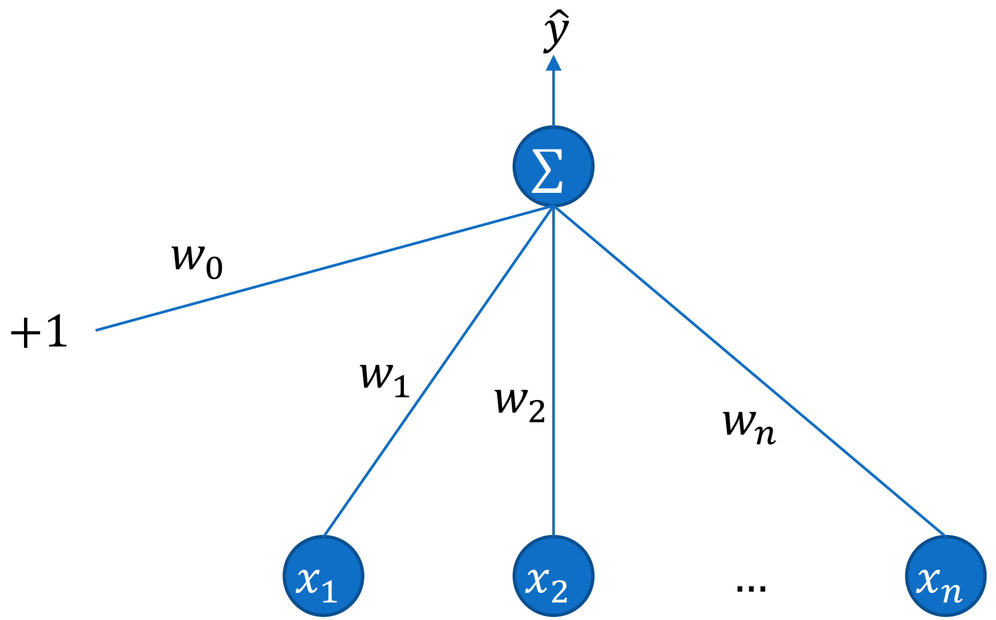
Review

LINEAR REGRESSION

Linear Regression: A Visual Perspective

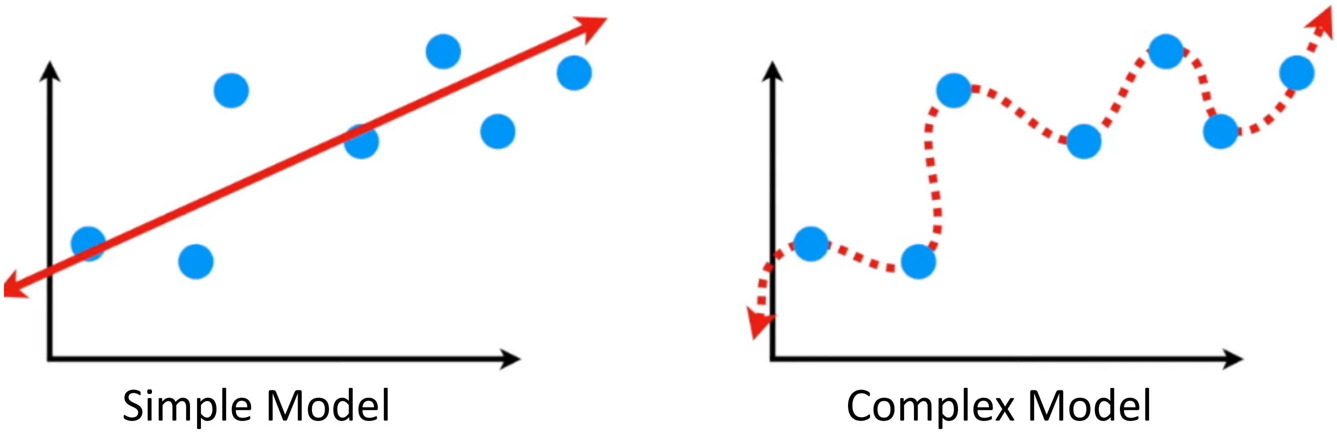
$$\boldsymbol{h(X) = W^T X = w_0 x_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n}$$

Compute Error: $y - \hat{y}$



Bias-Variance Tradeoff

- This is achieved by finding sweet spot between simple model (left) and complex model (right)



How to find the sweet spot between Bias and Variance?

Finding Sweet Spot between Simple and Complex Model

- **Bagging**
- **Feature Reduction**
  - Feature Selection (Statistical, Automated, and Manual)
  - Feature Extraction
- **Regularization**
  - Reduce magnitude/values of parameters w_j
  - Works well when we have a lot of features, each of which contributes a bit to prediction
- **Boosting**

These solutions are used to remove overfitting

Bias and Variance: Summary

- **High Bias:**
  - High Training Error
  - Validation Error or Testing Error is Close to Training Error
- **High Variance:**
  - Low Training Error
  - High Validation Error or High Testing Error
- **Fixing High Bias (possibly): It’s due to simple model.**
  - Add more input features
  - Add more complexity by introducing polynomial features
  - Decrease regularization term
- **Fixing High Variance (possibly): It’s due complex model.**
  - Getting more training data
  - Reduce input features
  - Increase regularization term

Credit: Elements of Statistical Learning by Trevor Hastie, Robert Tibshirani and Jerome Friedman
https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229

Regularization

AVOID OVERFITTING, AUTOMATIC FEATURE SELECTION

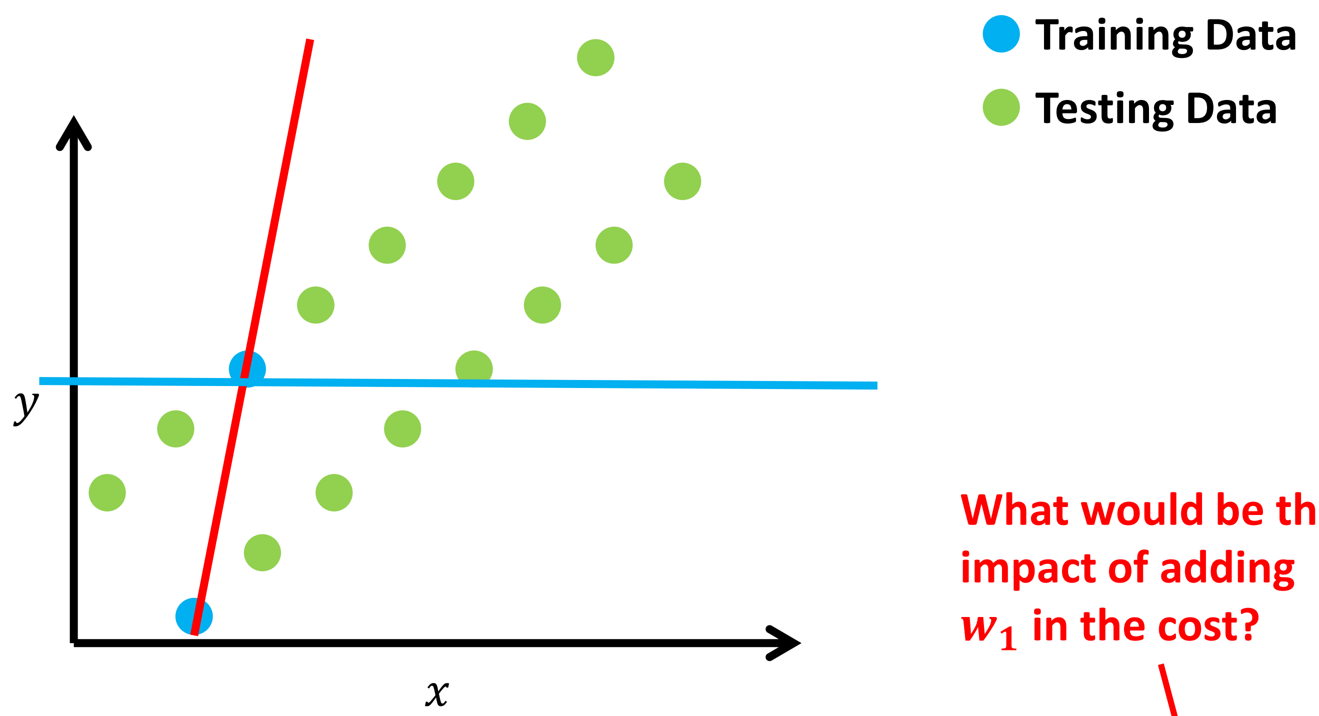
Regularization: Intuition

- Regularization works on assumption that **smaller weights generate simpler model** and thus, helps avoid overfitting.
  - Larger weights indicate overconfidence on the respective feature.
  - If we can produce a mechanism to reduce the weights, it will lead to simpler model.

$$\boldsymbol{h(x) = w_0 + w_1x_1 + w_2x_2^2 + \cancel{w_3x_3^3} + \cancel{w_4x_4^4}}$$

$$\boldsymbol{h(x) = w_0 + w_1x_1 + w_2x_2^2}$$

Regularization: Intuition



$$h(x) = w_0 + w_1 x$$

$$J(W) = \frac{1}{2m}\sum_{i=1}^{m}\left(h\left(x^{(i)}\right) - y^{(i)}\right)^2$$

What would be the impact of adding w_1 in the cost?

What to do if we want to reduce the contribution of w_1?

$$min\frac{1}{2m}\sum_{i=1}^{m}\left(w_0 + w_1 x_1^{(i)} - y^{(i)}\right)^2$$

$$min\left(\frac{1}{2m}\sum_{i=1}^{m}\left(w_0 + w_1 x_1^{(i)} - y^{(i)}\right)^2 + w_1\right)$$

Regularization: Intuition

$$min\left(\frac{1}{2m}\sum_{i=1}^{m}\left(w_0 + w_1x_1^{(i)} - y^{(i)}\right)^2 + w_1\right)$$

If w_1 is high, we are penalizing the algorithm by increasing the cost.

$$min\left(\frac{1}{2m}\sum_{i=1}^{m}\left(w_0 + w_1x_1^{(i)} - y^{(i)}\right)^2 + 10\,w_1\right)$$

What if w_1 is negative!

$$min\left(\frac{1}{2m}\sum_{i=1}^{m}\left(w_0 + w_1x_1^{(i)} - y^{(i)}\right)^2 + 10\,w_1^2\right)$$

Take absolute or squared in penalizing term.

Larger number of this term would force smaller w_1

In order to minimize this updated cost function, the value w_1 has to be small because we are penalizing high values of weights.

Regularization

$$\boldsymbol{h(X) = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n}$$

$$\min_w J(W) = \min_w \frac{1}{2m} \sum_{i=1}^{m} \left(h\left(x^{(i)}\right) - y^{(i)}\right)^2$$

$$\min_w J(W) = \min_w \frac{1}{2m} \sum_{i=1}^{m} \left(h\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} w_j^2$$

$$\min_w J(W) = \min_w \frac{1}{2m} \sum_{i=1}^{m} \left(h\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} \left|w_j\right|$$

- λ: Penalty term or regularization parameters, that determines how much to penalize the weights
- If $\lambda = 0$: Regularization is 0, we are back to original loss function.
- If λ is large: We penalize the weights and they become close to zero, resulting in a very simple model having high bias or is underfitting.
  - i.e., $h(X) = w_0$
- **Find optimal value of λ using cross-validation. (aka hyperparameter tuning)**

What is cross-validation?

Regularization

$$\boldsymbol{h(X) = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n}$$

$$\min_{w} J(W) = \min_{w} \frac{1}{2m} \sum_{i=1}^{m} \left(h\left(x^{(i)}\right) - y^{(i)}\right)^2$$

Original cost function.

Also known as Ridge Regression or L_2 regularization.

$$\min_{w} J(W) = \min_{w} \frac{1}{2m} \left[\sum_{i=1}^{m} \left(h\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} w_j^2\right]$$

Regularized cost function.
$\lambda = 0$ original cost function.
$\lambda \rightarrow \infty : w_j \approx 0$

Also known as Lasso Regression or L_1 regularization.

$$\min_{w} J(W) = \min_{w} \frac{1}{2m} \left[\sum_{i=1}^{m} \left(h\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} \left|w_j\right|\right]$$

Regularized cost function.
$\lambda = 0$ original cost function.
$\lambda \rightarrow \infty : w_j = 0$

Lasso Regression performs automatic feature selection. (The worst features get a weight of zero, making them zero).

Credit: https://medium.com/datadriveninvestor/l1-l2-regularization-7f1b4fe948f2

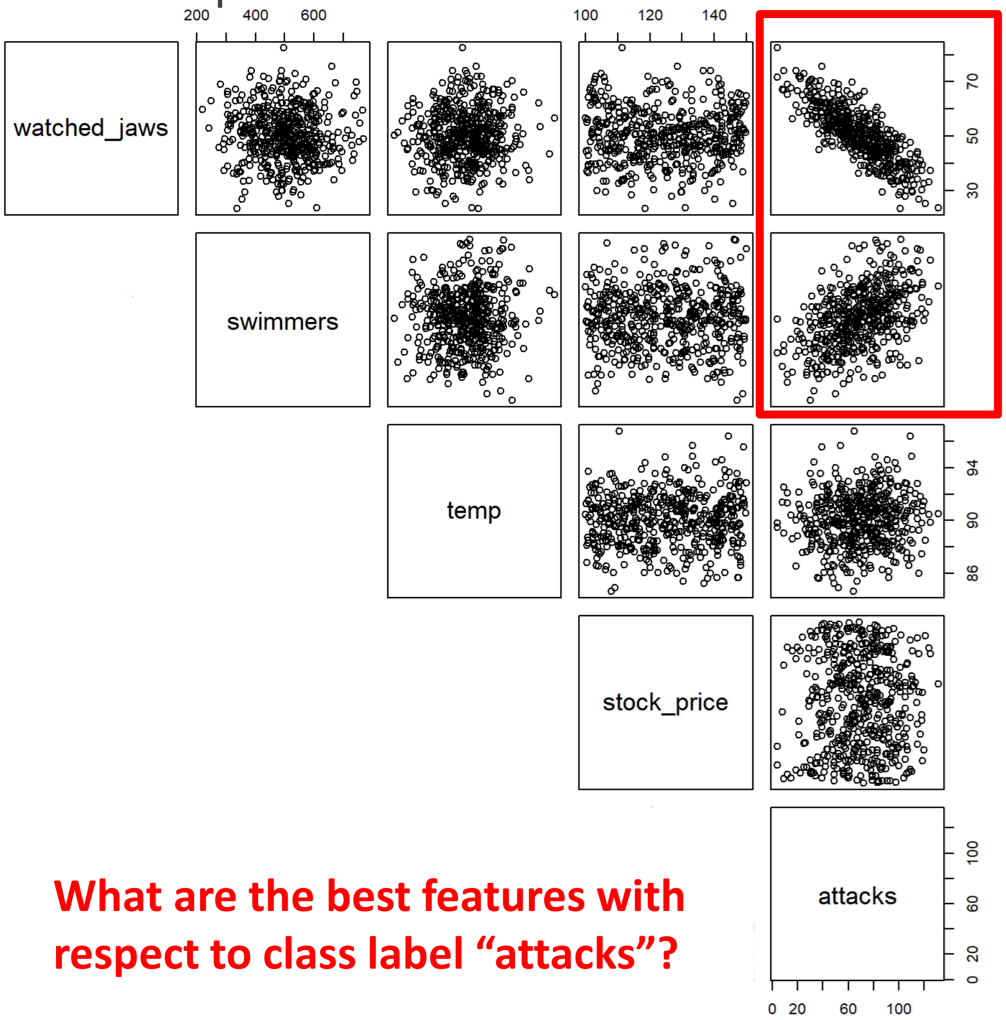
L_2 Regularization or Ridge Regression

$$\min_{w} J(W) = \min_{w} \frac{1}{2m} \left[\sum_{i=1}^{m} \left(h\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} w_j^2 \right]$$

- L_2 regularization forces the weights to be small, but does not make them zero and produces a **non-sparse solution** (no weights are equal to zero).
- L_2 is **not robust to outliers** as square terms blow up the error differences of the outliers and the regularization terms tries to fix it by penalizing the weights.
- Ridge regression performs better when all the input features influence the output and **all the wights are of roughly equal size**.

Recall : Manual Feature Selection with Correlation

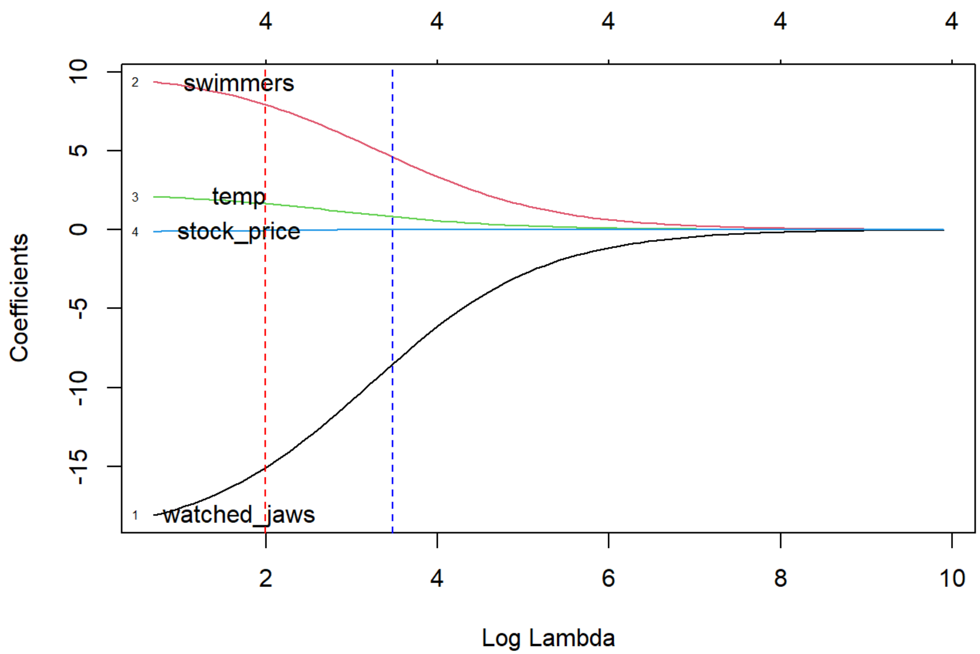
- Plot all features and label as a scatter plot



What are the best features with respect to class label "attacks"?

- **Y-axis:** Regularized coefficients (weights) for each variable (feature) after penalization
- **X-axis:** Log of the penalization parameter Lambda. Higher value of lambda indicates more regularization (i.e., reduction in coefficient magnitude, or shrinkage)
- **Curves:** Change in the predictor coefficients as the penalty term increases.
- **Numbers on top:** The number of variables in the regression model. Since Ridge regression does not do feature selection, all the predictors are retained in the final model.
- **Red dotted line:** The minimum value of lambda that results in the smallest cross-validation error.
- **Blue dotted line:** The largest value of lambda within the 1 standard error of the lambda.min. This value represents a more penalized model and can be chosen for a simpler model (less impact from features/coefficients)

- **watched_jaws:** has the strongest potential to explain the variation in the output variable, and this remains true as the model regularization increases.
- **swimmers :** has the second strongest potential to model the response, but it's importance diminishes near zero as the regularization increases.



L_1 Regularization or Lasso Regression

$$\min_{w} J(W) = \min_{w} \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n}\left|w_j\right|\right]$$

- L_2 shrinks the **parameters to zero.**
  - When input features have weights closer to zero, that leads to sparse L_1 norm. In sparse solution, majority of the input features have zero weights and very few features have non-zero weights.

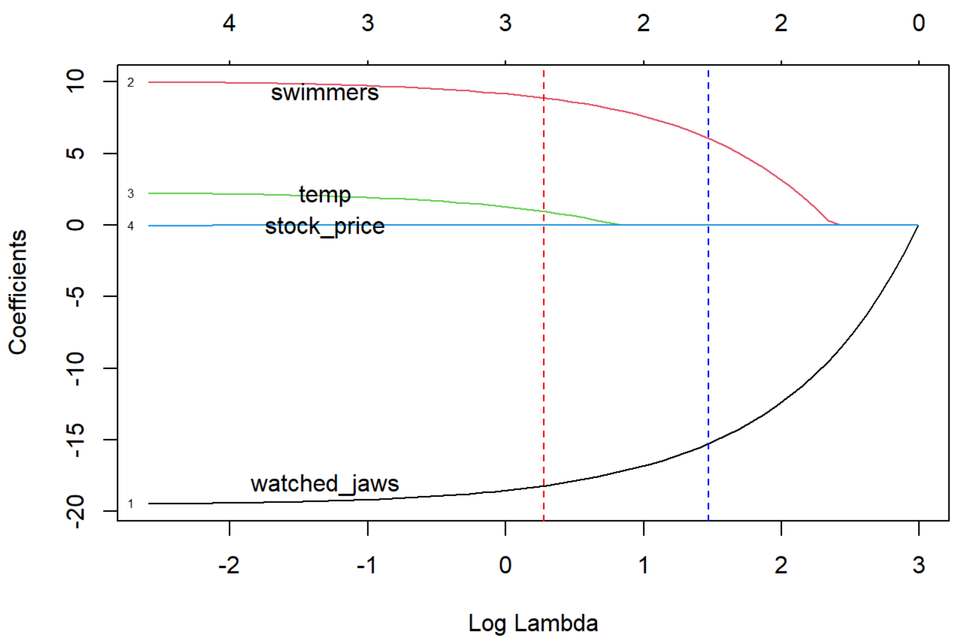
- Not all the input features have the same influence on the prediction. L_1 norm will assign **a zero weight to features with less predictive power**.

- L_1 regularization **does automatic feature selection**. It does this by assigning insignificant input features with zero wight and useful features with a non-zero weight.

Credit: https://medium.com/datadriveninvestor/l1-l2-regularization-7f1b4fe948f2

- **Y-axis:** Regularized coefficients (weights) for each variable (feature) after penalization
- **X-axis:** Log of the penalization parameter Lambda. Higher value of lambda indicates more regularization (i.e., reduction in coefficient magnitude, or shrinkage)
- **Curves:** Change in the predictor coefficients as the penalty term increases.
- **Numbers on top:** The number of variables in the regression model. Since Ridge regression does not do feature selection, all the predictors are retained in the final model.
- **Red dotted line:** The minimum value of lambda that results in the smallest cross-validation error.
- **Blue dotted line:** The largest value of lambda within the 1 standard error of the lambda.min. This value represents a more penalized model and can be chosen for a simpler model (less impact from features/coefficients)

- **temp** and **stock_price** get eliminated quickly.



Credit: https://scienceloft.com/technical/understanding-lasso-and-ridge-regression/

Differences

- **L_1 Regularization:**
  - Penalizes sum of absolute value of weights
  - Has a sparse solution
  - Has built-in feature selection
  - Is robust to outliers
  - Generates model that are simple and interpretable but cannot learn complex patterns.
- **L_2 Regularization:**
  - Penalizes sum of squared weights
  - Has a non-sparse solution
  - Has no feature selection
  - Not robust to outliers
  - Gives better prediction when output variable is a function of all input features
  - Is able to learn complex data patterns.

Credit: https://medium.com/datadriveninvestor/l1-l2-regularization-7f1b4fe948f2

Elastic Net Regularization

- **A combination of both L_1 Regularization and L_2 Regularization**

$$\min_{w} J(W) = \min_{w} \frac{1}{2m} \left[\sum_{i=1}^{m} \left(h\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda_1 \sum_{j=1}^{n} |w_j| + \lambda_2 \sum_{j=1}^{n} w_j^2 \right]$$

Excellent Explanation: https://www.youtube.com/watch?v=1dKRdX9bfIo

Implementation in sklearn for SGD Regressor

- Find optimal value for λ (or alpha in SGD regressor implementation) and penalty type (i.e., regularization)
  - Use **GridSearchCV** to find optimal hyperparamter value for alpha (λ) and penalty with respect to SGDRegressor estimator.
  - **penalty *{'l2', 'l1', 'elasticnet', None}***
  - **alpha *[0.0001, 0.001 …]***
  - **Set cv=10 (i.e., 10 fold cross validation)**
- Print optimal hyperparameters (i.e., best parameters) from **GridSearchCV**
- Use best found hyperparameters to train SGDRegressor

Implementation in sklearn for OLS Regression

- Find optimal value for λ (or alpha in sklearn implementation) for:
  - RidgeCV, ElasticNetCV, LassoCV
    - **Set cv=10 (i.e., 10 fold cross validation)**
    - **Set alphas = (0.0001, 0.001, …)**
- Print optimal hyperparameter (i.e., best alpha) from each regression
- Compute relevant metrics for all three regression models

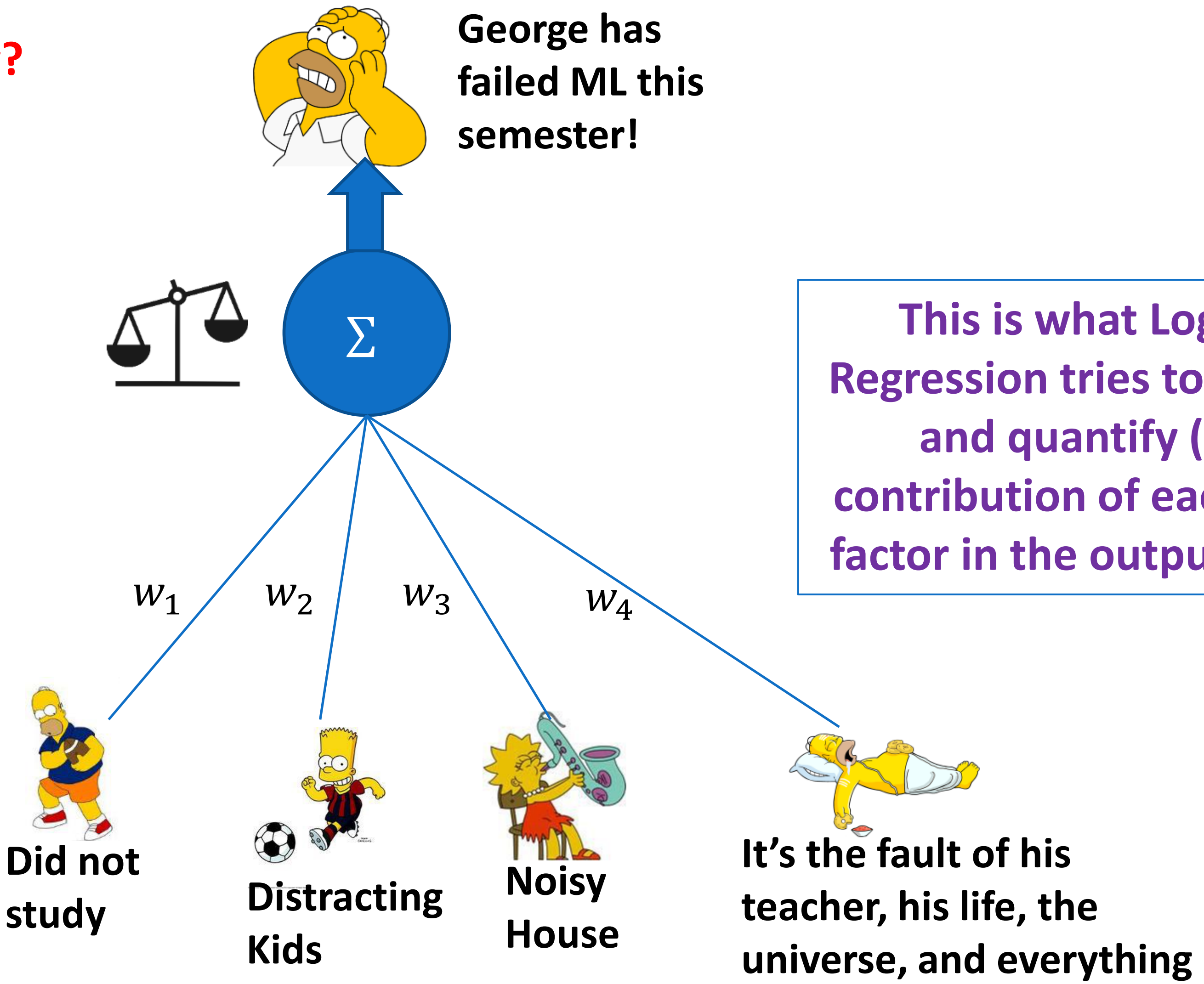
Logistic Regression

Sources

- Machine Learning for Intelligent Systems, Kilian Weinberger, Cornell, Video Lectures 35-37,
  - https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote20.html
- Deep Learning Specialization, Andrew Ng
  - https://www.coursera.org/specializations/deeplearning?utm_source=deeplearningai&utm_medium=institutions&utm_campaign=SocialYoutubeDLSC1W1L1
  - Video Lectures: C1W3L1 – C1W4L6: https://www.youtube.com/playlist?list=PLpFsSf5Dm-pd5d3rjNtIXUHT-v7bdaEIe
- Machine Learning Playground: https://ml-playground.com/#

An Event Occurs

- **But Why?**



This is what Logistic Regression tries to capture and quantify (The contribution of each input factor in the output event)

Logistic Regression

- Discover the relation between features and some outcome (categorical)
- Logistic Regression is the baseline supervised machine learning algorithm for classification
- It has a very close relationship with neural networks
  - A neural network can b viewed as a series of logistic regression classifiers stacked on top of each other.

Logistic Regression

- The outcome is viewed as a weighted sum of features
- A bias (w_0, or y-intercept) to adjust the trend of the sum
- The learned weights enhance important (discriminating) features and suppress unimportant ones.

Logistic Regression

- Classify an observation into:

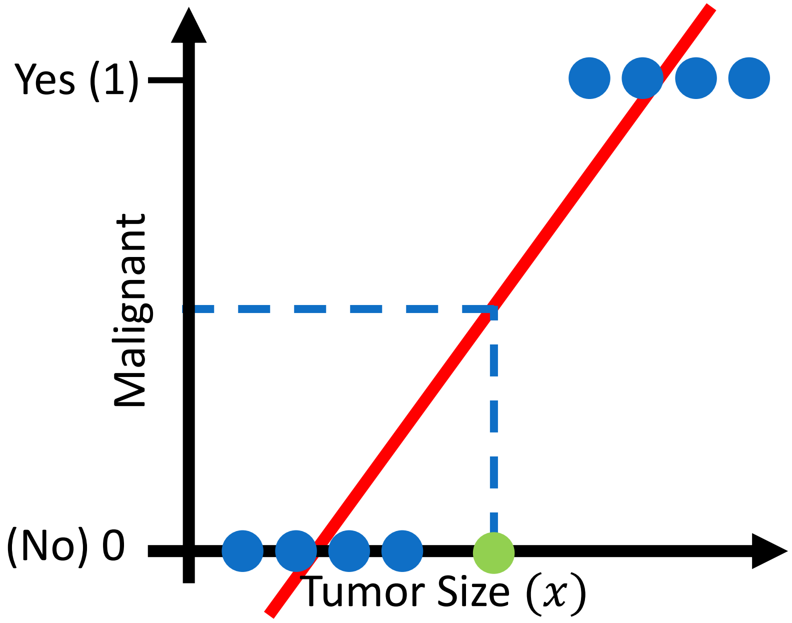
- **One of two classes (binary or binomial)**
  - Sentiment: Positive, Negative
  - Email: Spam / Not Spam
  - Online Transaction: Fraudulent (Yes / No)
  - Tumor: Malignant / Benign

$$\boldsymbol{y} \in \{\mathbf{0}, \mathbf{1}\} \quad \begin{array}{c} \text{e.g.}, 0\text{: Negative Class} \\ 1\text{: Positive Class} \end{array}$$

- **One of many classes (multinomial)**
  - Sentiment: Positive / Negative / Neutral
  - Emotion: Happy, Sad, Surprised, Angry / …
  - Part-of-Speech Tag: Noun / Verb / Adjective / Adverb / …
  - Recognize a Word: One of $|V|$ tags

$$\boldsymbol{y} \in \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \dots\} \quad \begin{array}{c} \text{e.g.}, 0\text{: } Happy \\ 1\text{: } Sad \\ 2\text{" } Angry \\ \dots \end{array}$$

Can we use Regression for Classification?



What will happen if we use Linear Regression?

$h(X) = W^T X$

What is the label for this data point?

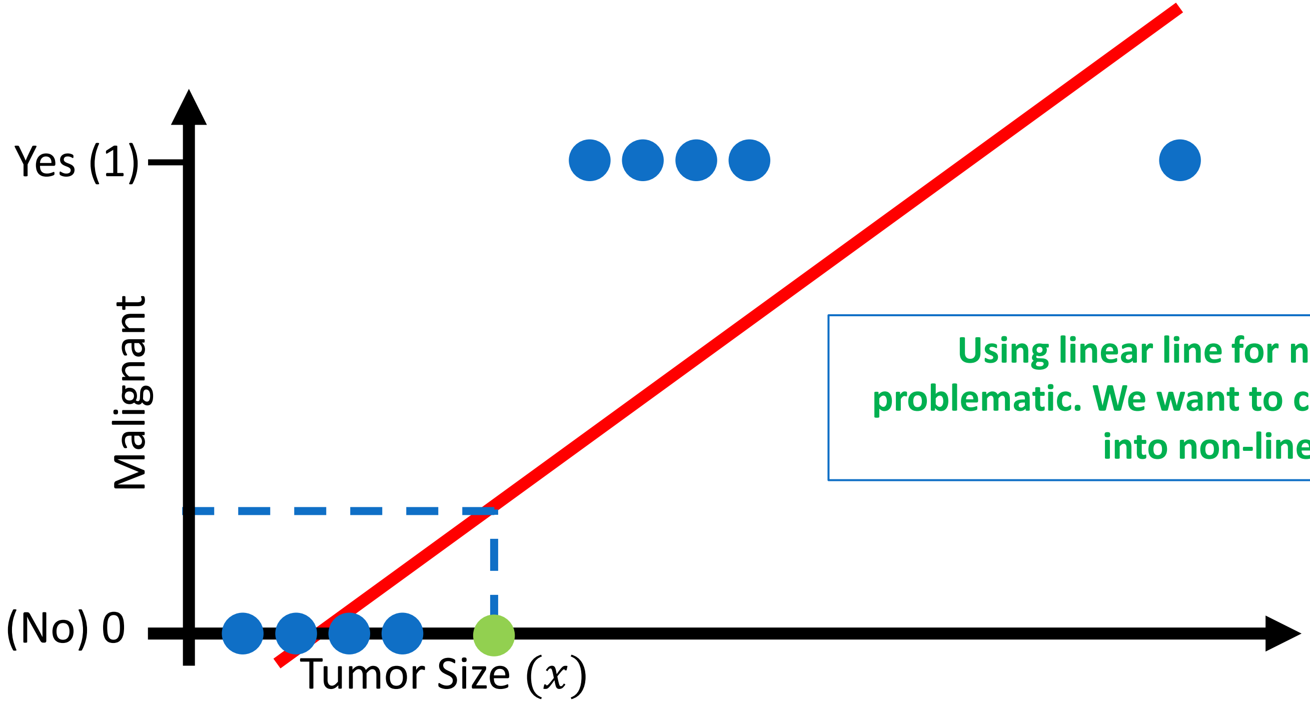
We need to define some threshold!

If $h(X) \geq 0.5$, predict $y = 1$
Else, predict $y = 0$

This also mean the output should be between 0-1 for this threshold to work!

A Threshold classifier $h(X)$ at 0.5

What about this case?



Using linear line for non-linear data is problematic. We want to convert this linear line into non-linear line!

This threshold does not work now!

A Threshold classifier $h(X)$ at 0.5

If $h(X) \geq 0.5$, predict $y = 1$
Else, predict $y = 0$

Logistic Regression

- Logistic Regression learns a vector of weights w_i from a training set
- Each weight $w_i \in \mathbb{R}$ is associated with one of the input features x_i
- The weight w_i represents how important x_i is to the classification decision
  - Can be positive (meaning the feature is associated with the class)
  - Can be negative (meaning the feature is not associated with the class)
- E.g., “positive sentiment” versus “negative sentiment”
  - The features represent counts of words in a document
  - $P(y = 1|X)$ is the probability that the document has negative sentiment
  - $P(y = 0\ |X)$ is the probability that the document has a negative sentiment
  - *“awesome”* has a high positive weight
  - *“worst”* has a high negative weight

$$\boxed{\boldsymbol{h(X) = W^T X}}$$

“Squishing” results to be between 0 and 1

- $h(X)$ is not forced to be a legal probability, that is, to lie between 0 and 1
  - In fact, since weights are real-valued, the output might even be negative
  - $h(X)$ ranges from $-\infty$ to ∞ **thus any threshold is arbitrary**
  - **We need to “squish” the outputs between 0-1**

- To create a probability, we pass $h(X)$ through the **sigmoid** function (aka logistic function) $\sigma(z)$

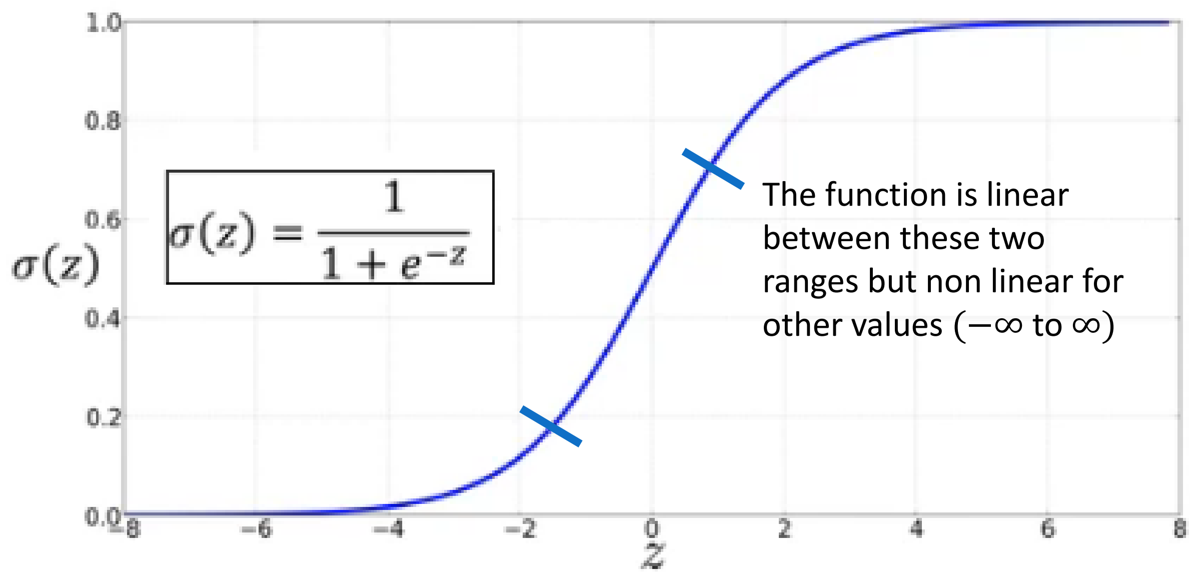
$$\boldsymbol{h(x) = \sigma(z) = \frac{1}{1+e^{-(w_0+w_1x_1)}}}$$

$$\boldsymbol{z = w_0 + w_1 x_1}$$

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

$$\boldsymbol{\sigma(h(x)) = \frac{1}{1+e^{-h(x)}}}$$

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



Online Demo

- https://www.desmos.com/calculator

$$\boldsymbol{h(x) = \sigma(z) = \frac{1}{1 + e^{-(w_0 + w_1 x_1)}}}$$

Or equally…

$$\boldsymbol{h(x) = \sigma(z) = \frac{1}{1 + e^{-(W^T X)}}}$$

Or equally…

$$\boldsymbol{z = w_0 + w_1 x_1}$$

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

Just finding the S curve is not important. Bias (w_0) is also equally important that determines the location of the threshold 0.5.

Advantages of a Sigmoid

- Maps real-valued numbers ($\mathbb{R}$) into the range $[0,1]$
- Nearly linear around 0 but has a sharp slope toward the ends
- It tends to squash outlier values toward 0 or 1
- It is differentiable, which is handy for learning
- To make it a probability:

$$\boldsymbol{P(y=1) \quad = \quad \sigma(W^T X) \quad = \frac{1}{1+e^{-W^T X}}}$$

$$\boldsymbol{P(y=1) \quad = \quad 1-\sigma(W^T X) \quad = 1-\frac{1}{1+e^{-W^T X}}}$$

- **How do we make decisions about label?**
  - For a test instance x_1, we say **yes** if the probability $P(y=1)$ is equal or greater than 0.5, and **no** otherwise.
  - We call **0.5 the decision boundary**

$$\mathbf{h(X) = \hat{y} = \begin{cases} 1 \text{ if } P(y=1|x) \geq 0.5 \\ 0 \text{ otherwise} \end{cases}}$$

Putting it all together…

- Use Sigmoid to squash the output in range 0-1
- Perform thresholding to convert the output probabilities into categorical labels
- That's how, we can use regression for classification!

Now that the output is "activated" by sigmoid function, what will happen to the cost function?

Book Reading

- Murphy – Chapter 8
- Jurafsky – Chapter 5