(Linear) Regression

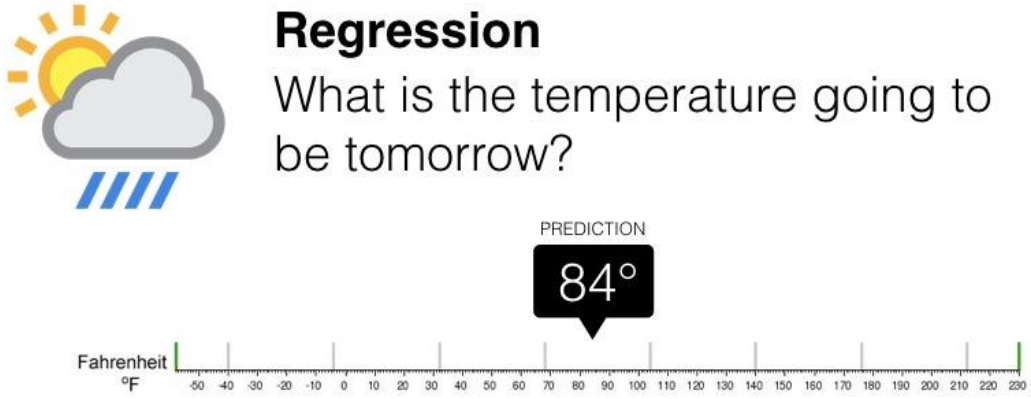
3 February 2026

Alex Lyman

Classification vs Regression

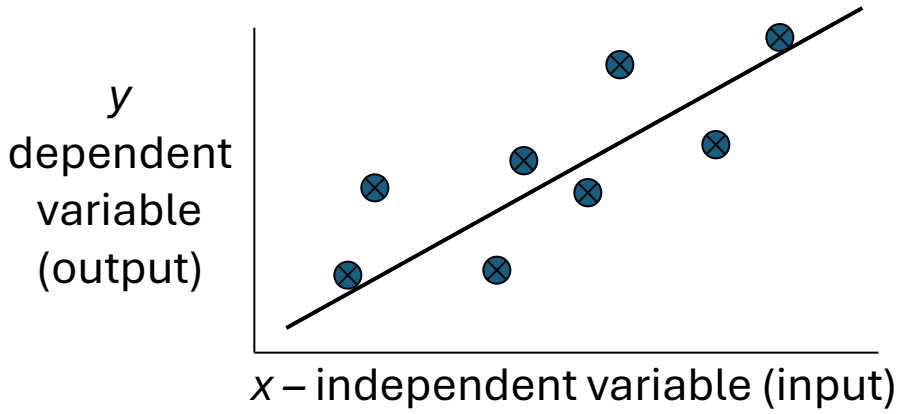
Classification & Regression

- **Classification** predicts a **discrete class** for a given input
  - And/or a probability of belonging in that class
- **Regression** predicts a **continuous value** based on the input
  - Fit the data and use the fit to predict new values



Classification vs. Regression

- For classification the output(s) is *nominal*
- In regression the output is *continuous*
  - Function Approximation
- Many models could be used – Simplest is linear regression
  - Fit data with the best hyper-plane which "goes through" the points

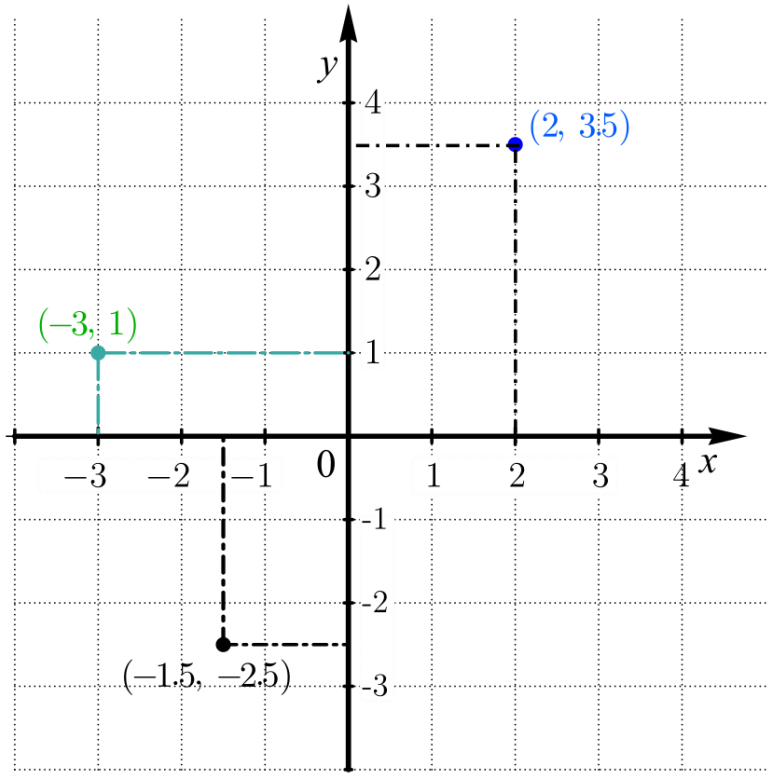


What's a Hyperplane?

High-Dimensional Space

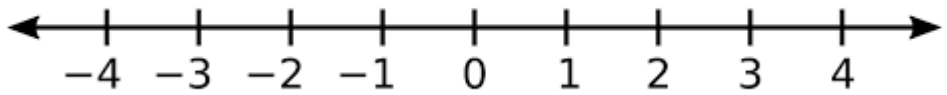
Dimensionality

- Imagine a 2-D space
- There are two dimensions, X and Y.
- Moving in one dimension doesn't have any effect on the other dimension.
- Dimensions are *orthogonal* to one another.
- That means moving in the x direction is independent of the y direction.



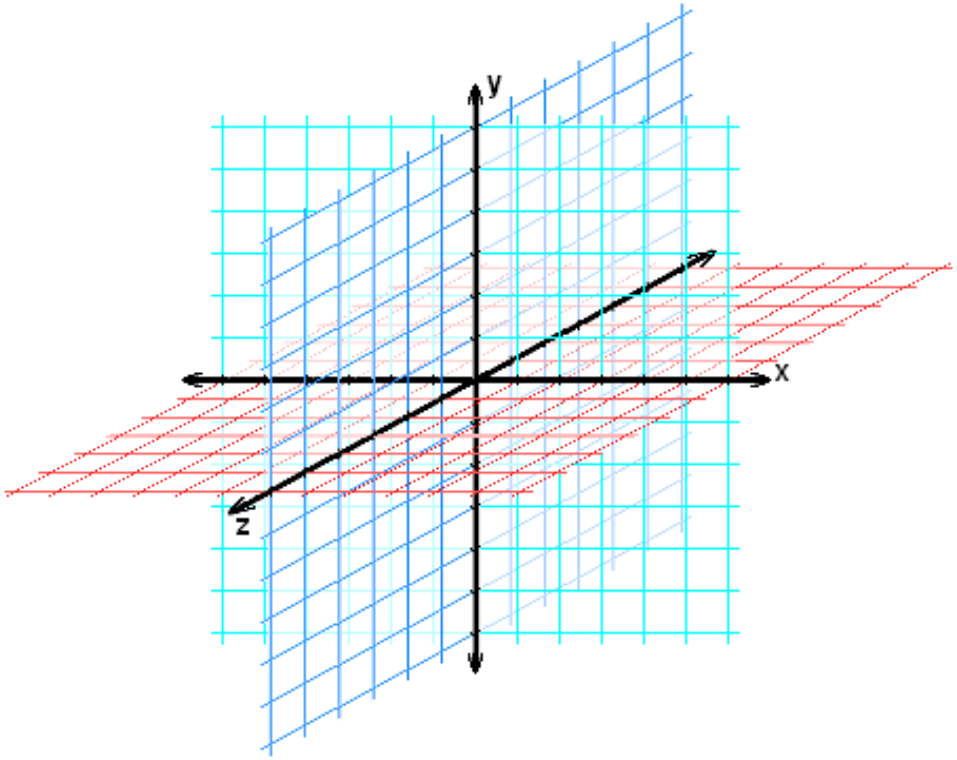
Dimensionality

- Now, imagine a 1-D space
- There is one dimension, X.



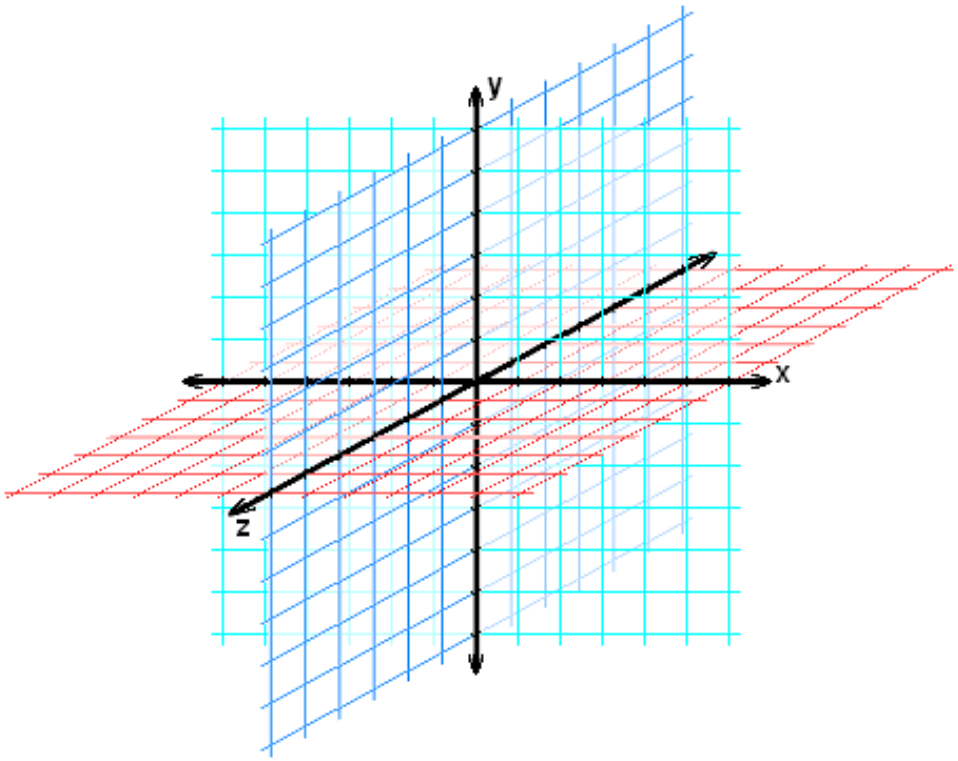
Dimensionality

- Let's go the other direction.
- Imagine a 3-D space
- There are three dimensions, X, Y, and Z.
- Moving in one dimension still doesn't have any effect on the other dimensions.
- Dimensions are still *orthogonal* to one another.
- That means moving in the x direction is independent of the y direction and the z direction.



Dimensionality

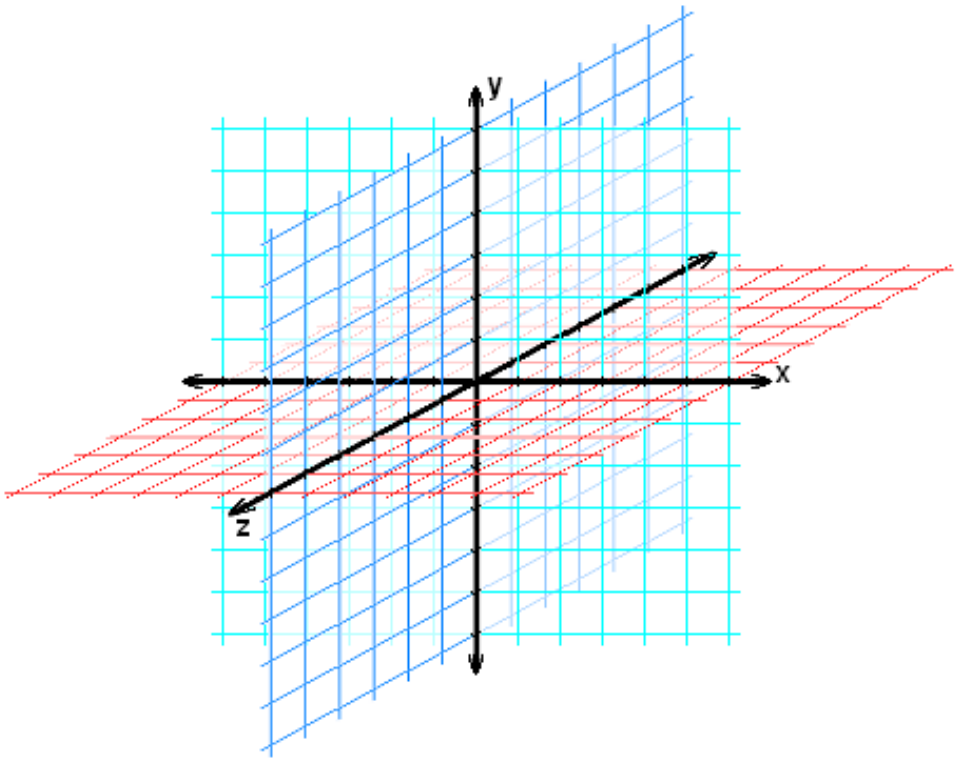
- We can generalize to more than 3 dimensions.
- This is easy to do in math, but hard to do in our minds.
- We use concepts we're familiar with from 3-D space, just generalized to more dimensions.
- For example:
  - A hyperplane is a flat, (n-1)-dimensional subspace that divides an n-dimensional ambient space into two distinct half-spaces.
  - It generalizes the concepts of a line (1D in 2D space) and a plane (2D in 3D space) to higher dimensions.



Dimensionality - Joke

A physicist asks a mathematician how he visualizes 11-dimensional space.

The mathematician replies, "It's easy. I just visualize n-dimensional space and let n equal 11. "



Dimensionality

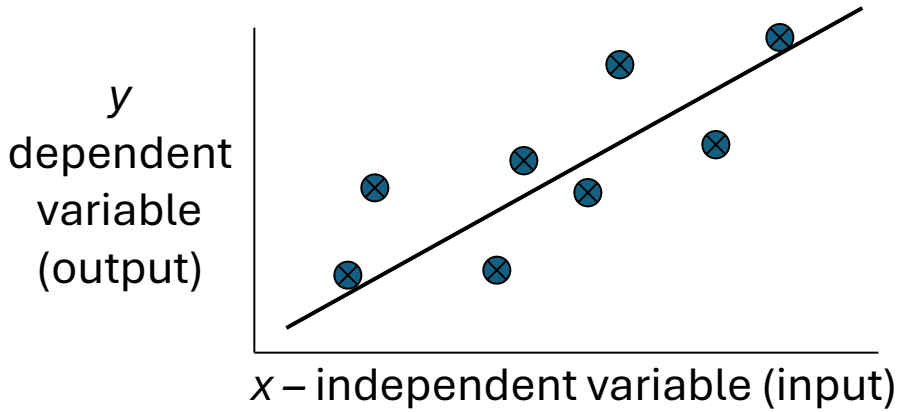
- “To deal with hyper-planes in a 14-dimensional space, visualize a 3-D space and say 'fourteen' to yourself very loudly. Everyone does it.”

  -Geoffrey Hinton

- If you can’t imagine a hyperplane or a hypersphere, that’s okay.

Back to Regression

- Linear Regression
  - Fit data with the best hyper-plane which "goes through" the points
  - In the 2-D case, a hyperplane is a line.
- How do we fit the line?
  - We *minimize error* over the training examples (just like with any ML model)



Error and Loss

Error (and Loss)

- ML models make predictions, and then we need to see how well they do.
- We do that by looking at the *Error*.
- The *Error* is a measure of the difference between a prediction and the actual value.
- Simplest possible Error is something like this:

$$Error_i = y_{true} - y_{predicted}$$

Loss

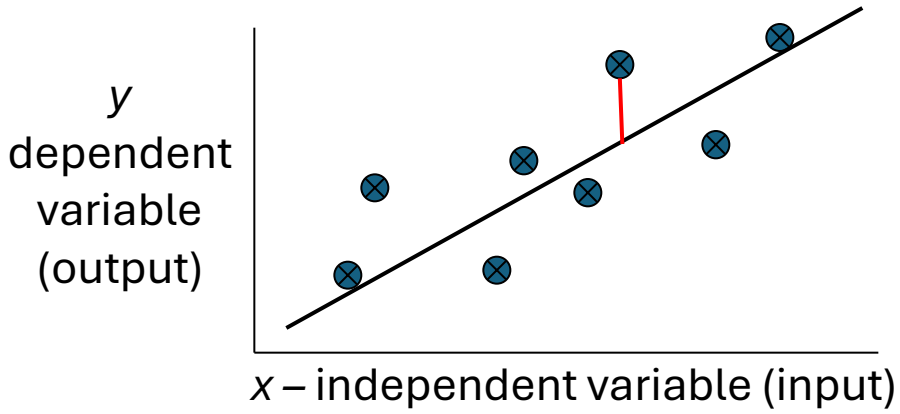
- *Loss* is the error over the dataset.
- The ML algorithm tries to minimize loss.
- For example: SSE as a *loss function*.

$$\text{Loss} = \sum(\text{Error}_i)^2$$

- Analogy: imagine you take a test:
  - Error: Did you get question 3 wrong?
  - Loss: How did you do on the test?
- You minimize total loss (on a macro level) by minimizing error.
  - Do you minimize loss by minimizing error for each example?
- Confusingly, the word *error* appears in the names of several loss functions.

Linear Regression Example

- Linear Regression Error
  - The distance between a given point and the line at that point (difference between predicted value and actual value)
- Linear Regression Loss
  - The sum of all the errors we calculated above.
  - We have lots of functions we can use to compute this!
  - Right now we'll consider L1 and L2 loss.



L1 and L2 Loss

L1 and L2 Loss

- If error = Target – output: Errors could cancel each out!
- However, two wrongs don't make a right.
- We have two very common ways of calculating loss:
  - $\Sigma|t_j - z_j|$ (L1 loss), where j indexes all outputs in the pattern

    *Add up all of the **absolute values** of the difference between each prediction and the actual value*
  - $\Sigma(t_j - z_j)^2$ (L2 loss), sum squared error (SSE)

    *Add up all of the **squares** of the difference between each prediction and the actual value*

- L2 loss is generally preferred except in cases of large outliers.
- Why outliers matter more for L2?

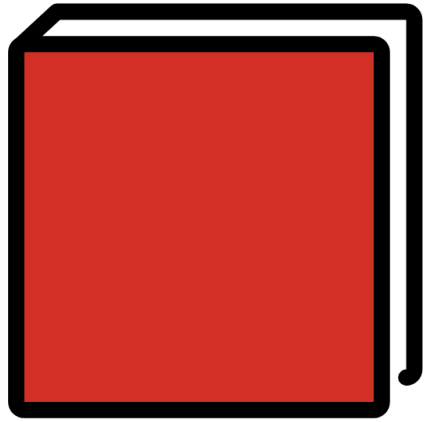
L1 and L2 Loss

Feature	L1 (Absolute)	L2 (Squared)
Outliers	Treats them like other points	**Hates them** (Sensitive)
Gradients	Constant (Hard to land exactly on 0)	Smooth (Easy to land on 0)
Solution	Median-based	Mean-based
Use Case	Noisy data with bad outliers	Clean data; Standard default

$$\Sigma|t_j - z_j| \qquad \Sigma(t_j - z_j)^2$$

L1 and L2 Loss

- Imagine a library fine policy.
- **L1 Policy:** $1 per day late.
  - 1 day late = $1.
  - 10 days late = $10.
  - 100 days late = $100.
- **L2 Policy:** The fine is **squared**.
  - 1 day late = $1.
  - 10 days late = $100.
  - 100 days late = **$10,000**.
- How do both of these deal with outliers?



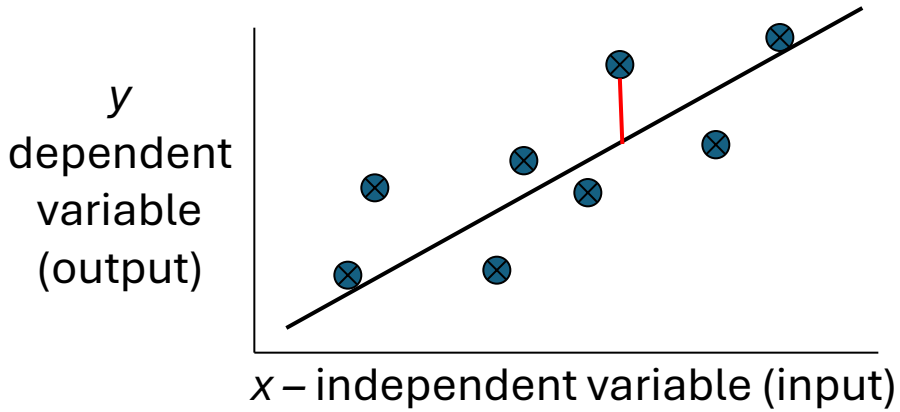
L1 and L2 Loss – Two Mnemonics to help you Remember

L1 loss – Think L1near (linear)

L2 loss – The error is **squared**. The 2 in L2 can remind you of *squared*.

Linear Regression Example

- L1 Loss
  - Add up the **absolute values** of all the distances between the points and the line.
- L2 Loss
  - Add up the **squares** of all the distances between the points and the line.



Back to Linear Regression

Simple Linear Regression

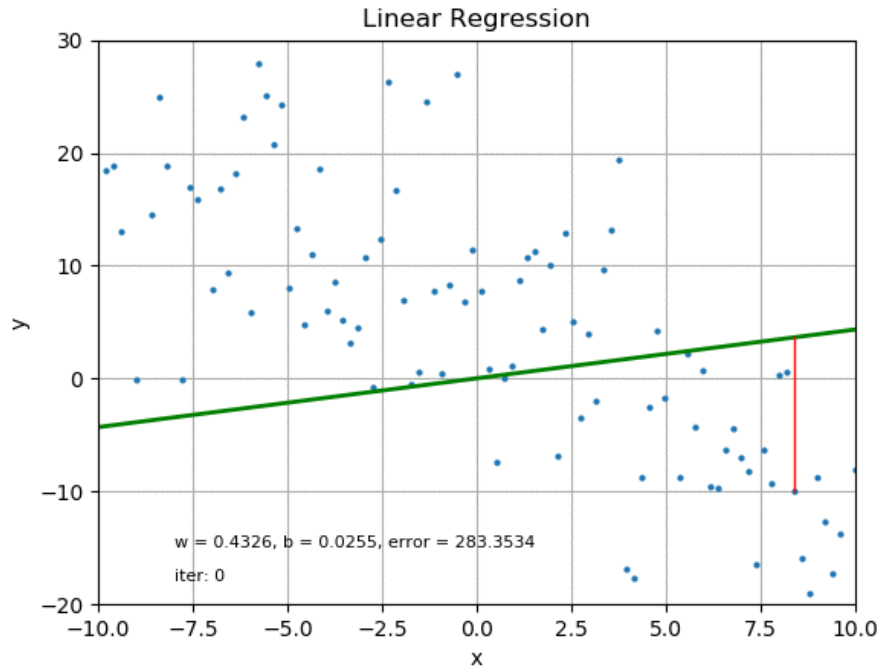
- For now, assume just one (input) independent variable *x*, and one (output) dependent variable *y*
  - *Multiple* linear regression assumes an input vector **x**
  - *Multivariate* linear regression assumes an output vector **y**
  - *Polynomial* regression allows for higher-order polynomials
- We "fit" the points with a line (i.e. hyperplane)
- Which line should we use?
  - Choose an objective function
  - For simple linear regression we usually use sum squared error (SSE)
    - $\sum (predicted_i - actual_i)^2 = \sum (residual_i)^2$
  - Thus, find the line which minimizes the sum of the squared residuals (e.g. least squares)
  - This exactly mimics the case assuming data points were sampled from an actual target hyperplane with Gaussian noise added

Linear regression

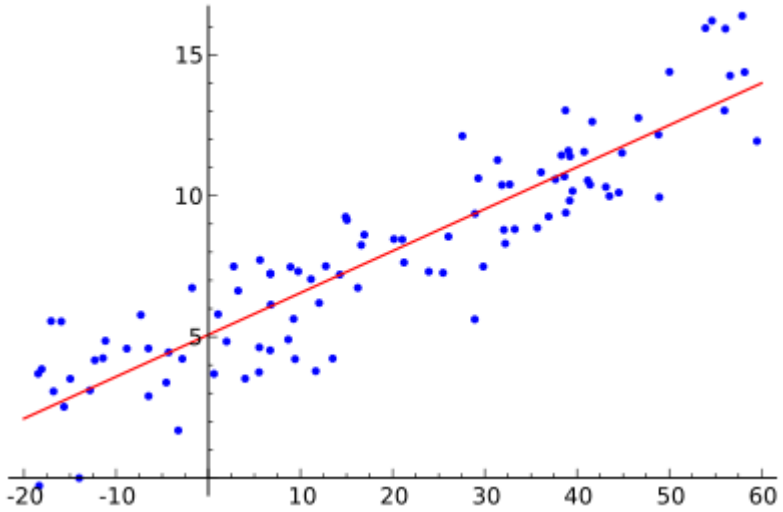
- We want to minimize the sum of square errors.

$$\sum_i (y_i' - y_i)^2$$

- Is this L1 or L2 loss?
- L2! That's Right!



Simple vs Multiple Linear Regression



$$\hat{y} = w_1 x + w_0$$

Simple Linear Regression

w_1 = weight

w_0 = bias

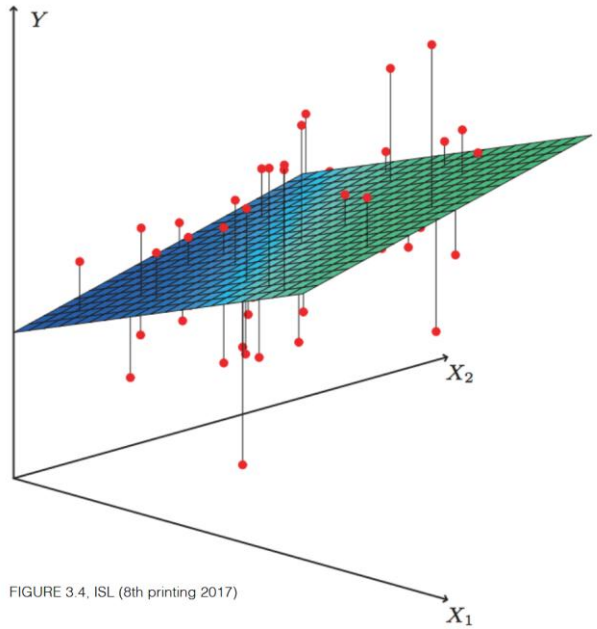


FIGURE 3.4, ISL (8th printing 2017)

$$\hat{y} = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_p x_p$$

Multiple Linear Regression (more than 2-D)

What's Going on Under the Hood?

- For the 2-*d* problem (line) there are coefficients for the bias and the independent variable (*y*-intercept and slope)

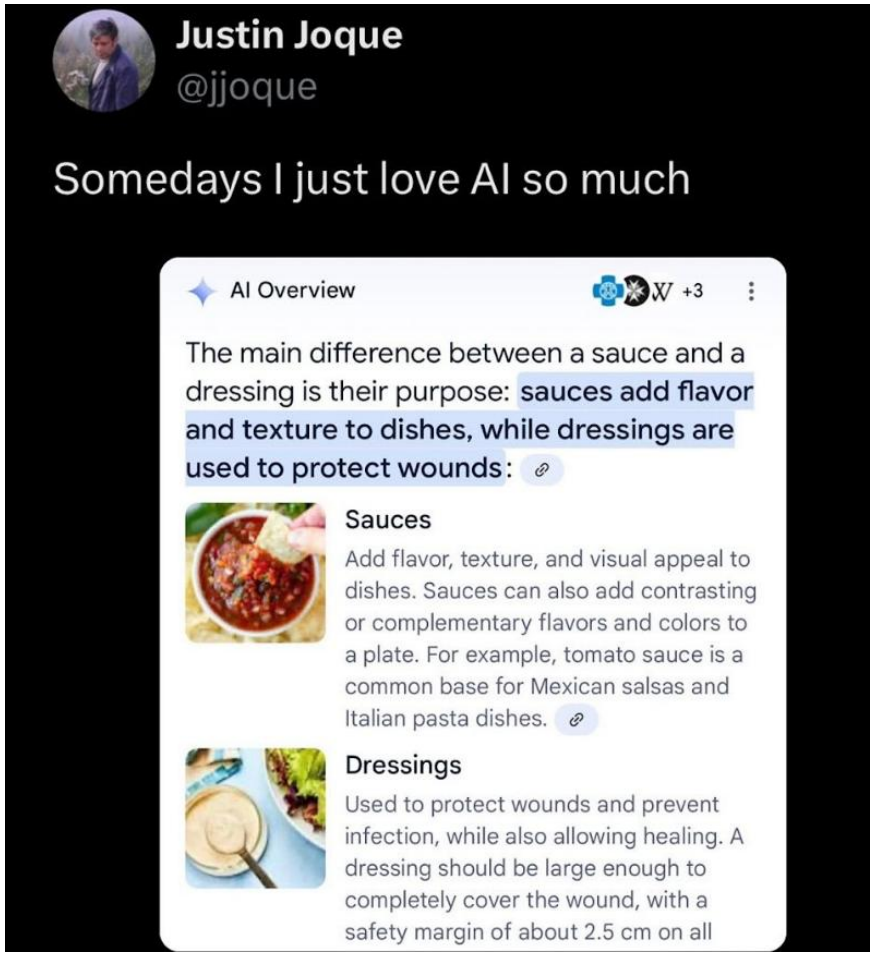
$$Y = \beta_0 + \beta_1 X$$

- To find the values for the coefficients (weights) which minimize the objective function we can take the partial derivates of the objective function (SSE) with respect to the coefficients. Set these to 0, and solve.

$$\beta_1 = \frac{n\square\, xy - \square\, x \square\, y}{n\square\, x^2 - \left(\square\, x\right)^2} \qquad \beta_0 = \frac{\square\, y - \beta_1 \square\, x}{n}$$

- You won't need to do this by hand for the test, this is mostly for your enrichment.

Break Time!


Justin Joque
@jjoque
Somedays I just love AI so much
AI Overview
+3
The main difference between a sauce and a dressing is their purpose: sauces add flavor and texture to dishes, while dressings are used to protect wounds:
Sauces
Add flavor, texture, and visual appeal to dishes. Sauces can also add contrasting or complementary flavors and colors to a plate. For example, tomato sauce is a common base for Mexican salsas and Italian pasta dishes.
Dressings
Used to protect wounds and prevent infection, while also allowing healing. A dressing should be large enough to completely cover the wound, with a safety margin of about 2.5 cm on all

Loss Functions for Regression

Linear Regression Loss

- L1 Loss
  - Add up the **absolute values** of all the distances between the points and the line.
  - Related to Mean Absolute Error

$$\Sigma|t_j - z_j|$$

- L2 Loss
  - Add up the **squares** of all the distances between the points and the line.
  - Related to Mean Squared Error

$$\Sigma(t_j - z_j)^2$$

- Zybook also talks about Huber and Quantile Loss, not going to cover in lecture today.

Quiz Time!

Evaluation Metrics for Regression

Evaluating Linear Regression

- We learned a lot of metrics for evaluating classification models.
  - Accuracy, precision, recall, etc.
- Can you calculate those for regression?
  - No!
- Good news is that there are regression evaluation metrics:
  - Mean Absolute Error(MAE)
  - Mean Squared Error (MSE)
  - Root Mean Squared Error (RMSE)

Mean Absolute Error

- Mean Absolute Error (MAE)
  - Take the mean of the absolute values of all of the residuals.
  - In practice, usually use MSE or RMSE

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Mean Squared Error

- Mean Squared Error (MSE) = SSE/*n* where *n* is the number of instances in the data set
  - This can be nice because it normalizes the error for data sets of different sizes
  - MSE is the average squared error per pattern

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- Root Mean Squared Error (RMSE) – is the square root of the MSE
  - This puts the error value back into the same units as the features and can thus be more intuitive
    - Since we squared the error on the SSE
  - RMSE is the average distance (error) of targets from the outputs in the same scale as the features
  - **Note RMSE is the root of the total data set MSE, and NOT the sum of the root of each individual pattern MSE**

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Linear Regression Interpretability

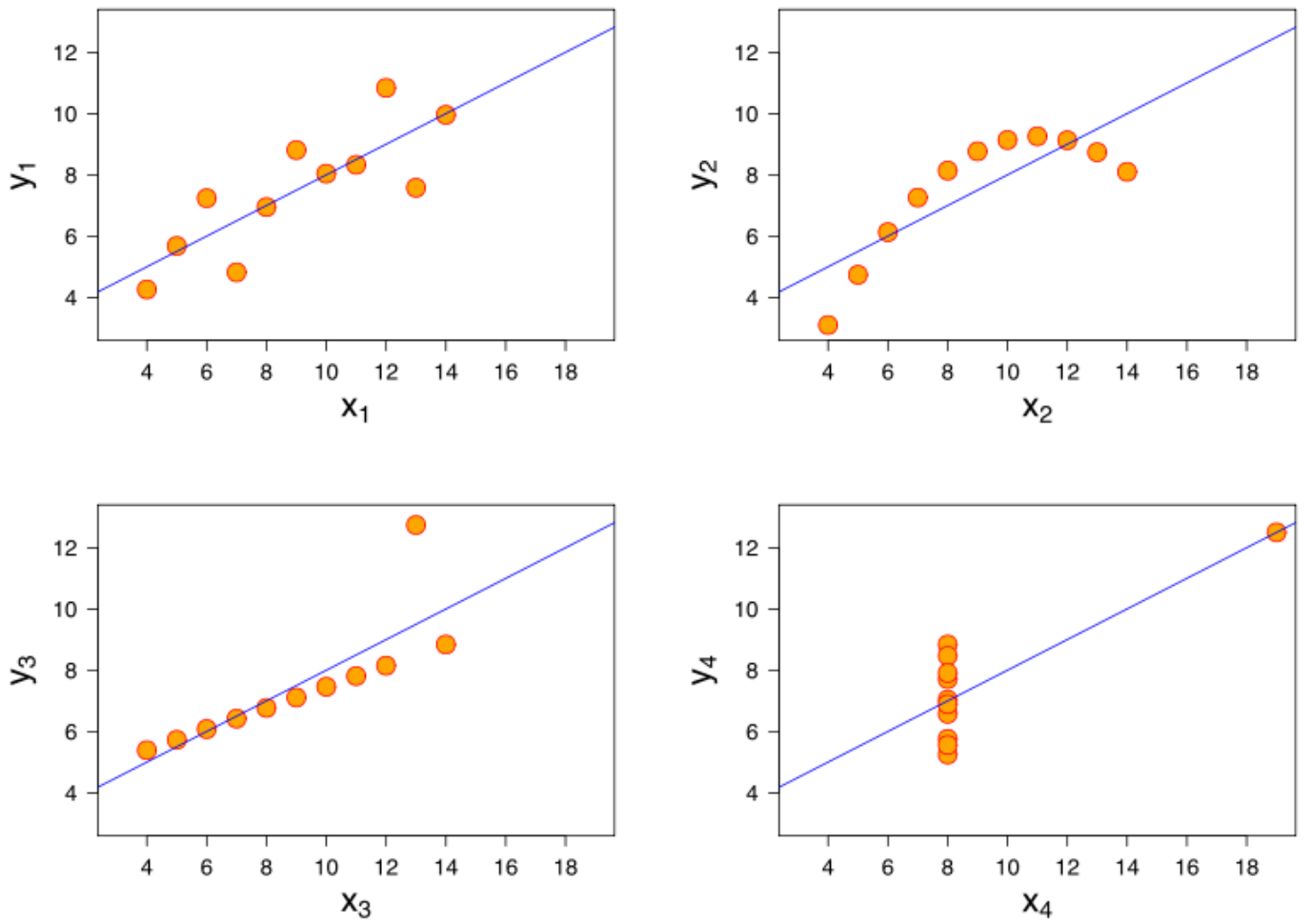
Interpretability

- One advantage of linear regression models (and linear classification) is the potential to look at the weights for insight into which input variables are most important in predicting the output
- The variables with the largest weight magnitudes have the highest correlation with the output
  - A large positive weight implies that the output will increase when this input is increased (positively correlated)
  - A large negative weight implies that the output will decrease when this input is increased (negatively correlated)
  - A small or 0 weight suggests that the input is uncorrelated with the output (at least at the 1st order)
- Linear regression/classification can be used to find best "indicators"
  - Be careful not to confuse correlation with causality
  - Linear models cannot detect higher order correlations!
    - Need the power of more complex machine learning models!!

Linear Regression Assumptions

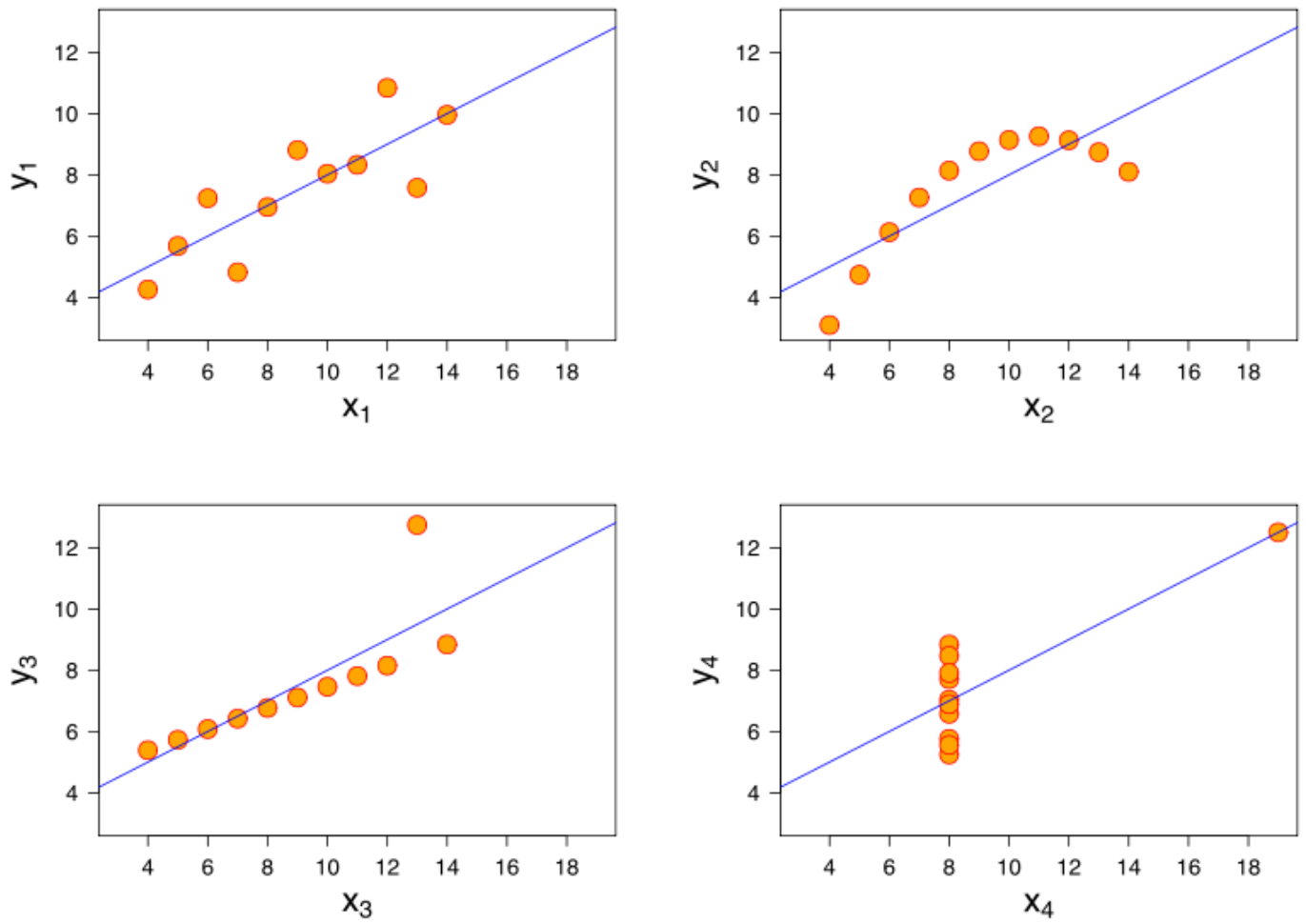
- Linear Regression makes a few assumptions:
  - The relationship between output feature and the input features is **linear** and **additive**.
  - The residuals are independent.
  - The residuals are normally distributed.
  - The residuals have constant variance across the range of values.
- What happens if these are not met?

Anscombe's Quartet



Same line for all of these datasets using standard Linear Regression

Anscombe's Quartet



Linear Regression Assumptions

- The relationship between output feature and the input features is **linear** and **additive**.
- The residuals are independent.
- The residuals are normally distributed.
- The residuals have constant variance across the range of values.

Which assumptions of Linear Regression appear to be violated?

Regularization for Regression

Reminder – What is Regularization?

- Regularization is *anything we do* to improve our model's *generalization accuracy* without improving our training performance.
- Imagine a polynomial regression problem:

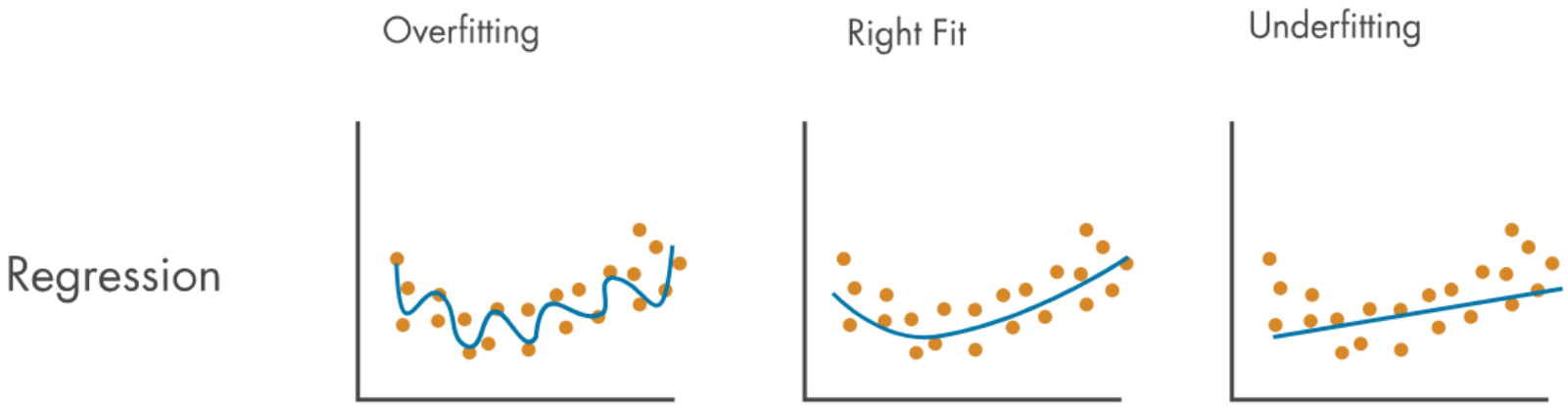
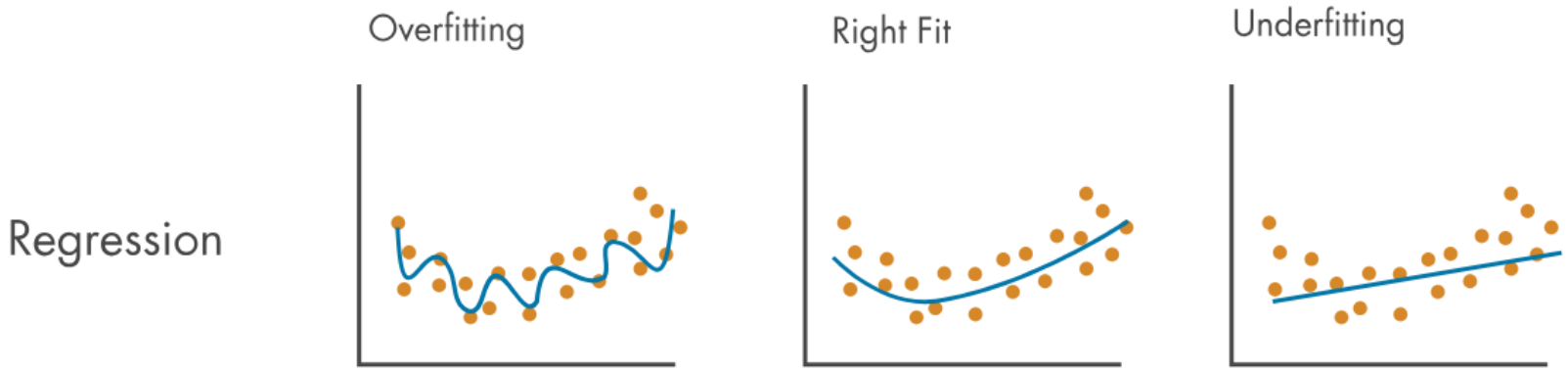


Image Credit: Mathworks

Reminder – What is Regularization?

- Imagine a polynomial regression problem:



- In the first graph, the model tried to minimize SSE, which led to it learning some crazy weights.
- This is bad.
- What if we put a 'tax' on complexity?

Regularization – The Idea

- When our model learns, it's trying to minimize a loss function.
- The loss functions we've looked at so far are just trying to minimize error.
- However, we can add additional terms to the loss function if there are other things we're trying to minimize.
- For example, if we can put a complexity term in our loss function, we can minimize **error and complexity** at the same time!

Regularization – The Idea

- Add a penalty to our equation to discourage weights from going crazy.

$$\text{Total Loss} = \underbrace{\text{Error (SSE)}}_{\text{Fit the Data}} + \underbrace{\lambda \cdot \text{Penalty}}_{\text{Keep it Simple}}$$

- Old Goal: Minimize Error
- New Goal: Minimize Error + Minimize Weights
- The Hyperparameter λ (Lambda):
  - **λ=0:** No penalty, overfitting might occur.
  - **λ=Big:** Penalty too high, model sets all weights to near 0, underfitting
  - We want to find the right λ to allow *some* complexity but prevent anything too crazy from happening.

Ridge (L2) Regularization

SSE $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \alpha \sum_{j=1}^{p} w_j^2$ Regularization Term

- Penalty = ∑(weight)^2
- Because it squares the weights, it hates huge numbers (just like L2 Loss).
- It shrinks all coefficients evenly toward zero.
- Alpha is a tunable hyperparameter.
- The Result:
  - It keeps all your features, but reduces their impact.
  - Great for handling *Collinearity* (when features are duplicates). It spreads the credit between them.

Lasso (L1) Regularization

SSE $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \alpha \sum_{j=1}^{p}|w_j|$ Regularization Term

- Penalty = ∑|weight|
- Because it uses Absolute Value, the geometry forces weights to hit exactly Zero.
- If two features are correlated will just delete one (maybe bad)
- The Result:
  - Lasso effectively **deletes** useless features.
  - If you give it 1,000 features but only 5 matter, Lasso will set 995 weights to 0.00.
  - It performs **Automatic Feature Selection**.

Elastic Net Regularization

SSE $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \alpha \sum_{j=1}^{p}|w_j| + \alpha_2 \sum_{j=1}^{p} w_j^2$ Regularization Terms

- Penalty $=\lambda_1 \sum|w| + \lambda_2 \sum w^2$
- Combines both Ridge (L2) and Lasso (L1) regularization.
- Can still zero out useless features (like Lasso).
- But for correlated features, it groups them together (like Ridge) rather than picking one at random.
- Sort of best-of-both-worlds scenario.

Warning

- L1 and L2 loss are **not the same** as L1 and L2 regularization.
- Do we understand why that is?