\frac{1}{1+e^{-(\beta_0+\beta_1 x)}} x_i + y_i x_i\right] = \sum_{i=1}^{n}(p_i - y_i)x_i$$

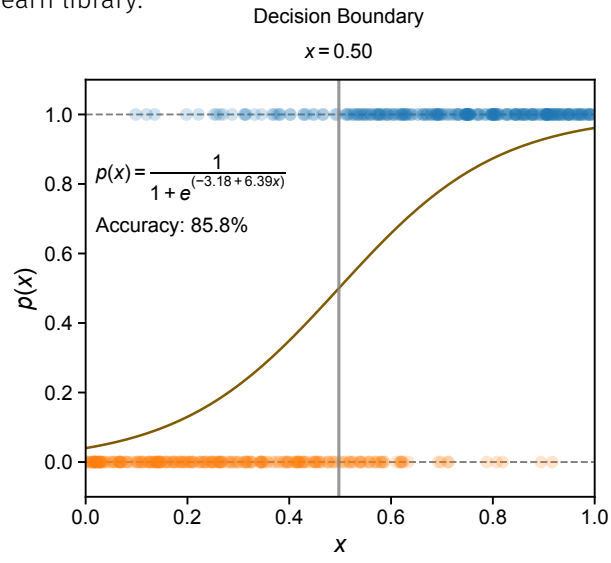
In practice, other methods like the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm are used instead of Newton's method.

Logistic Regression Demo: Cheater's Coin

Logistic regression using Python's scikit-learn library:

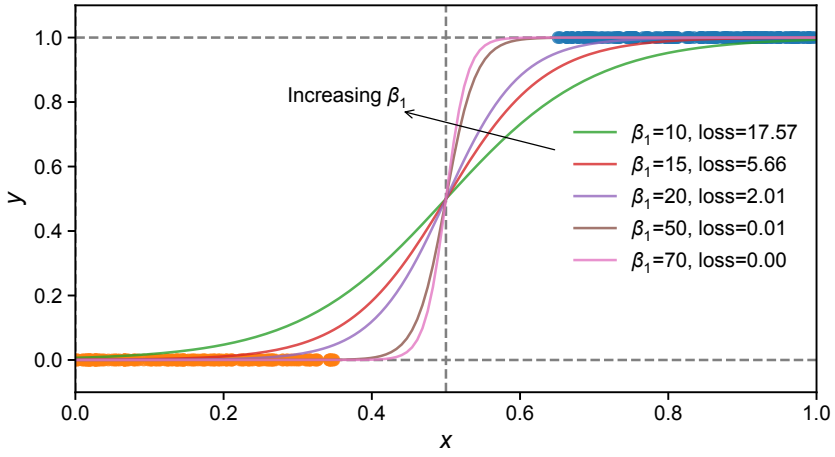
```
import numpy as np
from sklearn.linear_model \
    import LogisticRegression as LR
data = np.loadtxt("my_result.csv")
x, y = data[:, :-1], data[:, -1]
alpha = 0.1 # penalty strength
lr = LR(penalty="l2", C=1 / alpha)
# 1. create a model
model = lr.fit(x, y)

# 2. get weight and bias
weights = model.coef_[0]
bias = model.intercept_

# 3. probability
p_predict = model.predict_proba(x)

# 4. predict class
y_predict = model.predict(x)

# 5. accuracy (whole dataset)
acc = model.score(x, y)
```

Do you spot any issues?



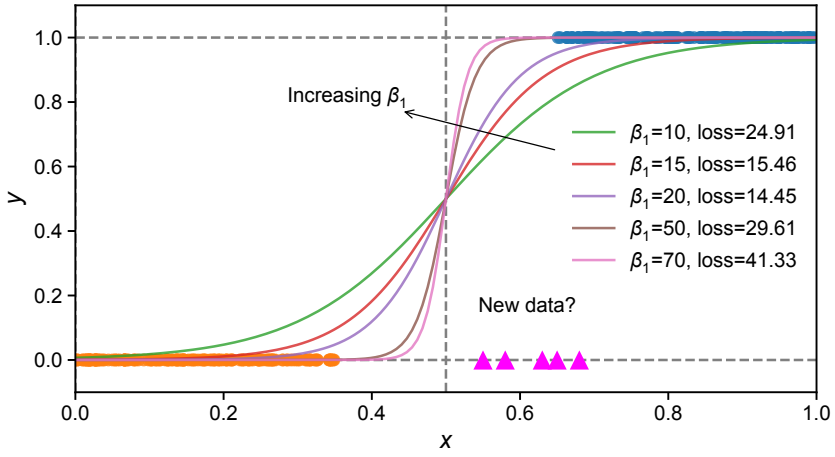
Caveat: Regularization

The naive logistic regression may work poorly on some datasets! Imagine the coin will always ends in head when $x > 0.5$, and tail if $x < 0.5$. Which model is correct?



Caveat: Regularization

The naive logistic regression may work poorly on some datasets! Imagine the coin will always ends in head when $x > 0.5$, and tail if $x < 0.5$. Which model is correct?



Regularization (1)

The naive logistic regression may lead to **over-fitting**. We can add additional penalty terms $P(\beta)$ to our cost function to avoid large coefficient β (weights). Two commonly used regularization methods in statistics and machine learning are :

- L1 (lasso) regularization

$$P(\beta) = \sum_{j=0}^{d} |\beta_j|$$

- L2 (ridge) regularization

$$P(\beta) = \sum_{j=0}^{d} \beta_j^2$$

The full cost function $C(\beta)$ for logistic regression is then:

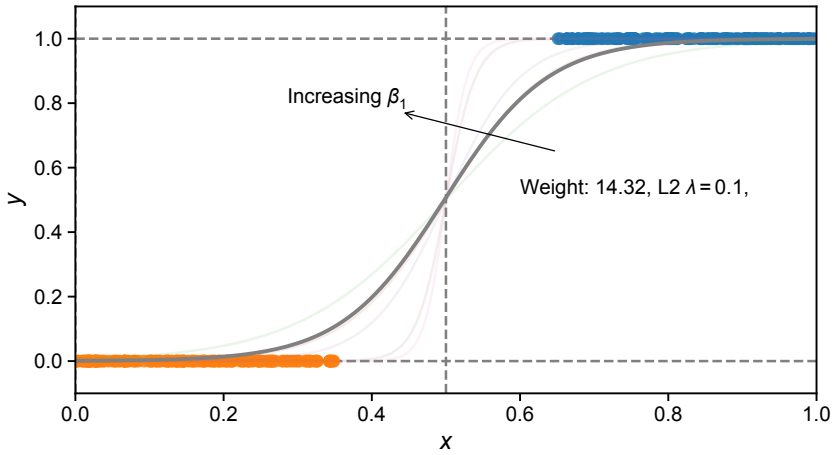
$$C(\beta) = \ell(\beta) + \lambda P(\beta)$$

where λ is the strength of regularization penalty.

The choice of L1 or L2 depends on the problem.

Regularization (2)

Let's see how regularization helps in the previous case:



How Do We Measure the Performance?

1. **Cost function**
   $\ell(\beta|\mathbb{D})$ can be a convenient measurement when rank different models.

2. **Classification accuracy**
   Accuracy = $\frac{N(\text{correctly predicted})}{n} \times 100\%$

3. **Confusion matrix**
   Similar to accuracy, but with percentage of true positive (TP), true negative (TN), false positive (FP), false negative (FN).

4. **(Pseudo)** R^2
   Unlike ordinary least square, logistic regression does not have uniquely defined goodness-of-fit metric R^2, although some pseudo-R^2 metrics have been proposed.

See Hosmer et al. *Applied Logistic Regression* 3rd Ed, Chapter 5.

Interpreting the Logistic Regression Results

Recalling the linearization of the logistic function in 1D:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x$$

There are some intuitions for the coefficients:

1. Increase x by 1 amplifies the odd ratio by e^{β_1}
2. At $x = -\frac{\beta_0}{\beta_1}$ (decision boundary), the odd ratio is 1
3. Positive β_1: the probability of predicted class $(y = 1)$ increases with larger x
4. Negative β_1: the probability of predicted class $(y = 1)$ decreases with larger x

Logistic Regression in Higher Dimensions

It is straightforward to extend the logistic regression if input variable $\mathrm{x} \in \mathbb{R}^d$.

Model:

$$p(\mathrm{x};\beta) = \frac{1}{1+e^{-(\beta_0+\sum_{j=1}^{d}\beta_d x_d)}} = \frac{1}{1+\exp(-\beta^T \mathrm{X})}$$

here $\mathrm{X} = 1, x_0, x_1, \cdots, x_d$.

Likelihood function:

$$\ell(\beta;\mathbb{D}) = -\sum_{i=1}^{n}[y_i \log p_i + (1-y_i)\log(1-p_i)]$$

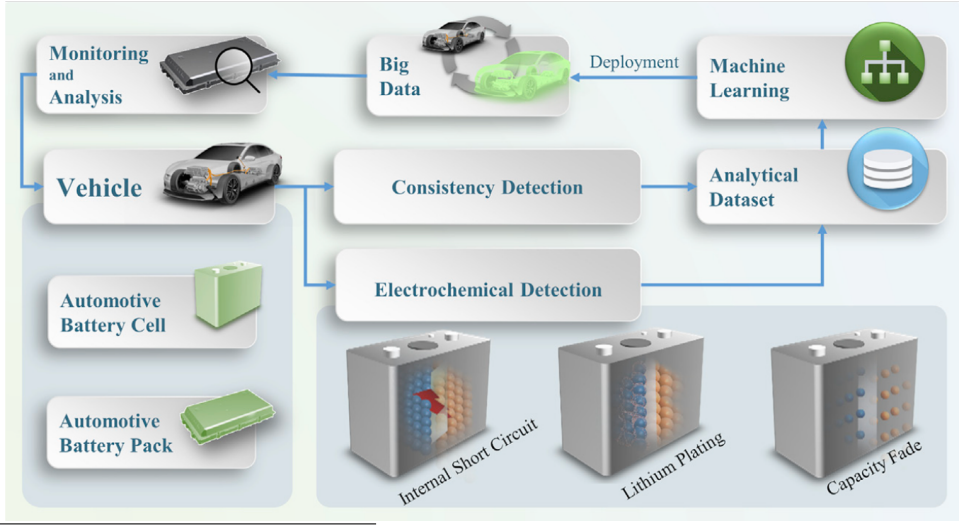
Gradient:

$$\begin{aligned}\frac{\partial \ell}{\partial \beta_j} &= -\sum_{i=1}^{n} -\frac{1}{1+\exp(-\beta^T \mathrm{X}_i)} x_{ij} + y_i x_{ij} \\ &= \sum_{i=1}^{n}(p_i - y_i)x_{ij}\end{aligned}$$

where x_{ij} is the j-th component in i-th datum.

Real-World Example: Li-ion Battery Failure (1)

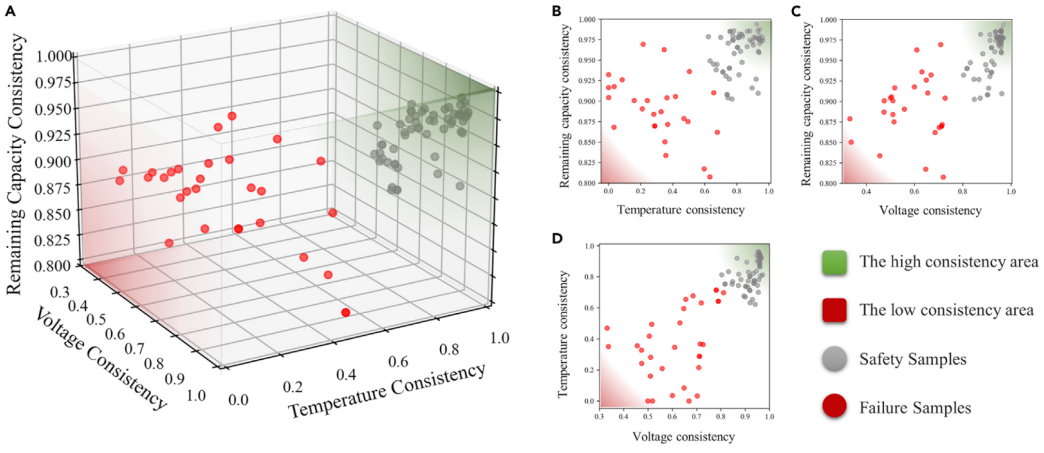
Can you predict if a lithium-ion battery in EV will fail from measurements?



Zhao et al. *iScience* **25**, 104172

Real-World Example: Li-ion Battery Failure (2)

Battery failure dataset. .



Zhao et al. *iScience* **25**, 104172

Real-World Example: Li-ion Battery Failure (3)

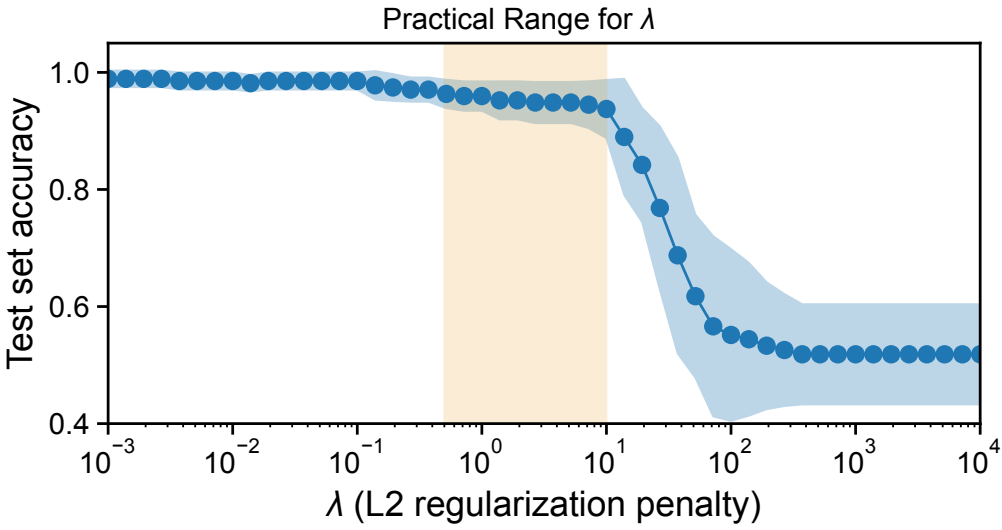
General procedures for fitting / classification using logistic regression:

- Step 1: **Dataset split**
  Rule of thumb: 80/20 or 70/30 (train/test) rule
- Step 2: **Create a logistic regression model**
  See our example in coin dataset
- Step 3: **Hyper-parameter tuning**
  Some hyper-parameters (e.g. regularization) may affect model performance beyond β
- Step 4: **Model accuracy test**
  Additional dataset or cross-validation

See course material `https://github.com/alchem0x2A/uoa_che358_mock`

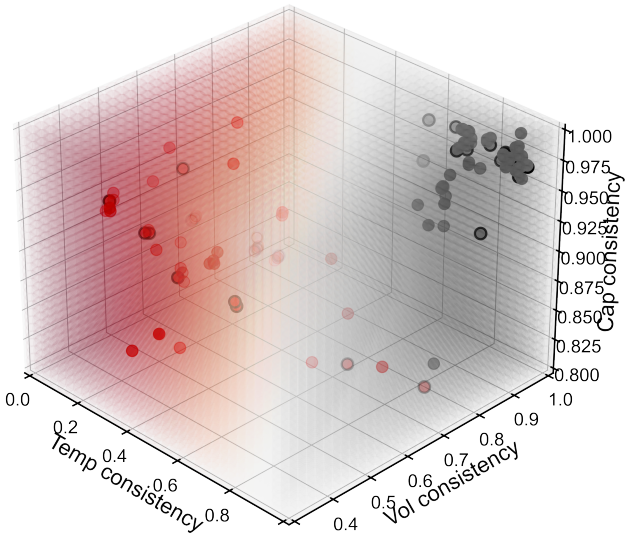
Real-World Example: Li-ion Battery Failure (4)

How does hyper-parameters affect test accuracy? (70/30 split, data tainted with 5% noise)



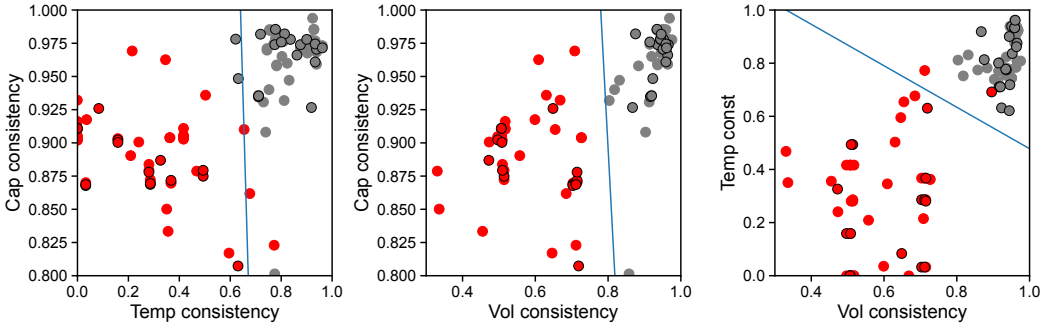
Real-World Example: Li-ion Battery Failure (5)

3D surface of the probability distribution ($\lambda = 1$). Solid circle: train data, circle with edge: test data.



Real-World Example: Li-ion Battery Failure (5)

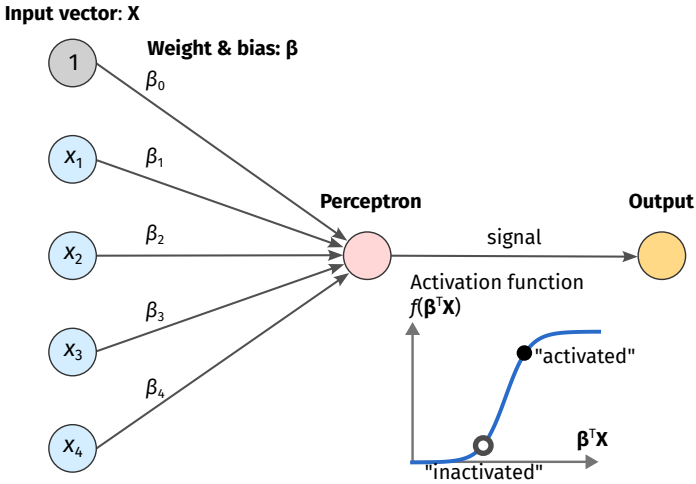
Decision boundary of the best model ($\lambda = 1$). Do you think we can further improve our model?



See course material `https://github.com/alchem0x2A/uoa_che358_mock`

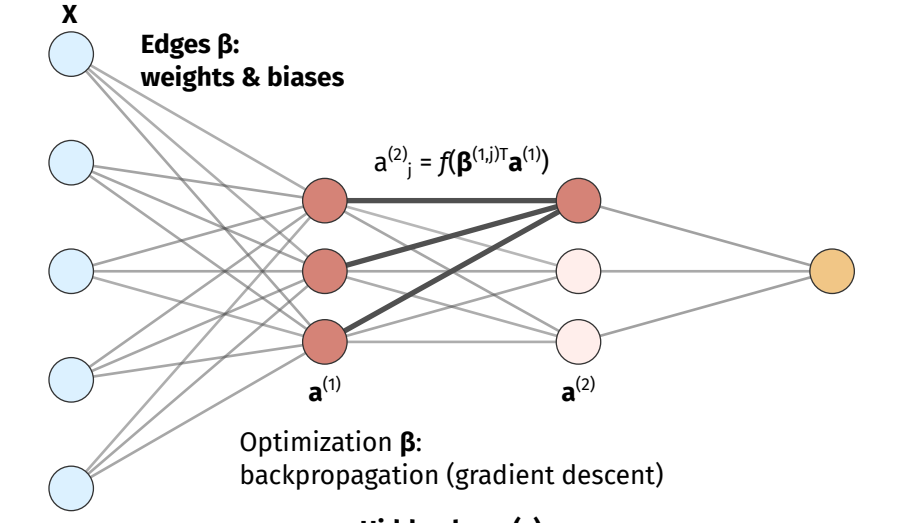
Advanced Topic: Logistic Regression to Neural Network (1)

In machine learning, the logistic regression can be viewed as a single perceptron (neuron).



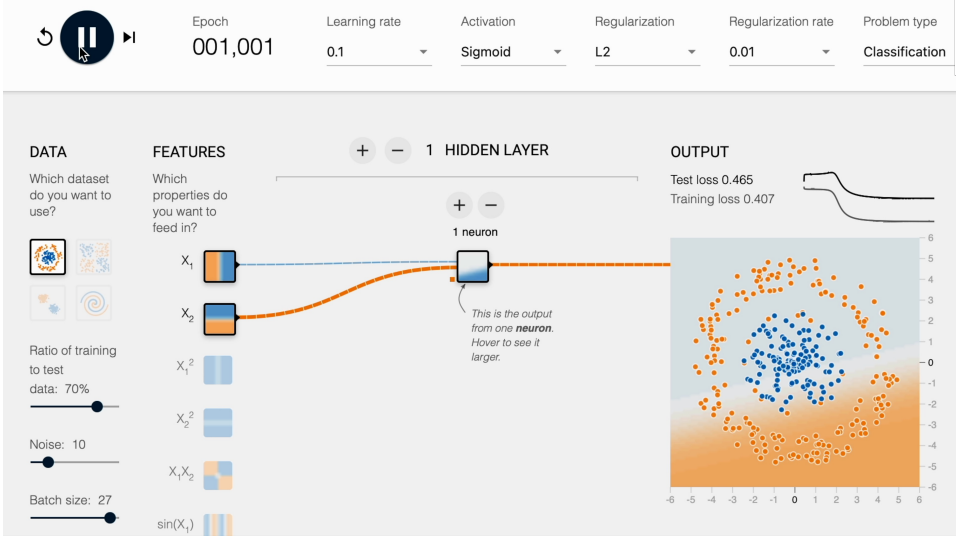
Advanced Topic: Logistic Regression to Neural Network (2)

We can extend the single “logistic perceptron” to a complex “neural network” (NN).



Logistic Regression as Single-Node NN

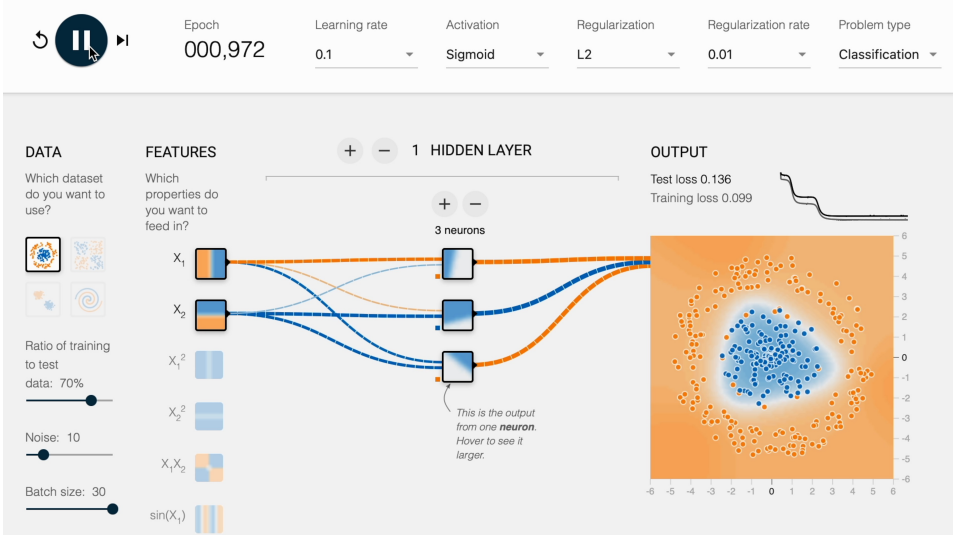
Can single logistic neuron solve non-linear classification problem?



Try it!

Performance of Deeper NNs

Multi-neuron classification



Try it!

Recap: Linear Regression vs Logistic Regression

Linear Regression

- Model: $y = \beta_0 + \beta_1 x$
- Continuous response variable
- Linear relation between x and y
- Fitting by minimizing least square error
- Model selection: maximizing R^2
- Reveal the trend in data

Logistic Regression

- Model: $p = 1/(1 + \exp[-(\beta_0 + \beta_1 x)])$
- Categorical response variable
- Linear between x and $\log[p/(1-p)]$
- Fitting by maximizing likelihood
- Model selection: multiple metrics
- Classification & probability estimation

Open Questions

1. Could you think of a way to extend binary logistic regression for multinomial classification problems?

2. Could you prove that the logistic function's log-loss is a convex function?

3. What if we use other sigmoid functions (e.g. scaled $\tanh$) for "logistic" regression? Could you compare the model robustness and computational expenses?

Questions?

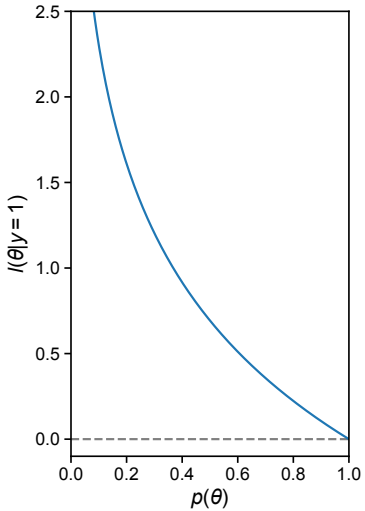
Usage of course material

Course material and codes can be found at:
`https://github.com/alchem0x2A/uoa_che358_mock`

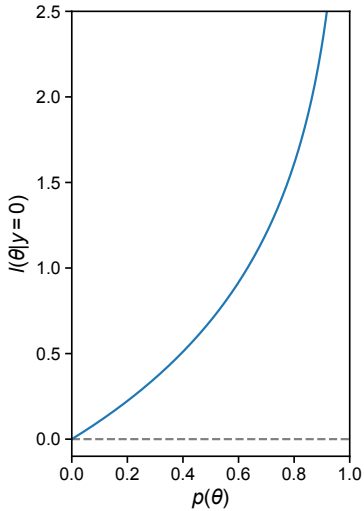
Appendix: Visualize the Cost Function

Let's take a look at the cost function if there is only 1 datum.

$y = 1$

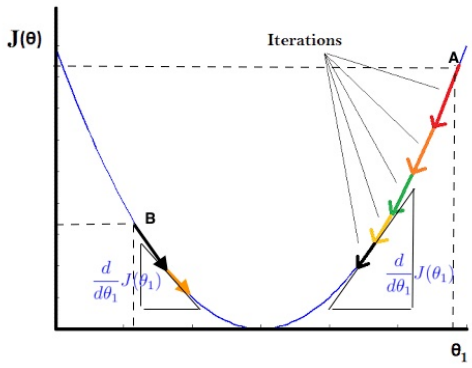


$y = 0$



Visualize the Gradient Descent

In practice, many other methods like the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm have better performance than the Newton's method.



Continuous vs Categorical Data

In contrast, response variables suitable for linear regression are:

- Continuous and quantitative
- Ordering is meaningful
- Distance is meaningful

The Classification Problem

Given:

1. Dataset: $D = \{(X_1, y_1), (X_2, y_2), \cdots, (X_n, y_n)\}$
2. Categorical response variable y: $y_i \in C$, $C = \{C_1, C_2, \cdots, C_k\}$
3. Find optimal classifier function: $p(X; \theta)$ that predicts the category using parameter set θ

What a classification model will achieve:

1. Predict the categorical label
2. Decision boundary for separation
3. Probability of outcome

Solving Classification Problem by Regression

If the classifier function $p(X;\theta)$ corresponds to the **probability** of the outcome belonging to one class, the classification problem can be solved by regression.