Model Learning, Validation, and Overfitting

May 25, 2019

Lecture 3, Applied Data Science
MMCi Term 4, 2019

Matthew Engelhard

MODEL LEARNING

Learning (or *training*) our model means:

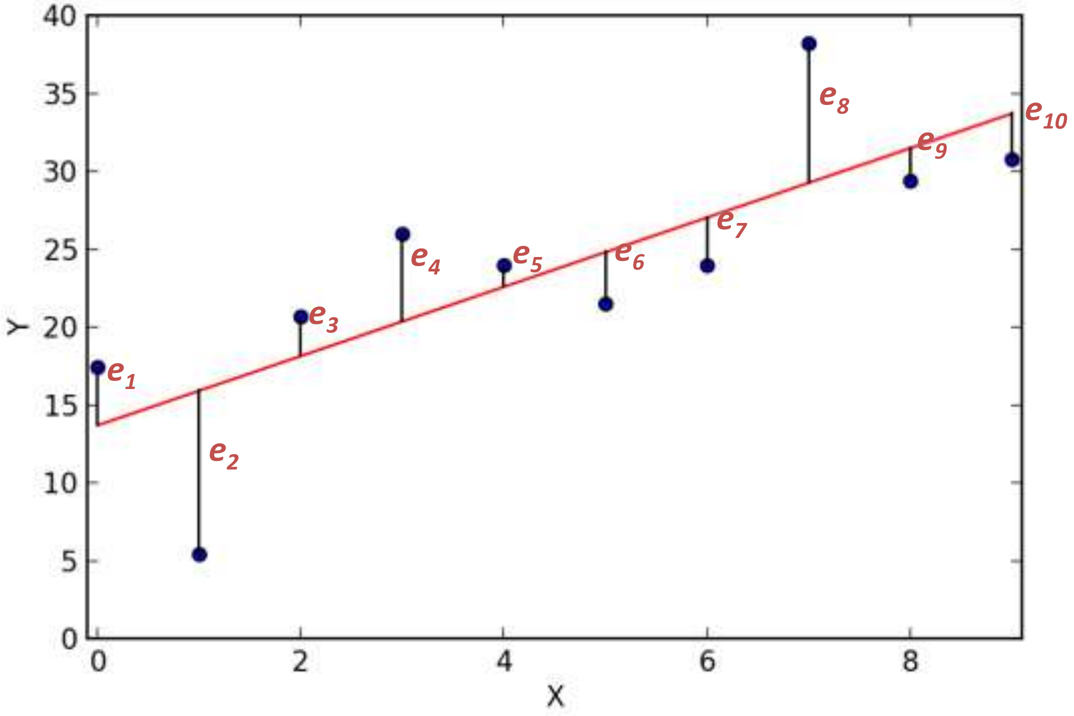
-> Setting our parameters to the specific values that are the best match for our training data

What do we mean by “best match”?

-> We set our parameters to the values that maximize some measure of fit

-> Alternatively, we set them to values that minimize a penalty (i.e. a *loss*) that we choose

Linear Regression Measure of Fit: Mean Squared Error



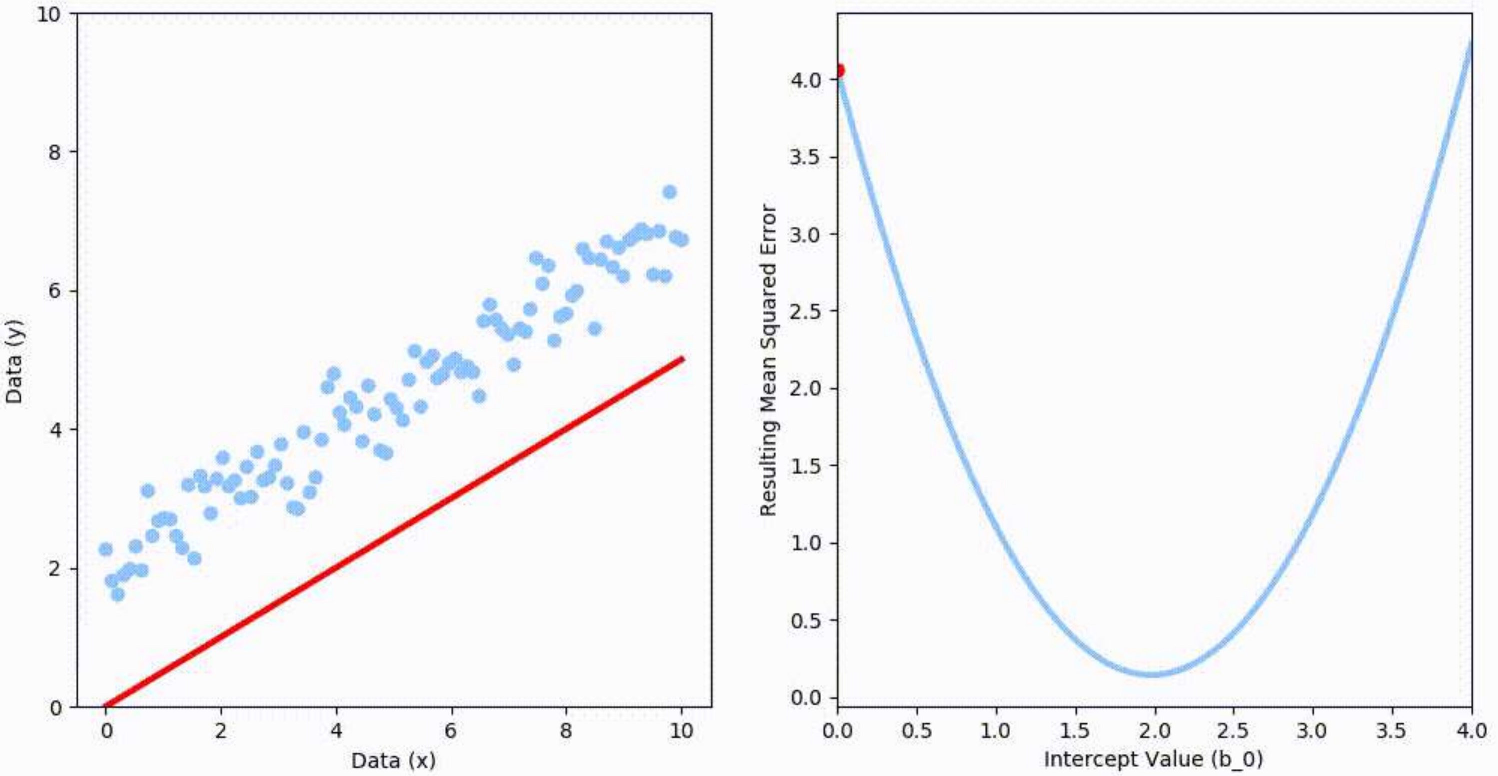
Error:

$$e_i = y_i - \hat{y}_i$$

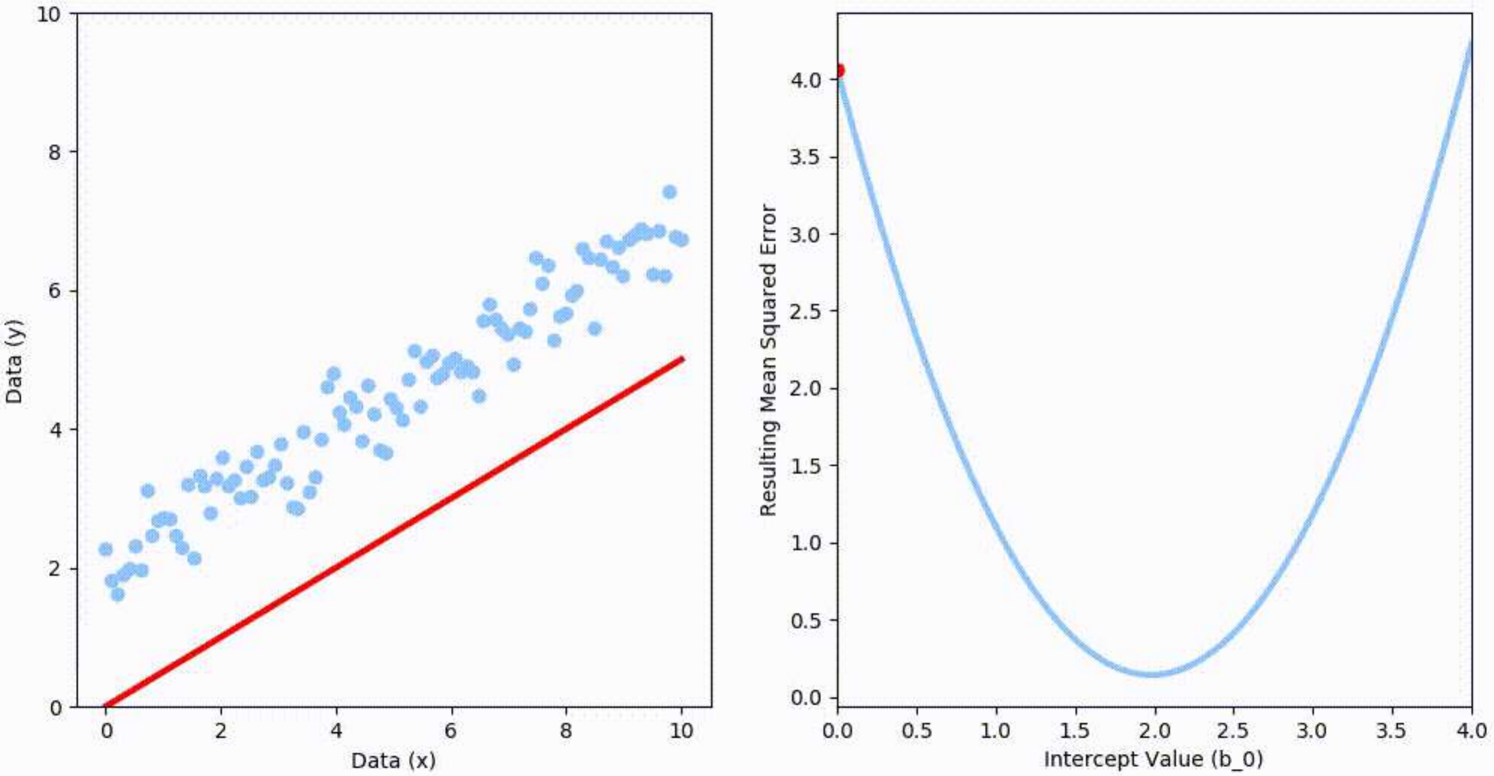
Mean square error (MSE)

$$\frac{1}{N}\sum_{i=1}^{N} e_i^2$$

Suppose the slope (b_1) is known.
What happens to the MSE as we move the intercept (b_0)?

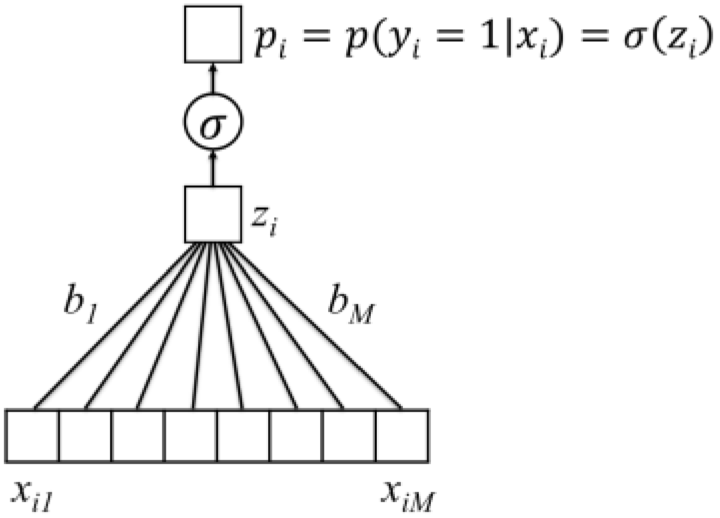


What is the best choice of intercept (b_0) for these data?



Logistic Regression Measure of Fit:

Probability that events $y_1, \ldots, y_N$ would have occurred if our model were correct

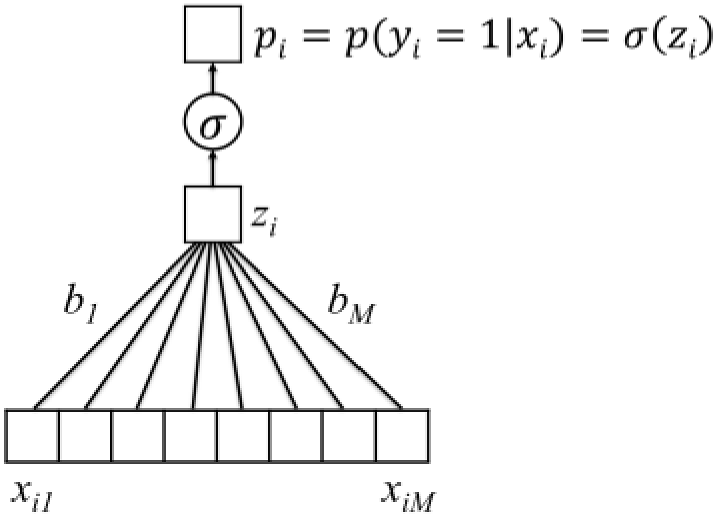


Probability that event y_i would have occurred if our model were correct:

$$\left(\sigma(z_i)\right)^{y_i}\left(1-\sigma(z_i)\right)^{(1-y_i)}$$

Logistic Regression Measure of Fit:

Probability that events $y_1, \ldots, y_N$ would have occurred if our model were correct

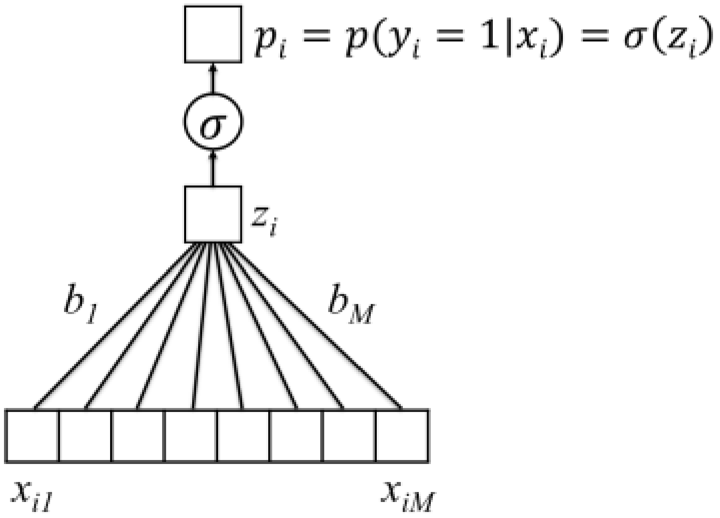


Probability that event y_i would have occurred if our model were correct:

$$\underbrace{\left(\sigma(z_i)\right)^{y_i}}_{\begin{cases} p_i, & y_i = 1 \\ 1, & y_i = 0 \end{cases}}\left(1-\sigma(z_i)\right)^{(1-y_i)}$$

Logistic Regression Measure of Fit:

Probability that events $y_1, \ldots, y_N$ would have occurred if our model were correct

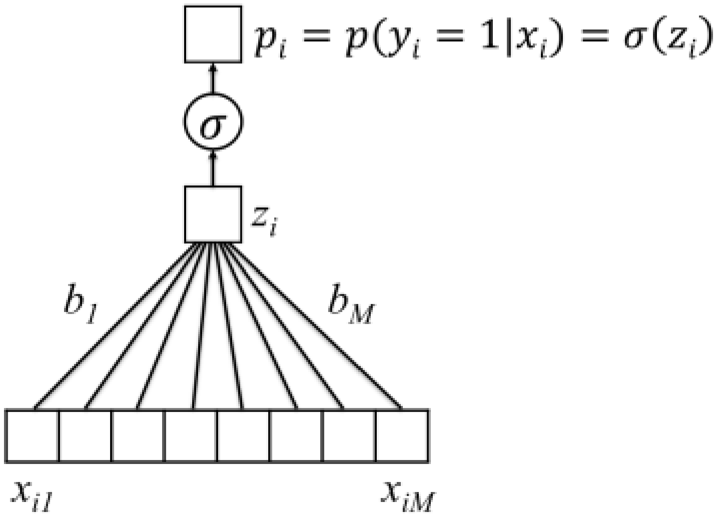


Probability that event y_i would have occurred if our model were correct:

$$\left(\sigma(z_i)\right)^{y_i}\underbrace{\left(1-\sigma(z_i)\right)^{(1-y_i)}}_{\begin{cases} 1, & y_i = 1 \\ 1 - p_i, & y_i = 0 \end{cases}}$$

Logistic Regression Measure of Fit:

Probability that events $y_1, \ldots, y_N$ would have occurred if our model were correct



Probability that event y_i would have occurred if our model were correct:

$$\left(\sigma(z_i)\right)^{y_i}\left(1-\sigma(z_i)\right)^{(1-y_i)}$$

Probability that events $y_1, \ldots, y_N$ would have occurred if our model were correct:

$$\prod_{i=1}^{N}\left(\sigma(z_i)\right)^{y_i}\left(1-\sigma(z_i)\right)^{(1-y_i)}$$

Logistic Regression Measure of Fit:

Probability that events $y_1, \ldots, y_N$ would have occurred if our model were correct

Maximize the probability that events $y_1, \ldots, y_N$ would have occurred if our model were correct:

$$\prod_{i=1}^{N} \left(\sigma(z_i)\right)^{y_i} \left(1 - \sigma(z_i)\right)^{(1-y_i)}$$

Easier to maximize the log of this quantity (i.e. the log-likelihood):

$$\sum_{i=1}^{N} y_i \log \sigma(z_i) + (1 - y_i) \log\left(1 - \sigma(z_i)\right)$$

Maximize the log-likelihood = minimize the "cross-entropy" loss

Log-likelihood:

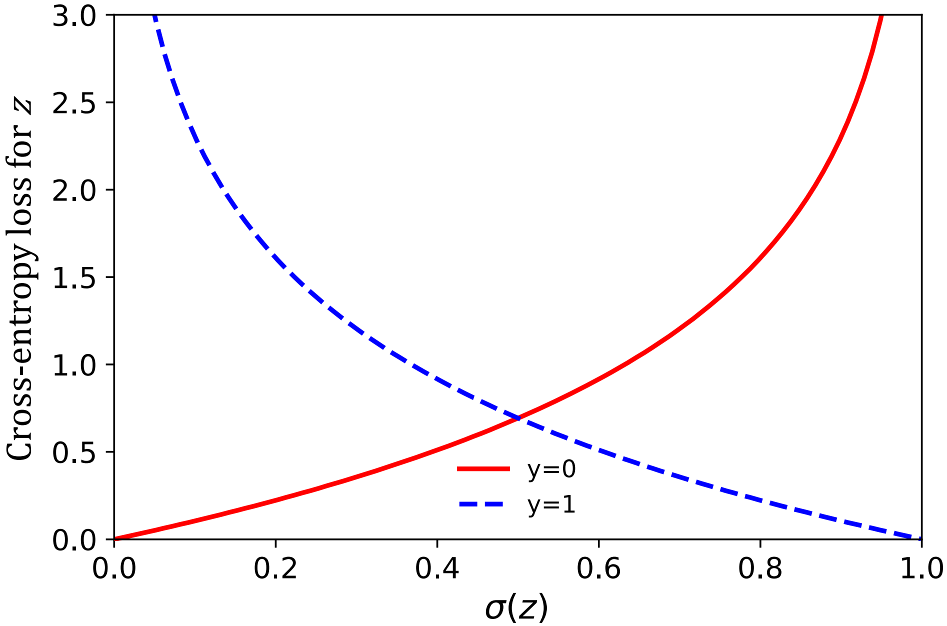
$$\sum_{i=1}^{N} y_i \log \sigma(z_i) + (1 - y_i) \log\big(1 - \sigma(z_i)\big)$$

Maximize the log-likelihood = minimize the "cross-entropy" loss

Cross-entropy loss:

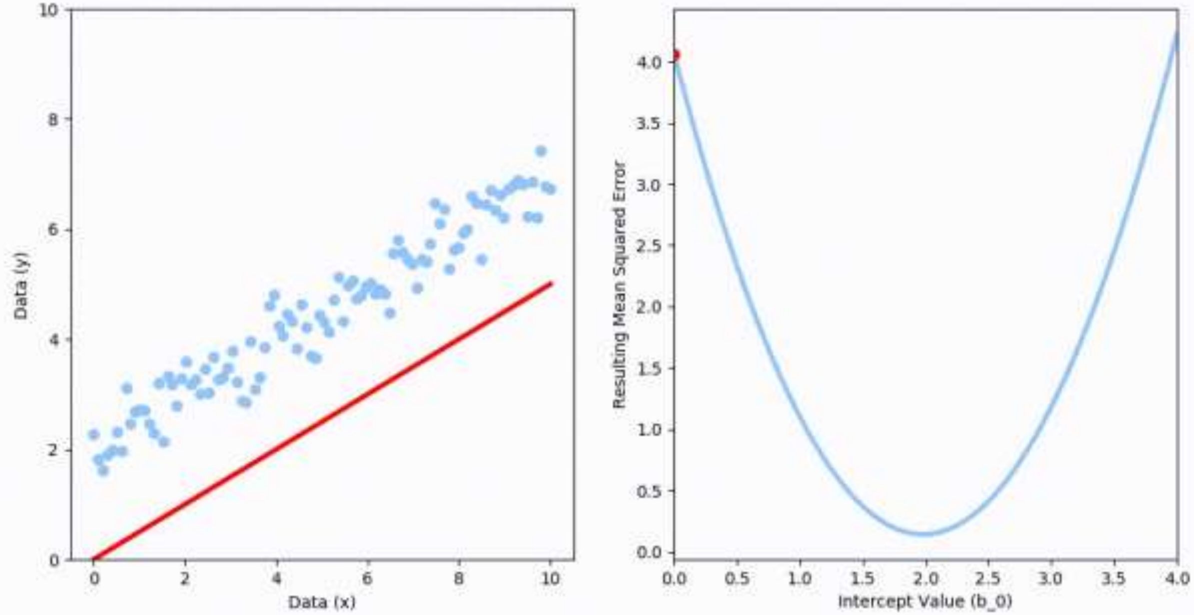
$$\sum_{i=1}^{N} -y_i \log \sigma(z_i) - (1 - y_i) \log\big(1 - \sigma(z_i)\big)$$

The cross-entropy loss is also used for other classification models, including convolutional neural network classifiers for image processing.



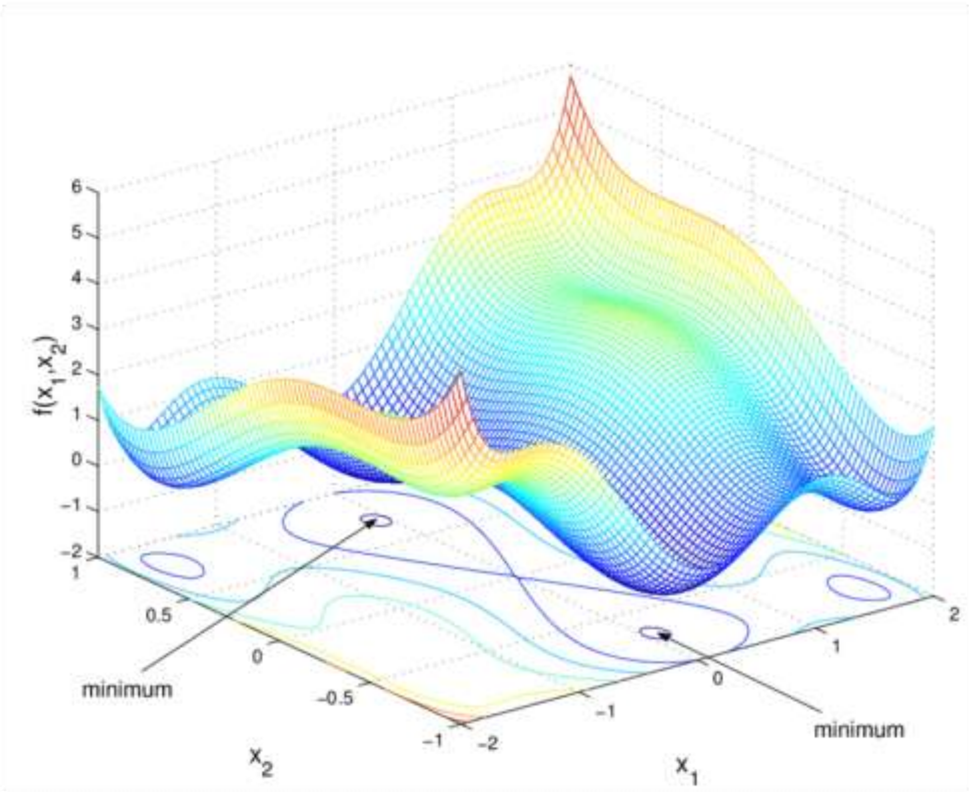
Learning: find parameters that minimize the loss

Easy!!



Learning: find parameters that minimize the loss

Easy?



Learning: find parameters that minimize the loss

Not easy.

Learning: find parameters that minimize the loss

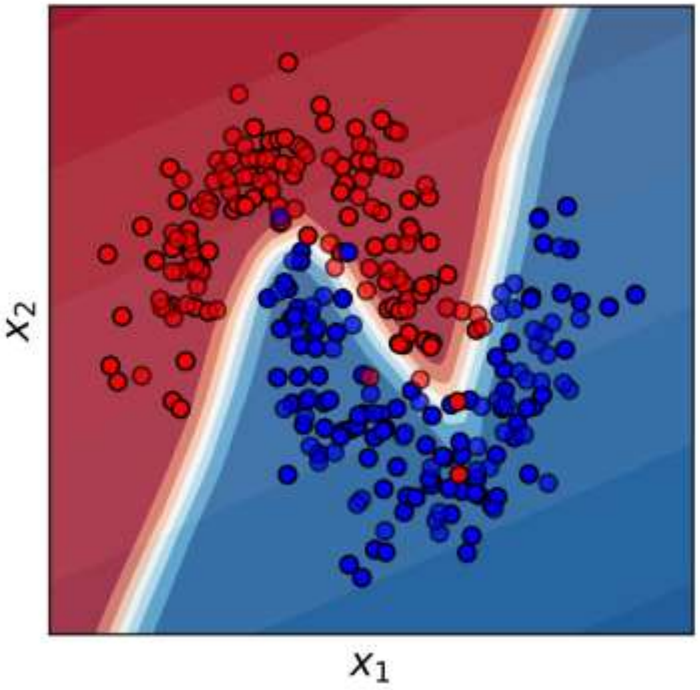
- With deep learning models, we are trying to minimize a function of many variables that we can't visualize

- It may have many "local minima"

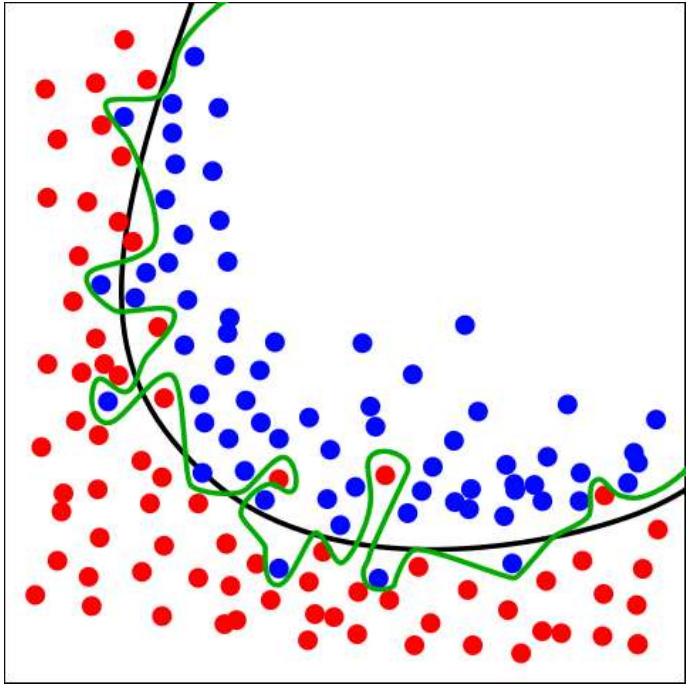
- Best we can do: follow the slope (i.e. gradient descent)

OVERFITTING

We like flexible, non-linear decision boundaries…



But some models can be *too* flexible.



By Chabacano - Own work, CC BY-SA 4.0,
https://commons.wikimedia.org/w/index.php?curid=3610704

Green boundary:
- This is overfitting

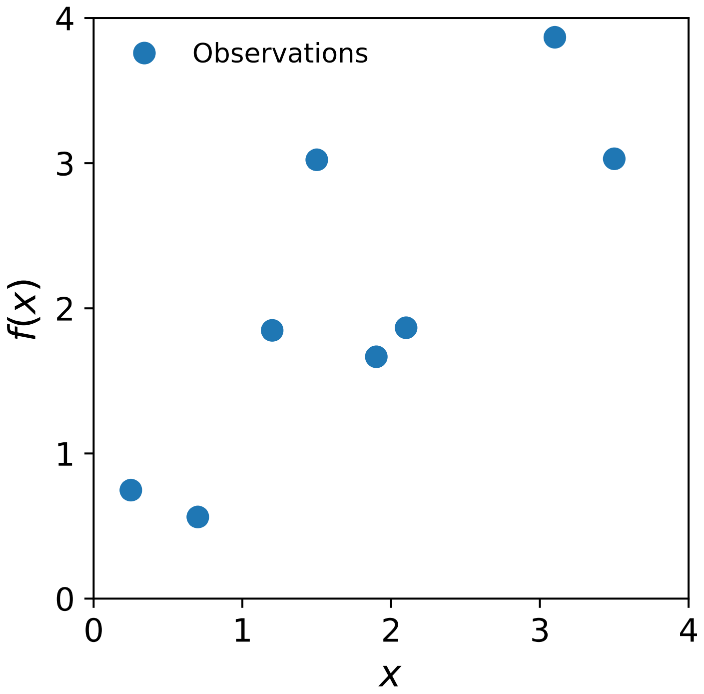
Black boundary:
- Balance between fit and model complexity

-> The black boundary is likely to perform better on new data

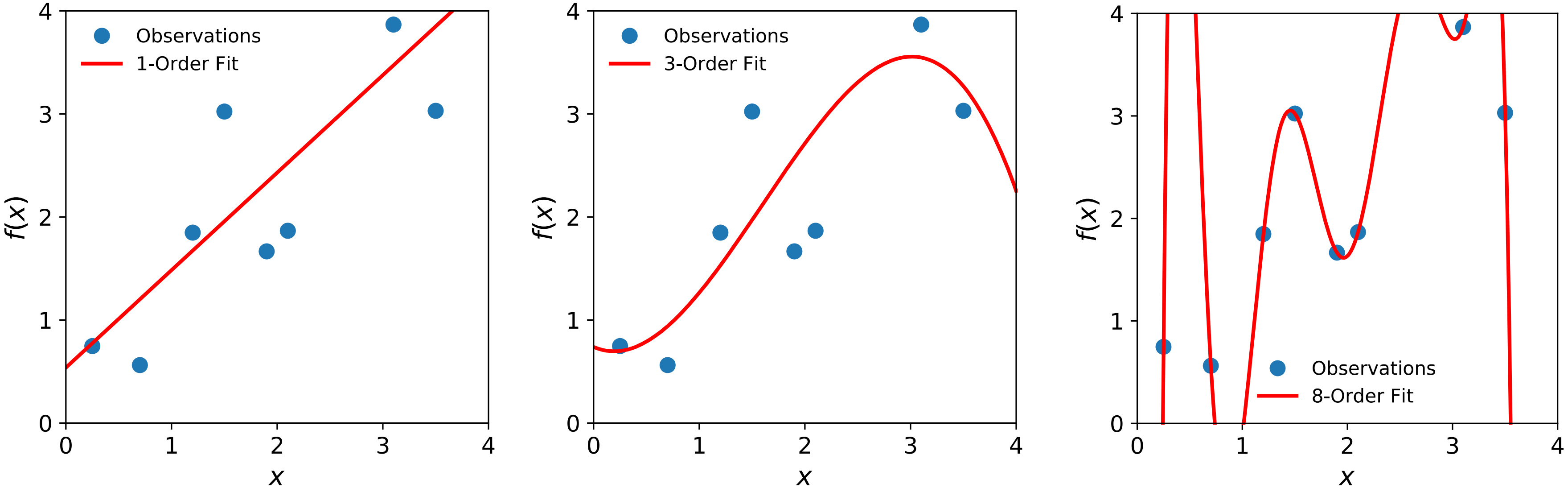
Overfitting

“Overfitting” happens when the learned model increases complexity to fit the observed training data *too well* – will not work to predict future data!

What would we want to use to fit these example data points?

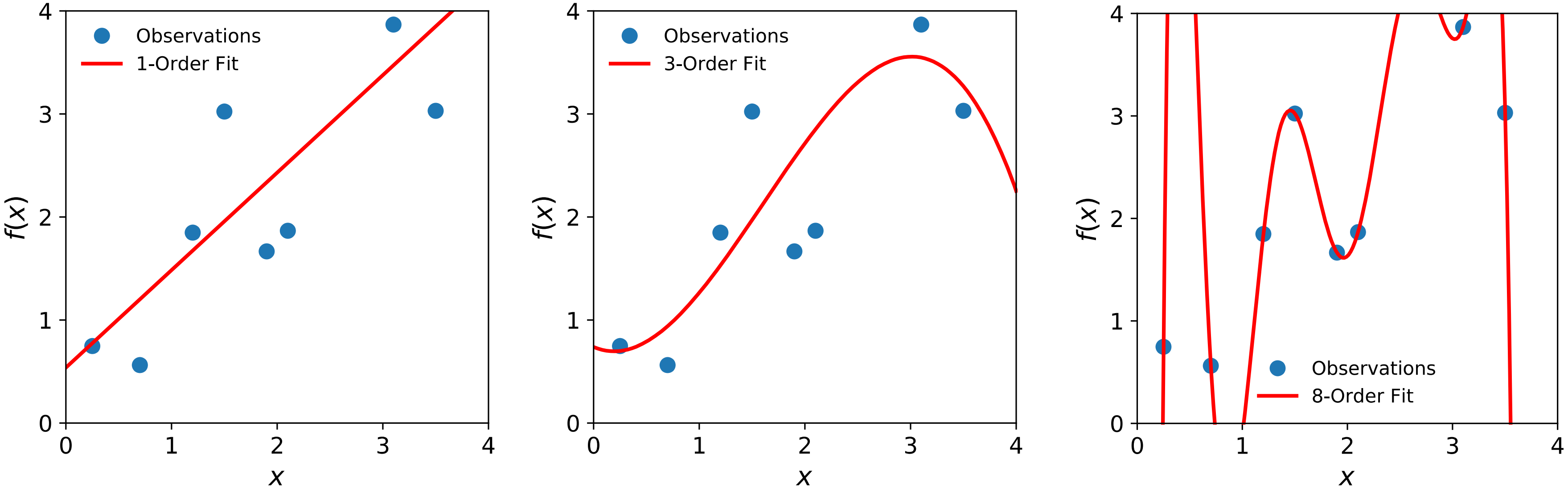


Classic Example: Increasing Polynomial Order



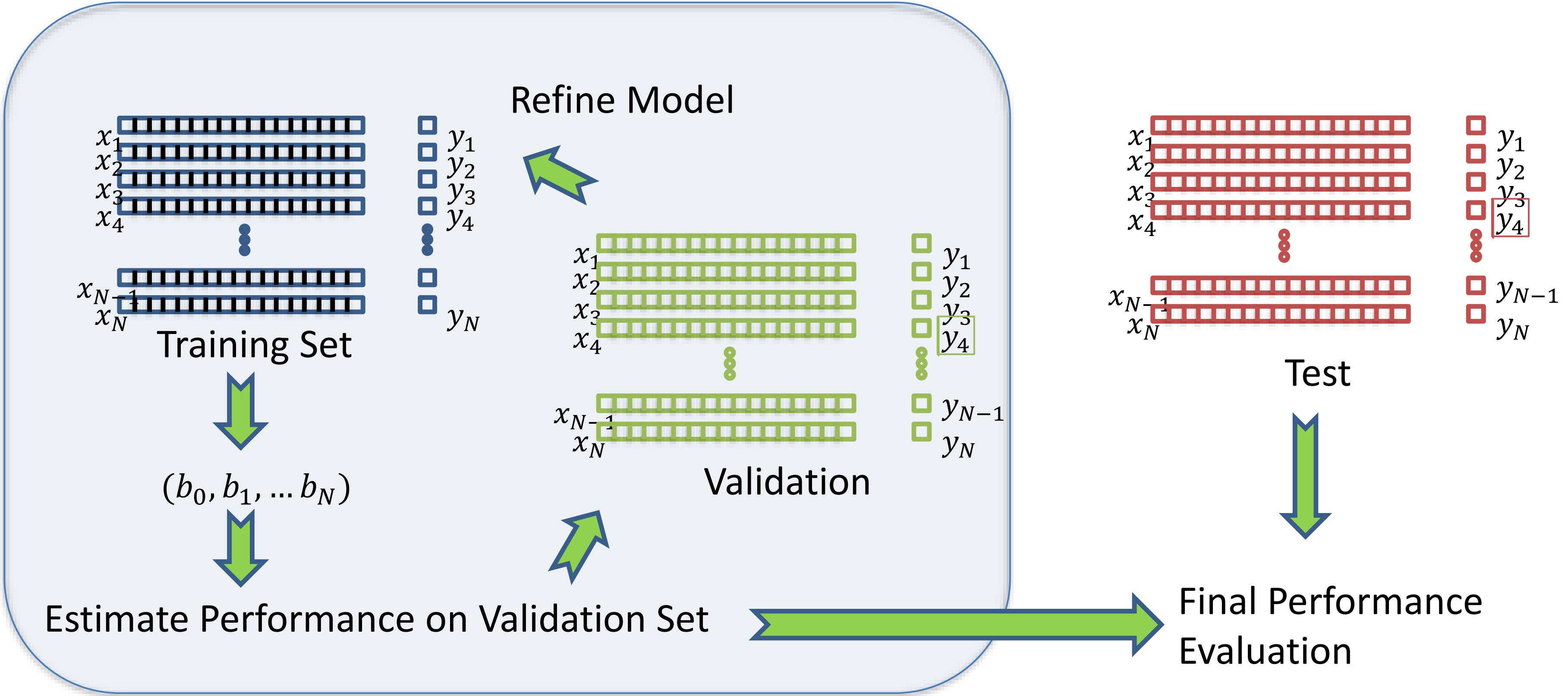
Increasing complexity does not seem appropriate…

With a flexible enough model, we can typically reach 100% accuracy on our training set



Increasing complexity does not seem appropriate…

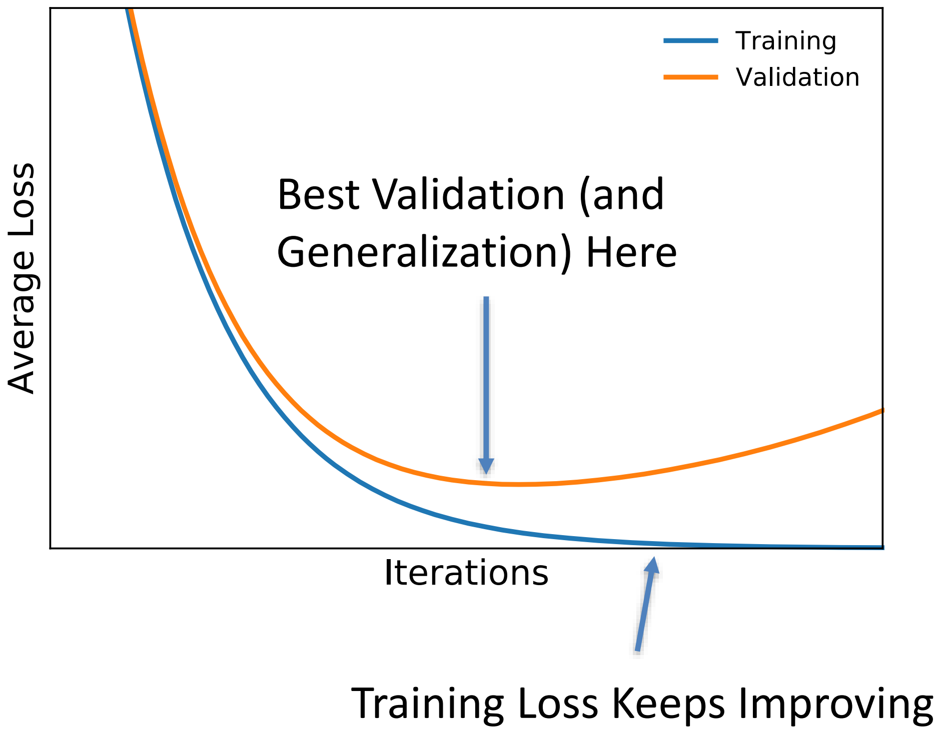
Using a Test Set Protects Against Overfitting: See how the model performs on *new* data



Early Stopping

- During optimization, we can check the validation loss as we go.

- Instead of optimizing to convergence, we can optimize until the *validation* loss stops improving
  - Saves computational cost
  - Performs better on validation (and test) sets
- Widely used technique in the field



Conclusions

- *Learning* consists in setting model parameters to maximize some measure of fit (or equivalently, minimize some loss)
- This sounds easy, but is difficult in practice when working with complex models
- Greater model complexity is often, but not always, advantageous
- Proper model validation is critical to estimate real-world performance and prevent overfitting