"By 1913, a new wave of [financial] problems was threatening the university's future. BYU faced mounting debt. Faculty salaries were so low the teachers ran farms to survive, returning home to irrigate between classes. [Its cornerstone laid in 1907,] the Maeser Memorial Building sat silent and unfinished for years. Finally, it seemed, the only way to finance its completion was to divide the land on Temple Hill into housing lots and sell them. A student named Alfred Kelly was selected to promote this idea during a commencement speech, but the assignment troubled him. Early one morning he walked to the top of Temple Hill to pray. What he saw that morning as he looked out across the valley left an unforgettable impression upon all who heard him relate it the day of his address, because what Kelly saw was you.

"Gradually the morning light advanced across the valley floor toward the spot where I stood. I closed my eyes partially to the advancing light and was startled by the strange vision that seemed to appear before me. The advancing sunlight took on the appearance of people, thousands of young people who approached me, their arms laden with books. I turned around to find the area behind me illuminated as well. In that light I saw hundreds of buildings, large and beautiful temples of learning. Those young people passed by me and entered in. Then, with cheerfulness and confidence, they turned toward the east and lifted their eyes heavenward, where, again becoming part of the sunlight, they gradually disappeared from my view."

Kelly sat down to a stunned silence. Suddenly Jesse Knight leaped to his feet, pledging several thousand dollars to BYU. Others followed [suit.] Eventually, under the direction of President Joseph F. Smith, the Church assumed the school's remaining debt. Finally, the future of the university had become secure.

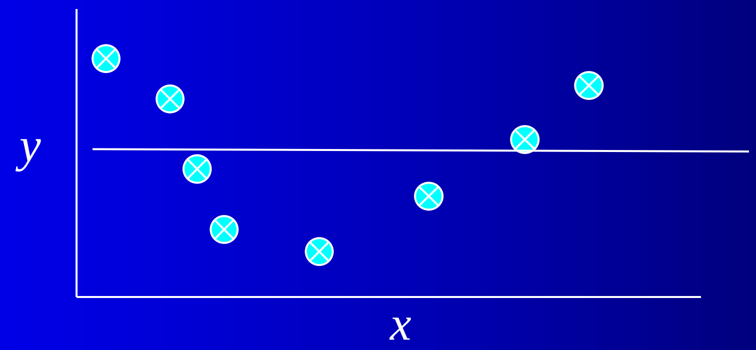
# Overfit and Inductive Bias: How to generalize on novel data

# Non-Linear Tasks

- Linear Regression will not generalize well to the task below
- Needs a non-linear surface – Could use one of our future models
- Could also do a feature pre-process like with the quadric machine
  - For example, we could use an arbitrary polynomial in *x*
  - Thus, it is still linear in the coefficients, and can be solved with delta rule

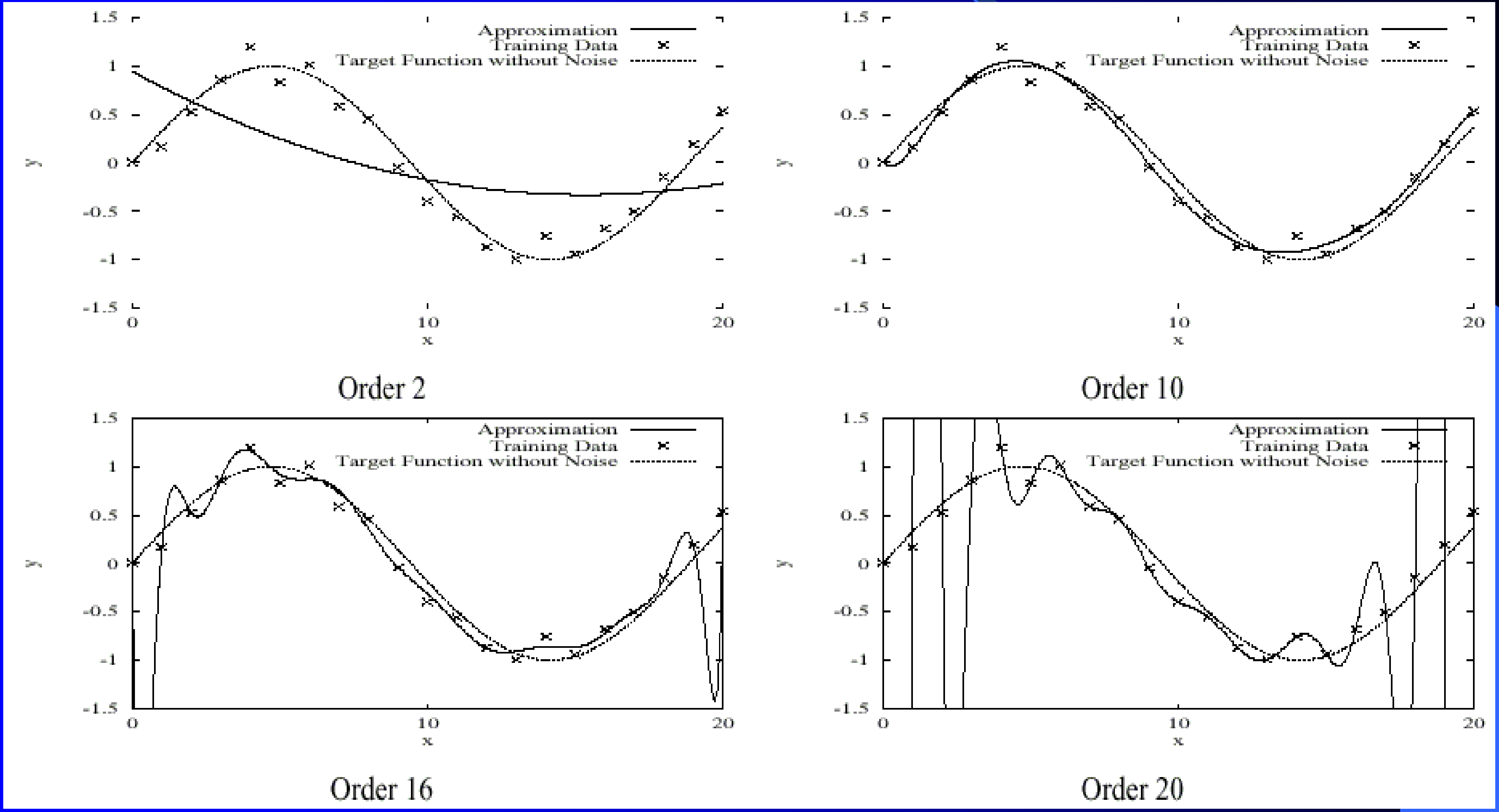
$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \ldots + \beta_n X^n$$

  - What order polynomial should we use? – Overfit issues can occur



# Overfitting

Typically try to learn a model just complex enough to do well and no more complex than that



# Avoiding Overfit

- Regularization: *any modification we make to learning algorithm that is intended to reduce its generalization error but not its training error*
- Occam's Razor – William of Ockham (c. 1287-1347)
  - Favor simplest explanation which fits the data
- General Key: Focus on patterns/rules that really matter and ignore others

- More Training Data (vs. overtraining on same data)
  - Data set augmentation – Fake data, Can be very effective, Jitter, but take care…
  - Denoising – add random noise to inputs during training – can act as a regularizer
  - Adding noise to models. e.g. (Random Forests, Dropout , discuss with ensembles)
- *Early Stopping* – Very common regularization approach: Start with simple model (small parameters/weights) and stop training as soon as we attain good generalization accuracy (and before parameters get large)
  - Common early stopping approach is to use a validation set (next slide)
- We will discuss other model specific approaches with specific models

# Regularization

- Way to avoid overfitting – Keep the model simple
  - E.g., keep decision surfaces smooth
- Make complexity an explicit part of the loss function

- Regularization approach: Model (*h*) selection
  - Minimize $F(h) = Error(h) + \lambda \cdot Complexity(h)$
  - Tradeoff error/accuracy vs complexity
- Two common approaches
  - Lasso (L1 regularization)
  - Ridge (L2 regularization)

Note: often used with regression; applicable to many techniques

# L1 (Lasso) Regularization

- We add a model complexity term to the loss function:

$$L(\vec{w}) = E(\vec{w}) + \lambda \sum |w_i|$$

- The gradient descent is given by:

$$-\nabla L(\vec{w}) = -\nabla E(\vec{w}) - \lambda$$

  - This is also called weight decay
  - Decay magnitude towards 0
  - $\nabla E(\vec{w})$ is just equations we have used if $E(\vec{w})$ is TSS
- Common values for lambda are 0, .01, .03, etc.
- Weights that should be significant stay large enough, but weights just being nudged by a few data instances go to 0

# L2 (Ridge) Regularization

- We add a model complexity term to the loss function:

$$L(\vec{w}) = E(\vec{w}) + \lambda \sum w_i^2$$

- The gradient descent is given by:

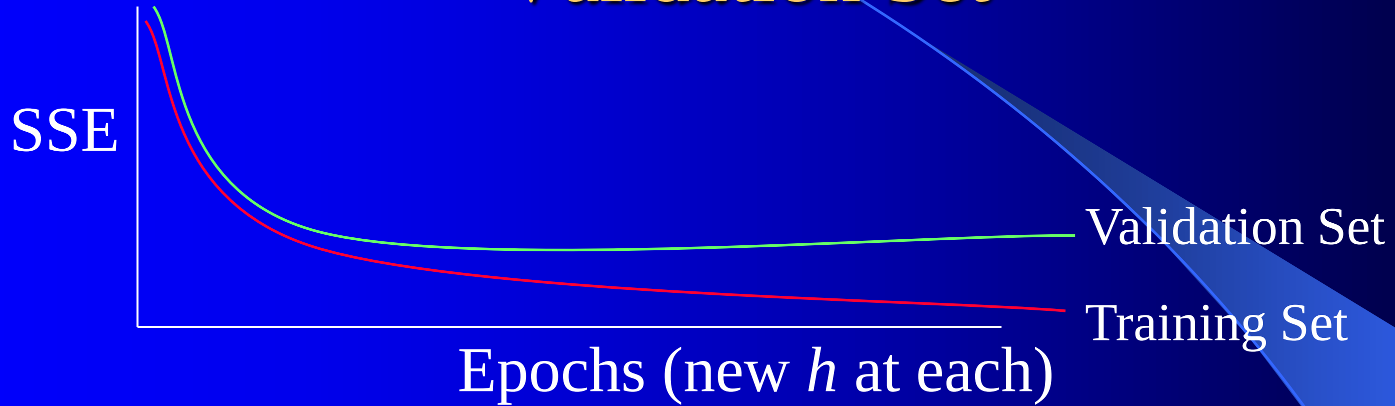
$$-\nabla L(\vec{w}) = -\nabla E(\vec{w}) - 2\lambda w_i$$

- Regularization portion of weight update is scaled by weight value (fold 2 into $\lambda$)
  - Decreases change when weight small (<0), otherwise increases
  - $\lambda$ is % of weight change, .03 means 3% of the weight is decayed each time

# L1 vs. L2 Regularization

- L1 drives many weights all the way to 0 (sparse representation and feature reduction)
- L1 more robust to large weights (outliers), while L2 acts more dramatically with large weights
- L1 leads to simpler models, but L2 often more accurate with more complex problems which require a bit more complexity

# Early Stopping/Model Selection with a Validation Set




- There is a different model *h* after each epoch
- Select a model in the area where the validation set accuracy flattens
- Keep *bssf* (Best Solution So Far). Once you go *w* epochs with no improvement stop and use the parameters at the *bssf* *w* epochs ago.
- The validation set comes out of training set data
- Still need a separate test set to use after selecting model *h* to predict future accuracy
- Simple and unobtrusive, does not change objective function, etc
  - Can be done in parallel on a separate processor
  - Can be used alone or in conjunction with other regularization approaches

# BIAS & VARIANCE

# Bias & Variance

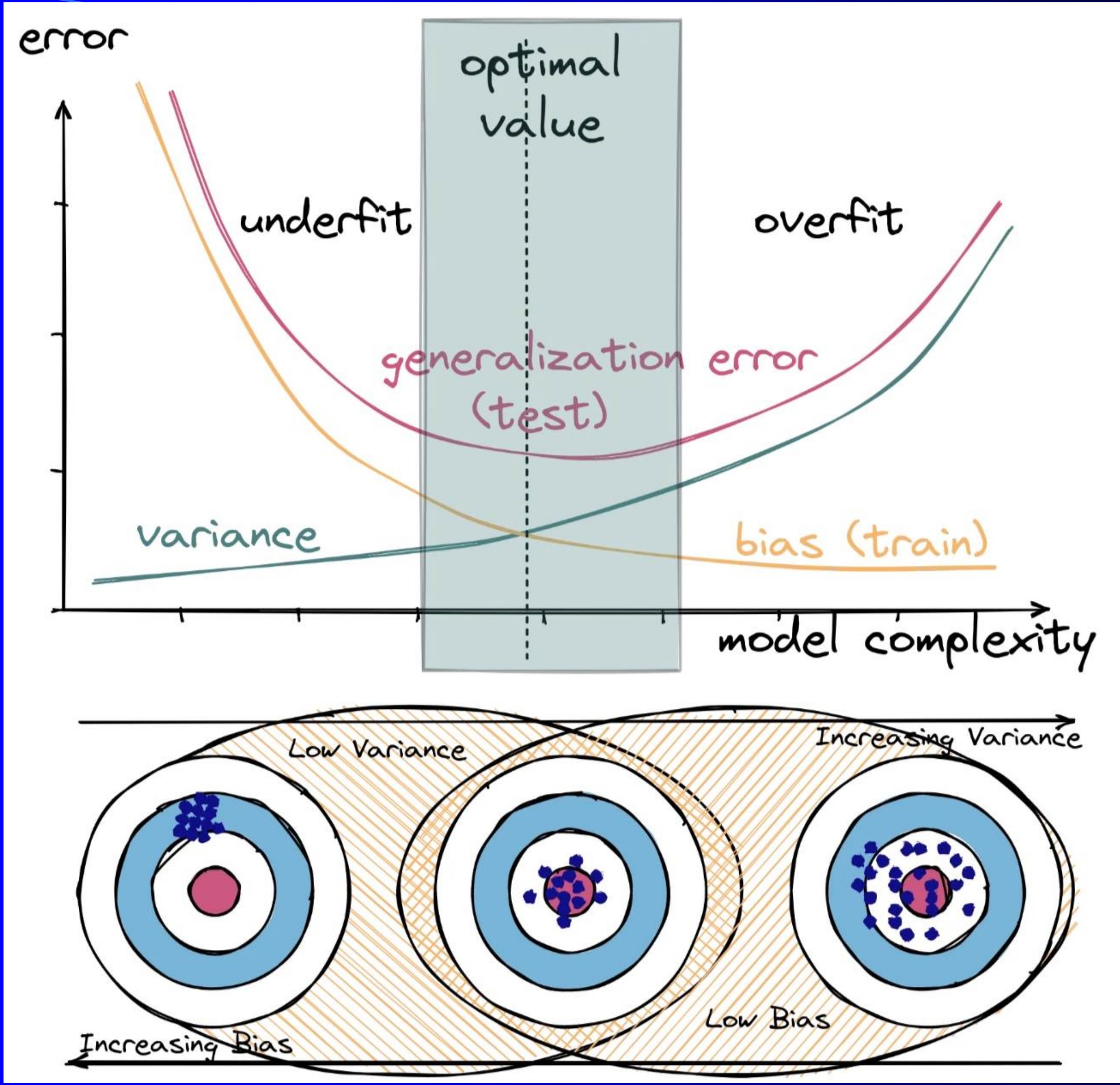
- Learning involves the ability to generalize from past experience to deal with new situations
- If we are only consistent with what we have seen, we can't generalize beyond our experience
  - All the cats I have seen are yellow.
  - What happens when I see a black cat?
- Bias is choosing one generalization over another
- Variance is how varied my choices are

# Bias vs. Variance

- If there is **no bias**, the outcome of the learner is highly dependent on the training data, and thus there is **much variance** among the models induced from different sets of observations
  - Learner memorizes (overfits)
- If there is a **strong bias**, the outcome of the learner is much less dependent on the training data, and thus there is **little variance** among induced models
  - Learner ignores observations
- Formalized as:
  - **Bias-variance trade-off**

# Bias-Variance Trade-off

- Weak/no bias
  - Observed instances are memorized
  - Learner **overfits**
- Strong/extreme bias
  - Observed instances are mostly ignored
  - Learner **underfits**
- Example: fitting arbitrary polynomial vs straight line to sine wave-like observations
- Bias-Variance Error Decomposition
  - All learning algorithms have a bias
    - Decision trees (simplicity), k-NN (similarity), etc.
  - All learning algorithms may be subject to variance in the data
  - Two sources of error
- How can we estimate the bias and variance of an algorithm?
  - Run the algorithm on different random variations of several datasets
  - Examine the errors made for each variation
    - If the algorithm tends to make the **same errors**, then it must have a **strong(er) bias** [one may need a more flexible algorithm]
    - If the algorithm tends to make **random errors**, then it must have a **strong(er) variance** [one may need a less flexible algorithm or more data]


error
optimal
value
underfit
overfit
generalization error
(test)
variance
bias (train)
model complexity
Low Variance
Increasing Variance
Low Bias
Increasing Bias

# Example Code

- BiasVariance.ipynb – on Learning Suite

# Your Projects

- Your goal in the group project is to get the highest possible generalization accuracy on a real-world application. You will come up with some task which you believe could be generalized with machine learning and as a group you will go through all the steps from beginning to end to get a good result.
- After you have come up with basic features and data, you will choose machine learning model(s), and format the data to fit the model(s).
- You will then try your model on novel data and report on your performance.

# Your Project Proposals

- Examples – Look at UC Irvine Data Set or Kaggle to get a feel of what data sets look like
- Stick with supervised classification problems for the most part for the project proposals
- Choose tasks which interest you
- Too hard vs Too Easy
  - Data should be able to be gathered in a relatively short time
  - And, want you to have to battle with the data/features a bit
- See description in Learning Suite
  - Remember your example instance!
  - As part of paragraph 2 ***give one fully specified example of a data set instance based on your proposed features, including a reasonable representation (continuous, nominal, etc.) and value for each feature.*** *Make sure you include the target output value as part of the instance.*

# Your Project Proposals

- After the Proposal Due Date, I will consider which proposals are most reasonable for the class.
- I will then distribute the proposals
- You will then rank your top-5
- I will try to match you up in groups for the projects
  - You may not get your first choice
  - If you proposed the project, and it is your top choice, …
- Examples online

# Feature Selection, Preparation, and Reduction

- Learning accuracy depends on the data!
  - *Is the data representative of future novel cases* - critical
  - Relevance
  - Amount
  - Quality
    - Noise
    - Missing Data
    - Skew
  - Proper Representation
  - How much of the data is labeled (output target) vs. unlabeled
  - Is the number of features/dimensions reasonable?
    - Reduction

# Gathering Data

- Consider the task – What kinds of features could help
- Data availability
  - Significant diversity in cost of gathering different features
  - More the better (in terms of number of instances, not necessarily in terms of number of dimensions/features)
    - The more features you have the more data you need
  - Data augmentation, Jitter – Increased data can help with overfit – handle with care!
- Labeled data is best
- If not labeled
  - Could set up studies/experts to obtain labeled data
  - Use unsupervised and semi-supervised techniques
    - Clustering
    - Active Learning, Bootstrapping, Oracle Learning, etc.

# Feature Selection - Examples

- Invariant Data
  - For character recognition: Size, Rotation, Translation Invariance
    - Especially important for visual tasks
  - Chess board features
    - Is vector of board state invariant?
- Character Recognition Class Assignment Example
  - Assume we want to draw a character with an electronic pen and have the system output which character it is
  - What features should we use and how would we train/test the system?

# When to Gather More Data

- When trying to improve performance, you may need
  - More Data
  - Better Input Features
  - Different Machine learning models or hyperparameters
  - Etc.
- One way to decide if you need more/better data
  - Compare your accuracy on training and test set
  - If bad training set accuracy then you probably need better data, features, noise handling, etc., or you might need a different learning model/hyperparameters
  - If test set accuracy is much worse than training set accuracy then gathering more data is usually a good direction, though overfit or learning model/hyperparameters could still be a major issue