Regression Pt. 2

5 February 2026

Alex Lyman

Spacing effect

7 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

This article has multiple issues. Please help **improve it** or discuss these issues [hide] on the **talk page**. *(Learn how and when to remove these messages)*

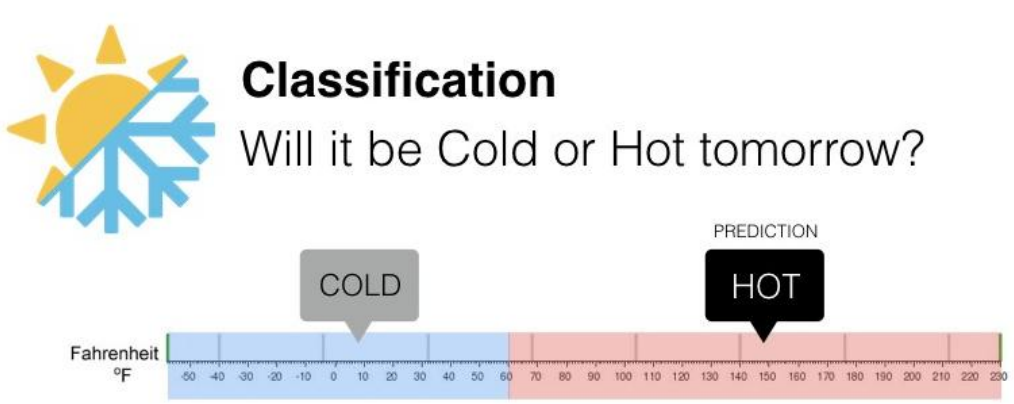
- This article **may be confusing or unclear to readers**. *(November 2014)*
- This article **contains an excessive amount of intricate detail**. *(November 2014)*

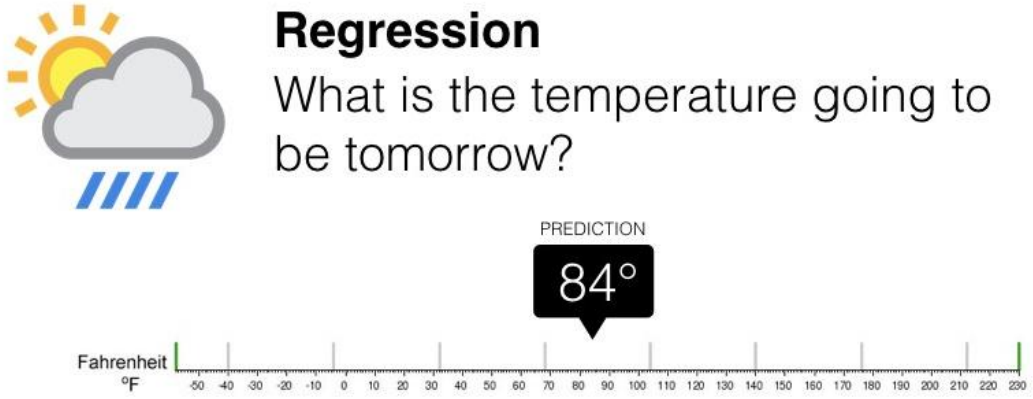
The **spacing effect** demonstrates that learning is more effective when study sessions are spaced out. This effect shows that more information is encoded into long-term memory by spaced study sessions, also known as *spaced repetition* or *spaced presentation*, than by massed presentation ("cramming").

Classification vs Regression Review

Classification & Regression

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





Model Review

Models

- A **model** is a mathematical function for describing the relationship between inputs and outputs and making predictions.
- Models learn by tweaking **parameters/weights** inside the model
- How does the model know which direction to tweak the parameters? The model tries to minimize per-example **error**.
- The model's error over the whole dataset is called the **loss**.

Loss vs Error Review

Loss vs Error

- **Error:** The difference between a model's predicted value and the actual value.
- The **Model** cares about error as it trains.
- Models learn by minimizing error (on aggregate).
- You can measure error different ways.

- **Loss:** The total error over the dataset.
- The **Model** cares about loss as it trains.
- The model learns by minimizing loss.
- You can use different functions to calculate loss.

- Error and Loss are calculated on the **Training Set**.

Loss vs Evaluation Review

Loss vs Evaluation

- **Loss:** The total error over the dataset.
- The **Model** cares about loss as it trains.
- The model learns by minimizing loss.

After the model trains:

- **You** want to see how well the model might generalize to new data.
- **You (ML engineer)** do this by evaluating your model on the validation set and/or test set.
- You can use evaluation metrics like: [*precision, recall, F1 or MSE, RMSE, MAE*] to see how well the model generalizes.

Loss vs Evaluation

- The math can be the **same** between loss functions and evaluation metrics.
- For example, you can use L2 loss (square all the errors and add them up) to train your model.
- Then, you can use Sum Squared Error to test your model on a test set.
- The math for both of those is the same.
- If the math is the same, then what's the difference?

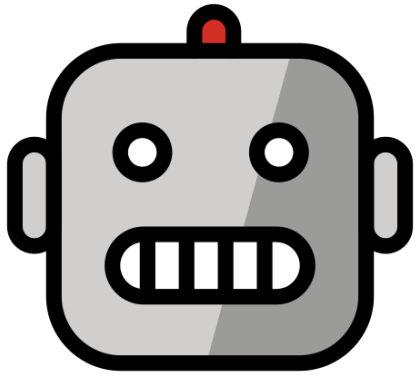
Example

Worked Example

- I’ve put together a high-level example that incorporates many aspects we’ve talked about this semester.

Error, Loss, Evaluation Example

Imagine we have a model (architecture doesn't matter) and a dataset.

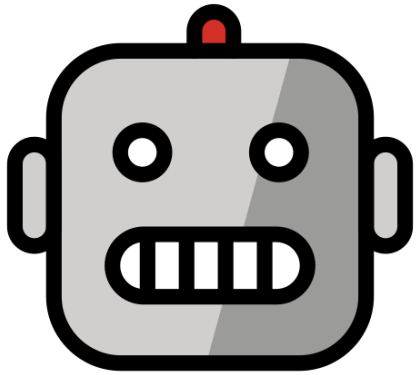


Model (dumb, doesn't know anything yet)

Dataset

Error, Loss, Evaluation Example

We split our data into a train, validation, and test set.

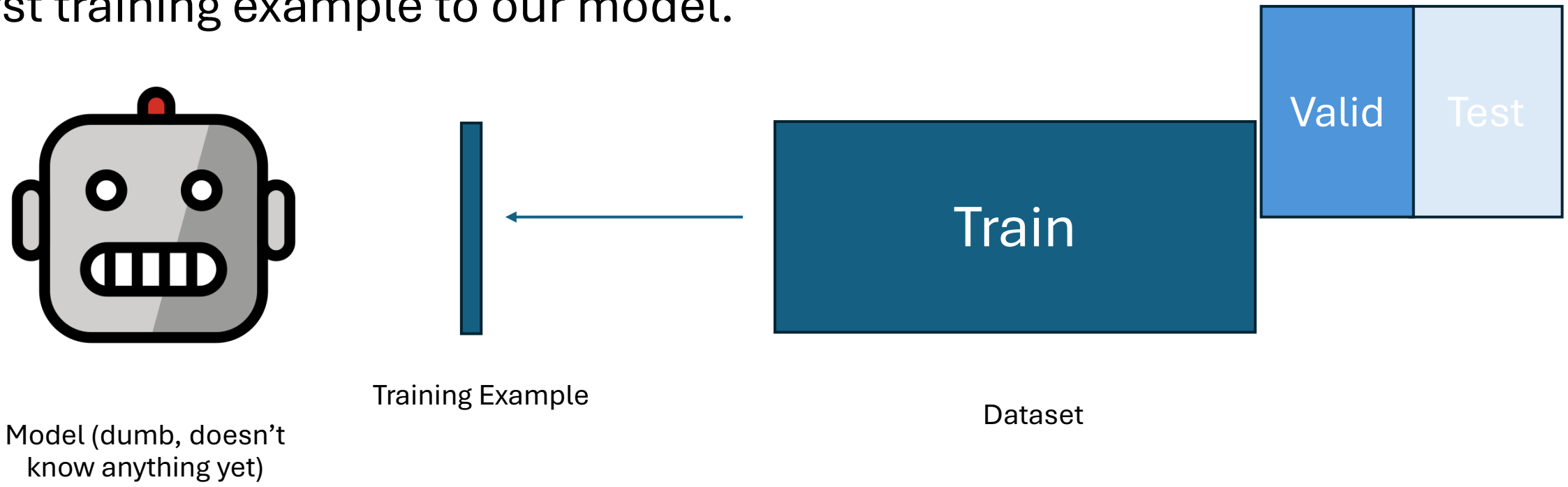


Model (dumb, doesn't know anything yet)

Dataset

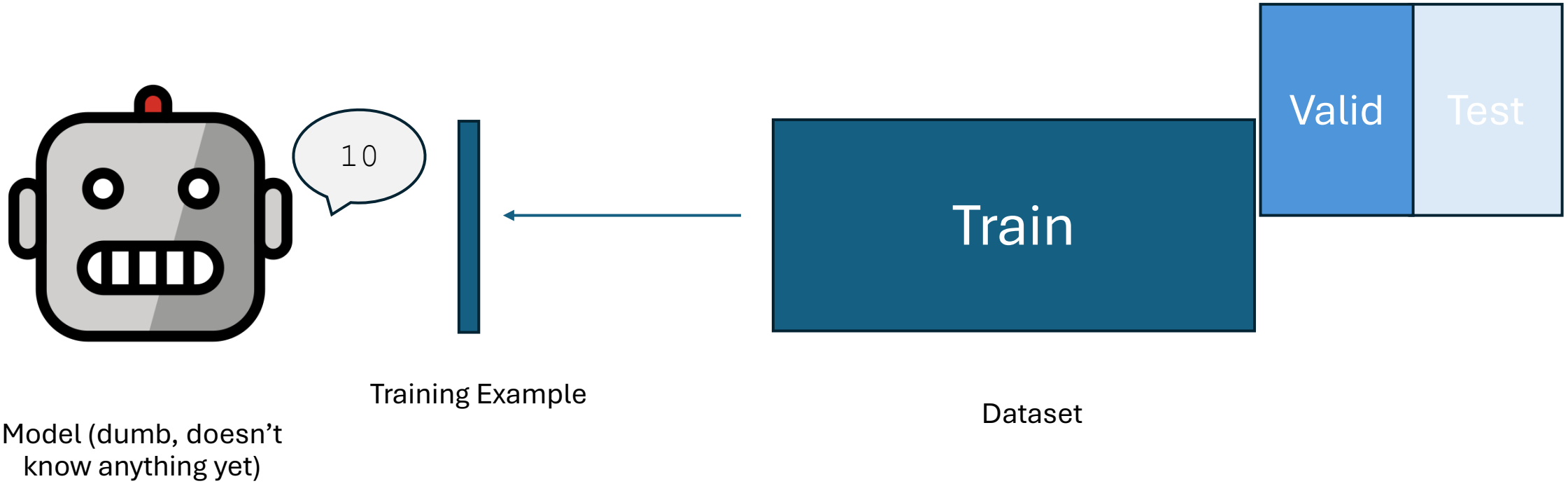
Error, Loss, Evaluation Example

Now, we put our validation and test set on the shelf and show our first training example to our model.



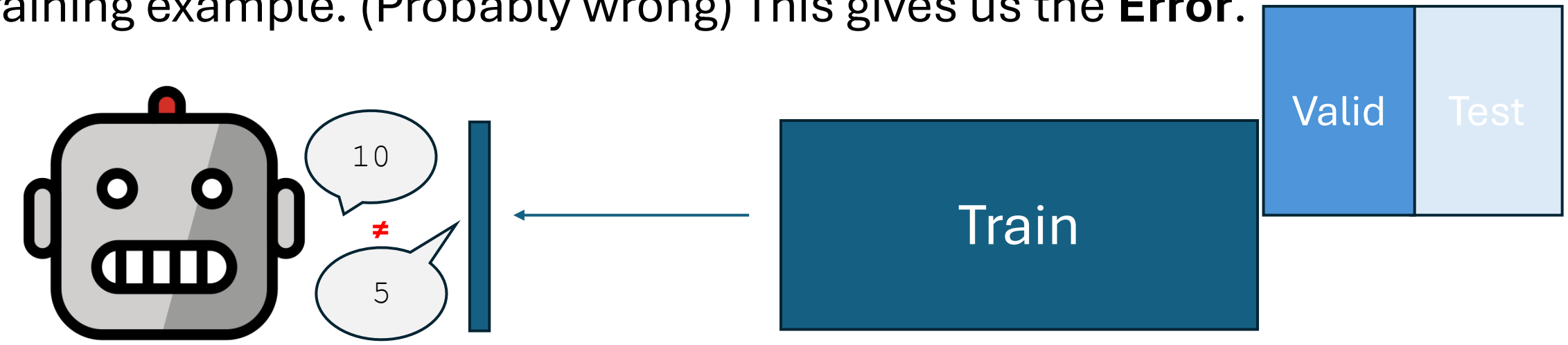
Error, Loss, Evaluation Example

Our model makes a prediction about this training example.



Error, Loss, Evaluation Example

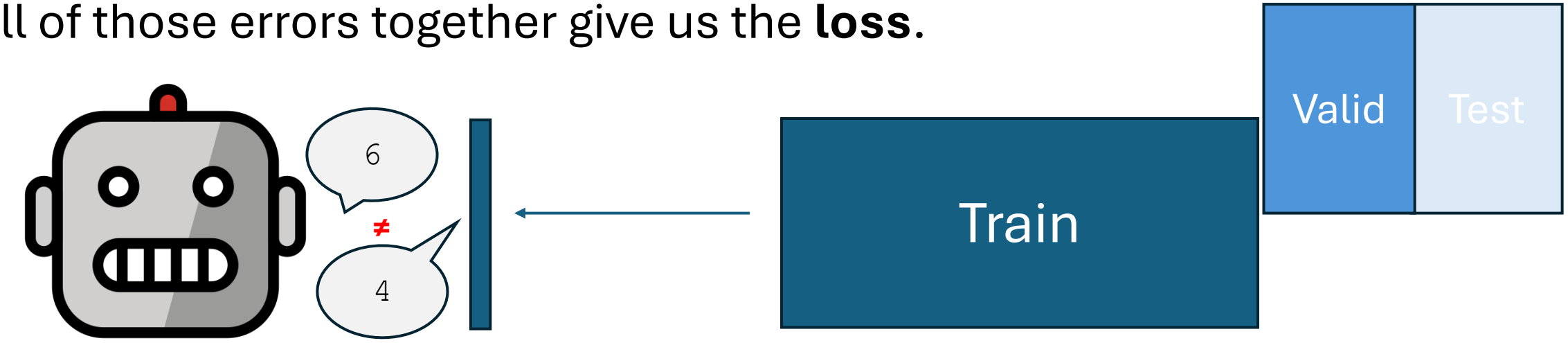
We then compare the model's prediction with the value of the training example. (Probably wrong) This gives us the **Error**.



Error, Loss, Evaluation Example

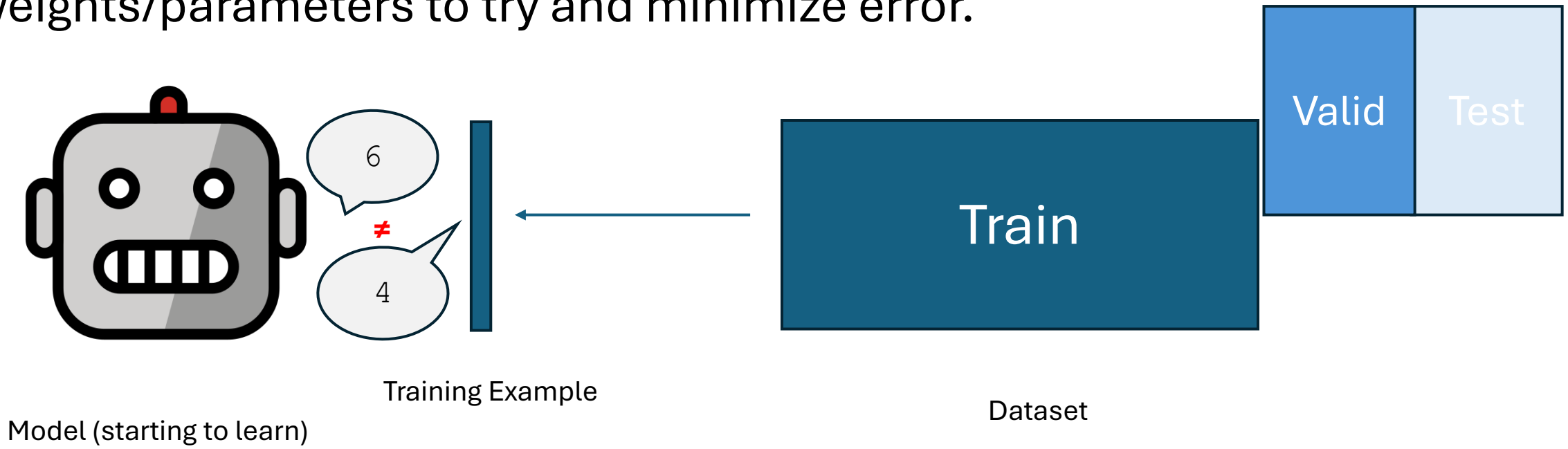
We repeat that process for every example in the training dataset.

All of those errors together give us the **loss**.



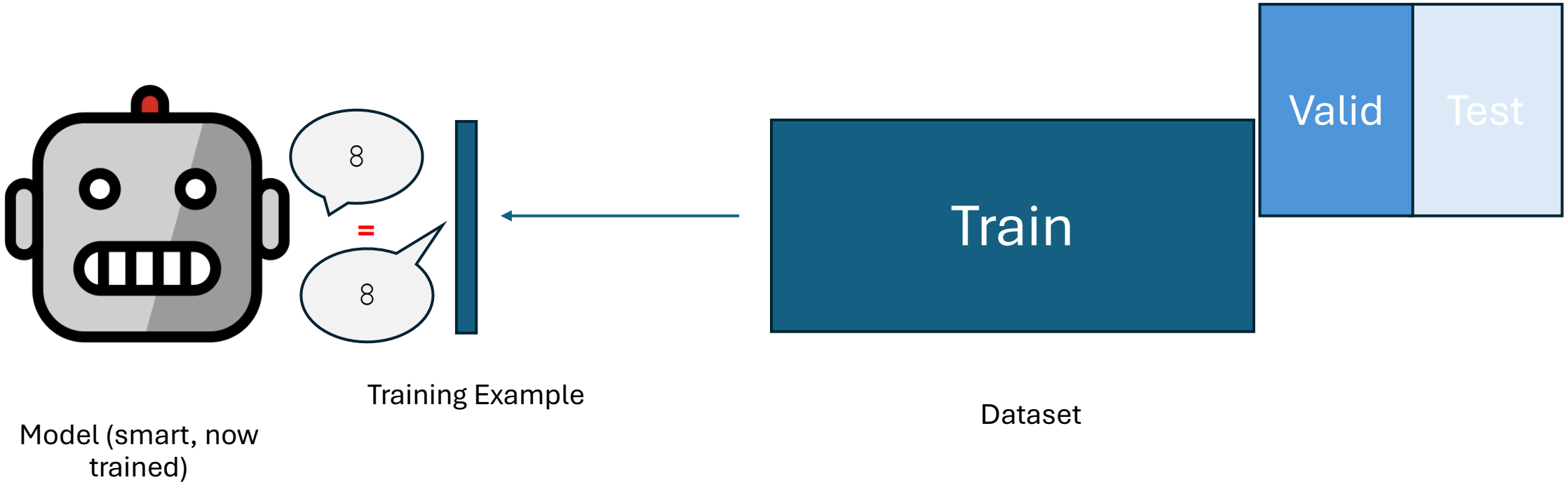
Error, Loss, Evaluation Example

After each training example, the model tweaks its weights/parameters to try and minimize error.



Error, Loss, Evaluation Example

Doing this over the dataset minimizes the **loss.**



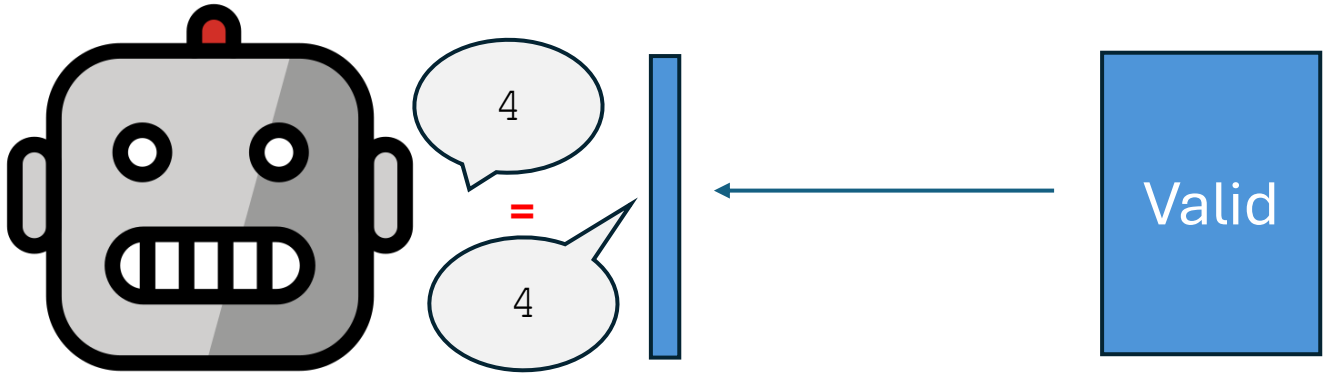
Error, Loss, Evaluation Example

Train

Now, we want to see if our model can generalize to unseen data.

We **evaluate** our model on the validation set.

Test



Model (smart, now trained)

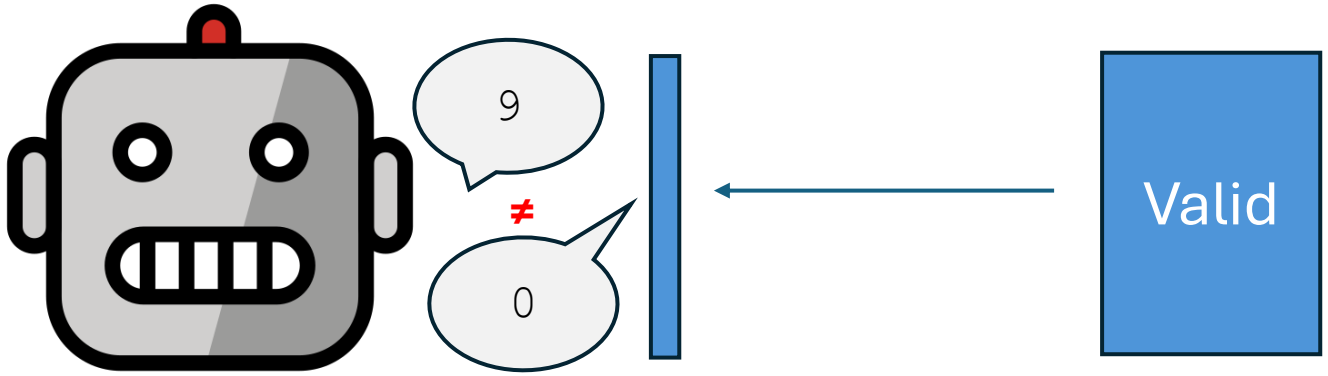
Training Example

Dataset

Error, Loss, Evaluation Example

Train

If our model does poorly, we can make changes. Then we can re-train the model on the train set and evaluate it again on the validation set. Here, we can tweak hyperparameters, choose different models, different loss functions, etc.



Model (smart, now trained)

Training Example

Valid

Dataset

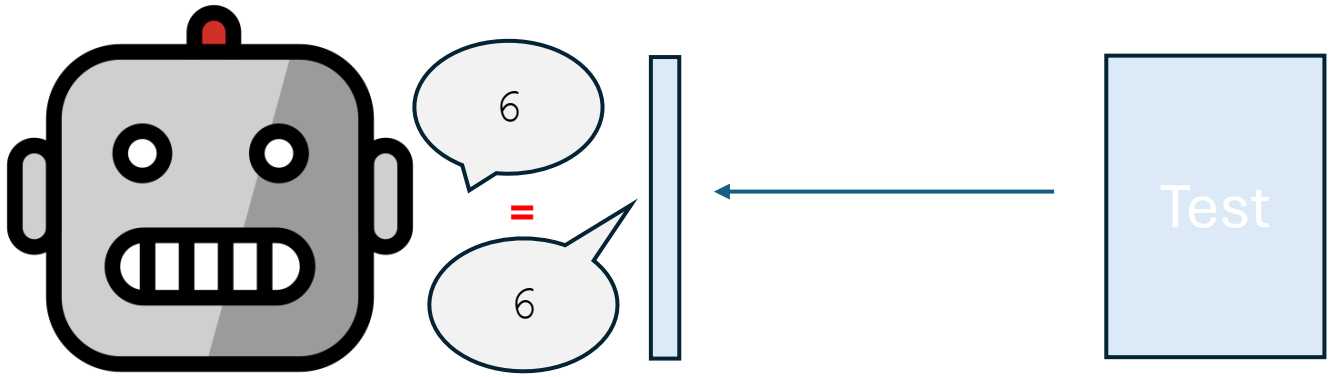
Test

Error, Loss, Evaluation Example

Train

Once we're happy with our model, we finally evaluate it on the Test set.

This is one final test of the model's generalizability.



Valid

Model (Final version)

Training Example

Dataset

Worked Example

- Do you have any questions about train, validation, and test sets?
- Do you have any questions about error and loss?
- Do you have any questions about loss vs evaluation?

KNN Regression

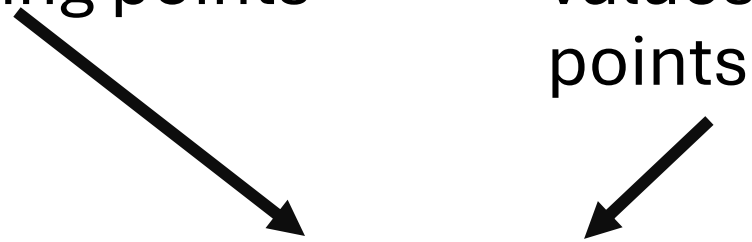
KNN Classification / Regression

Classification

- Predict class from surrounding points

Regression

- Predict value from values of surrounding points



x	*y*	*Class Label*	*Regression Label (value)*
2	3	A	8
5	3	B	5
5	4	B	3
5	5	A	1.2

x: Exercise sessions/week
y: Meals eaten/day
Class Label (A/B): Healthy/unhealthy
Regression label: Health Satisfaction

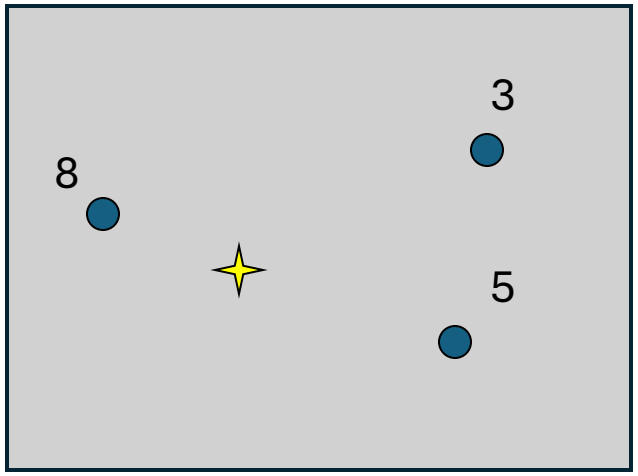
Regression with *k*-nn

- We can do regression by letting the output (regression value of the new point) be the mean of the values of the *k* nearest neighbors.
- Can also do weighted regression by letting the output be the weighted mean of the *k* nearest neighbors.
- For distance weighted regression

$$\hat{f}(x_q) = \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i} \qquad w_i = \frac{1}{dist(x_q, x_i)^2}$$

  - Where $f(x)$ is the output value for instance x
  - $w = 1$ for non-weighted

Regression Example



$$\hat{f}(x_q) = \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

$$w_i = \frac{1}{dist(x_q, x_i)^2}$$

- What is the value of the new instance?
- Assume dist(x_q, n_8) = 2, dist(x_q, n_5) = 3, dist(x_q, n_3) = 4

- $f(x_q) = (8/2^2 + 5/3^2 + 3/4^2)/(1/2^2 + 1/3^2 + 1/4^2) = 2.74/.42 = 6.5$

- The denominator renormalizes the value

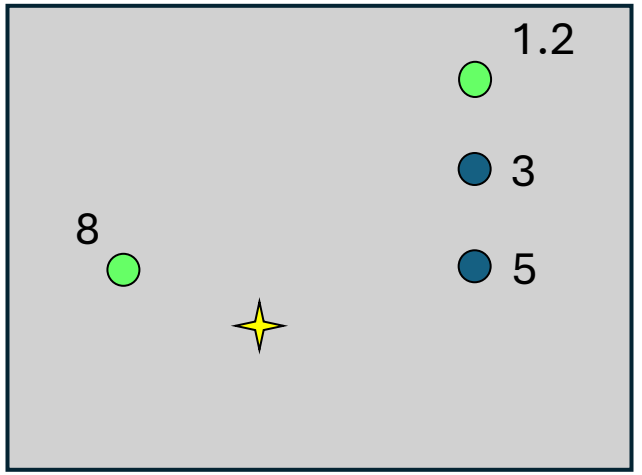
Regression Example 2

Point is at 3,2, k=3

3 Nearest Neighbors are labeled
A, B, B
majority rules B

Weighted votes would be
A=0.25, B = 0.11, B = 0.0625
A

Regression Value = (8 + 3 + 5)/3
=16/3 = 5.33



x: Exercise sessions/week
y: Meals eaten/day
Class Label (A/B): Healthy/unhealthy
Regression label: Health Satisfaction

Weighted Regression Value = (2 + 5/9 + 3/16) / (1/4 + 1/9 + 1/16)
= 2.743 / 0.42 = 6.53

x	*y*	*Class Label*	*Regression Label*	*Distance*	*Weighted Vote*	*Weighted Regression Value*
2	3	A	8	2	$1/2^2 = 1/4$	8*1/4 = 2
5	3	B	5	3	$1/3^2 = 1/9$	5*1/9 = 5/9
5	4	B	3	4	$1/4^2 = 1/16$	3*1/16 = 3/16
5	5	A	1.2	5	$1/5^2 = 1/25$	1.2*1/25 = 1/30

Linear Regression Refresher

Linear Regression

- In linear regression, we try to fit a hyperplane (line in 2-d case) to data.
- We calculate the residuals (distance between the line and each point).
- We want to minimize the residuals.
- We usually square the residuals.

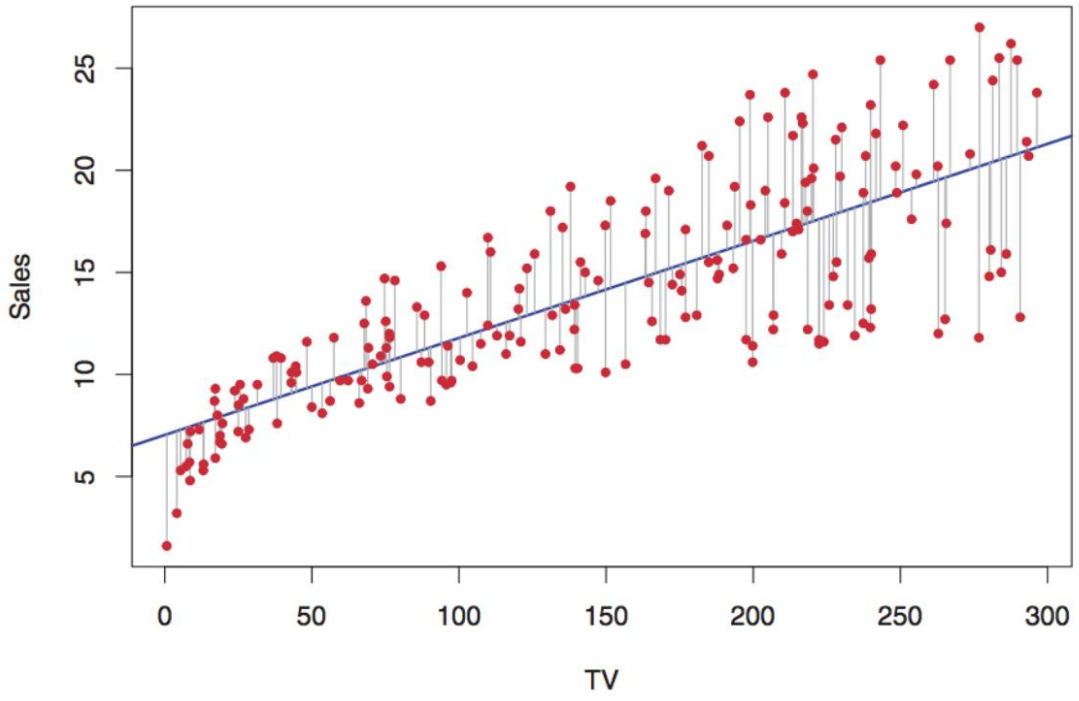


FIGURE 3.1, ISL (8th printing 2017)

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.
- Which one do we usually use?

$$\Sigma|t_j - z_j|$$

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

y dependent variable (output)

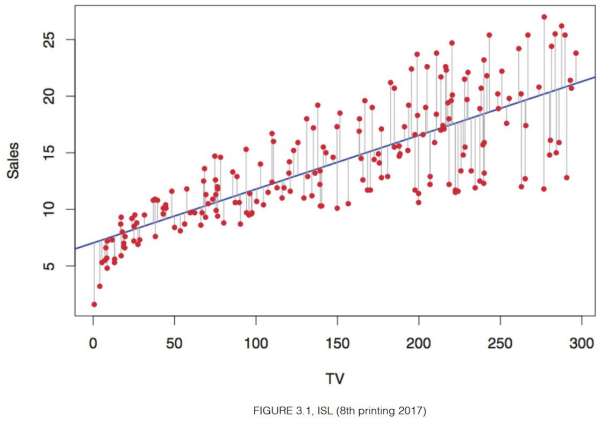


FIGURE 3.1, ISL (8th printing 2017)

x – independent variable (input)

Regularization (Second Try)

Regularization

- What is regularization?
  - Last time, my definition left some people scratching their heads.
- OLD DEFINITION:
  - Regularization is *anything we do* to improve our model's *generalization accuracy* without improving our training performance.
- NEW DEFINITION:
  - Regularization is a technique used to make a model simpler.
  - OR Regularization is a technique used to prevent a model from growing too complex.
- The goal of regularization is to improve generalization accuracy by preventing overfitting.
- The goal of regularization is **not** to improve the model's performance on the training set. (The loss of a perfectly overfit model is zero)
- In fact, by preventing overfitting, a regularized model might perform worse on the training set.
- But it might do better on the test set.

Regularization Continued

- How is regularization performed?
- Explicit Regularization:
  - Add a term to the model.
    - Example from last time: adding a penalty term to the loss function (Ridge and Lasso regression)
  - We are giving models an **Inductive Bias** when we do this.
- Implicit Regularization:
  - Eliminate outliers from data
  - Early stopping with Neural Nets

- For now, we're focusing on explicit, penalty-based regularization

Penalty-Based Regularization

- **Normal Training:** The model tries to minimize error at all costs, often memorizing noise in the training data by creating complex, high-magnitude weights.

- **Regularized Training:** The model must balance two competing goals:
  - Minimize the error (fit the data).
  - Minimize the complexity (keep weights small).

Penalty-Based Regularization

- **Normal Training:** The model tries to minimize error at all costs, often memorizing noise in the training data by creating complex, high-magnitude weights.

- **Regularized Training:** The model must balance two competing goals:
  - Minimize the error (fit the data).
  - Minimize the complexity (keep weights small).

Reminder: What are Weights?

- **Weights** are the things the model is learning (tunable parameters)
- In the case of multiple linear regression, it's all of those ws.

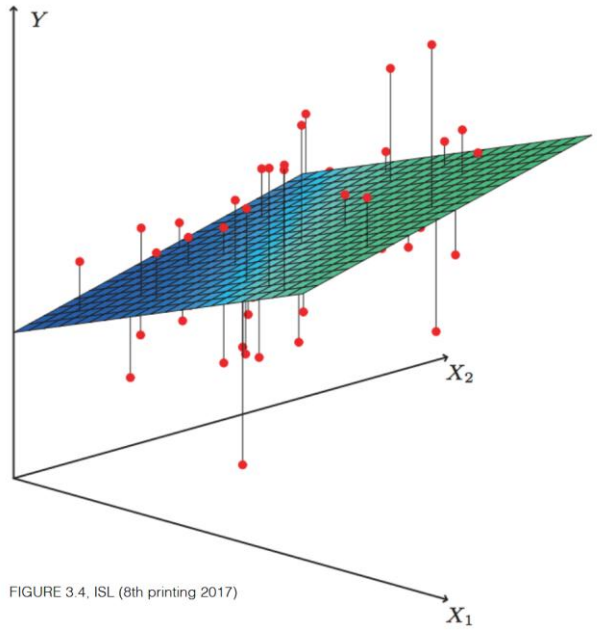


FIGURE 3.4, ISL (8th printing 2017)

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

Penalty-Based Regularization for Linear Regression

- We discussed three types of regularization last class:
  - Ridge (L2) Regression
  - Lasso (L1) Regression
  - Elastic Net Regression
- Let's review

No Regularization

SSE

Minimize the error (fit the data)

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

- Standard Loss for Linear Regression.
- Is this L1 or L2 Loss?

- It would be better if we had the model balance two competing goals:
  - Minimize the error (fit the data).
  - Minimize the complexity (keep weights small).

Ridge (L2) Regularization

SSE

Minimize the error (fit the data)

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

Regularization Term

Minimize the complexity (keep weights small).

- 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

Minimize the error (fit the data)

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

Regularization Term

Minimize the complexity (keep weights small).

- 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

Minimize the error (fit the data)

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

Regularization Terms

Minimize the complexity (keep weights small).

- 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.
- Best-of-both-worlds scenario.
- We have two hyperparameters, one for each regularization term.
  - This lets us choose how much of each type of regularization we want.

Break Time!


nick got his O1
@spreen_co
for anyone who's wondered what an apple intelligence summary of a breakup text looks like
Maddi Broke
1m ago
No longer in a relationship; wants belongings from the apartment.