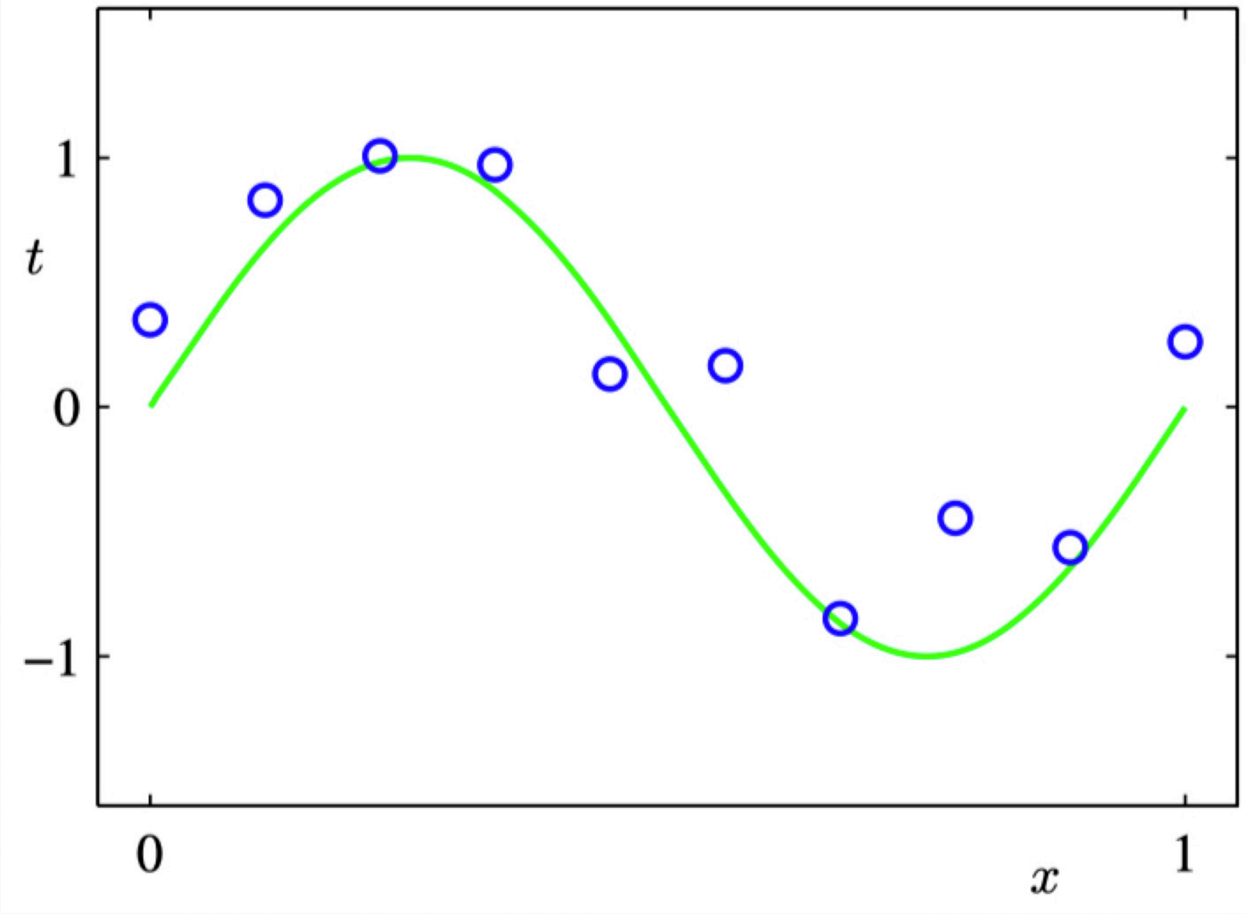
Dealing with Data: Overfitting and model selection

Day 2

Labatt Impact Lab Bootcamp

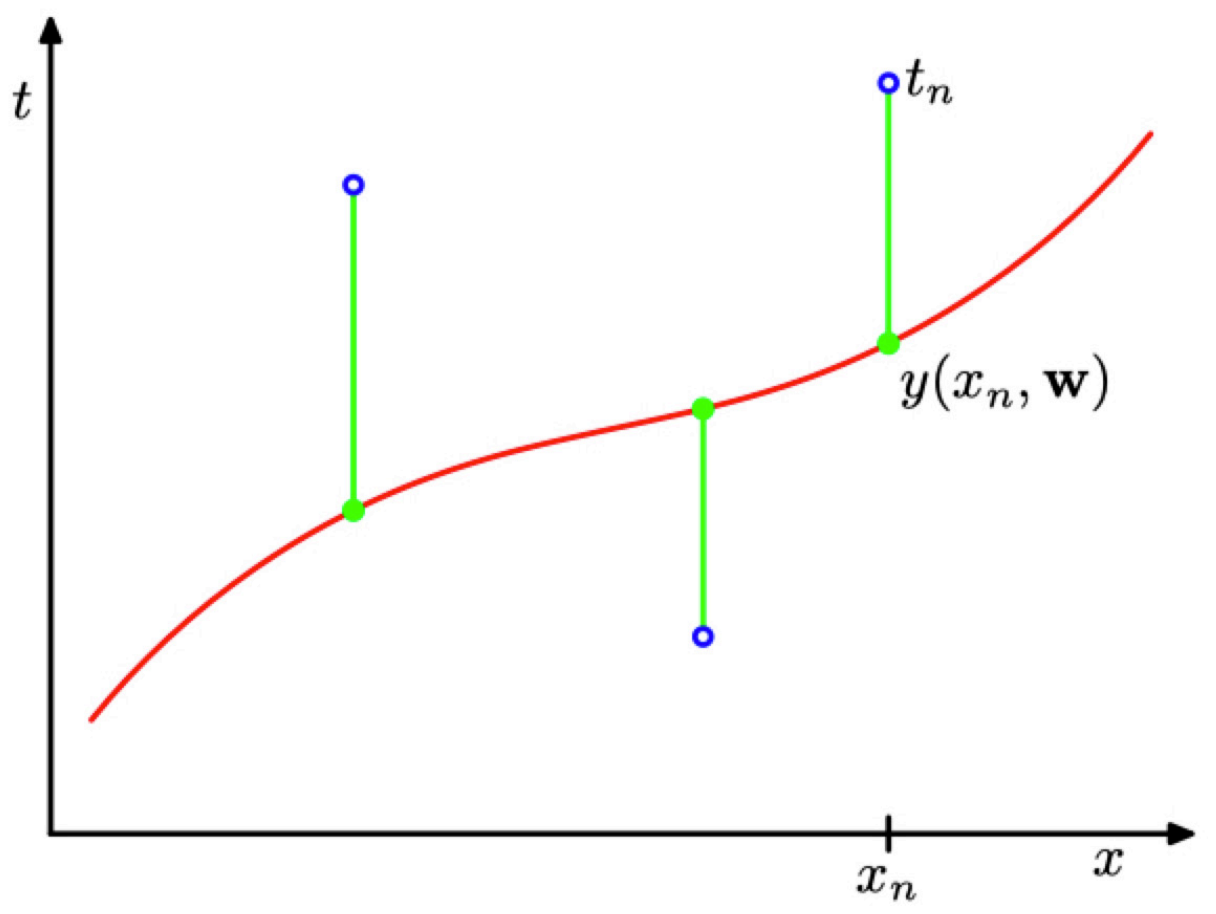
Regression

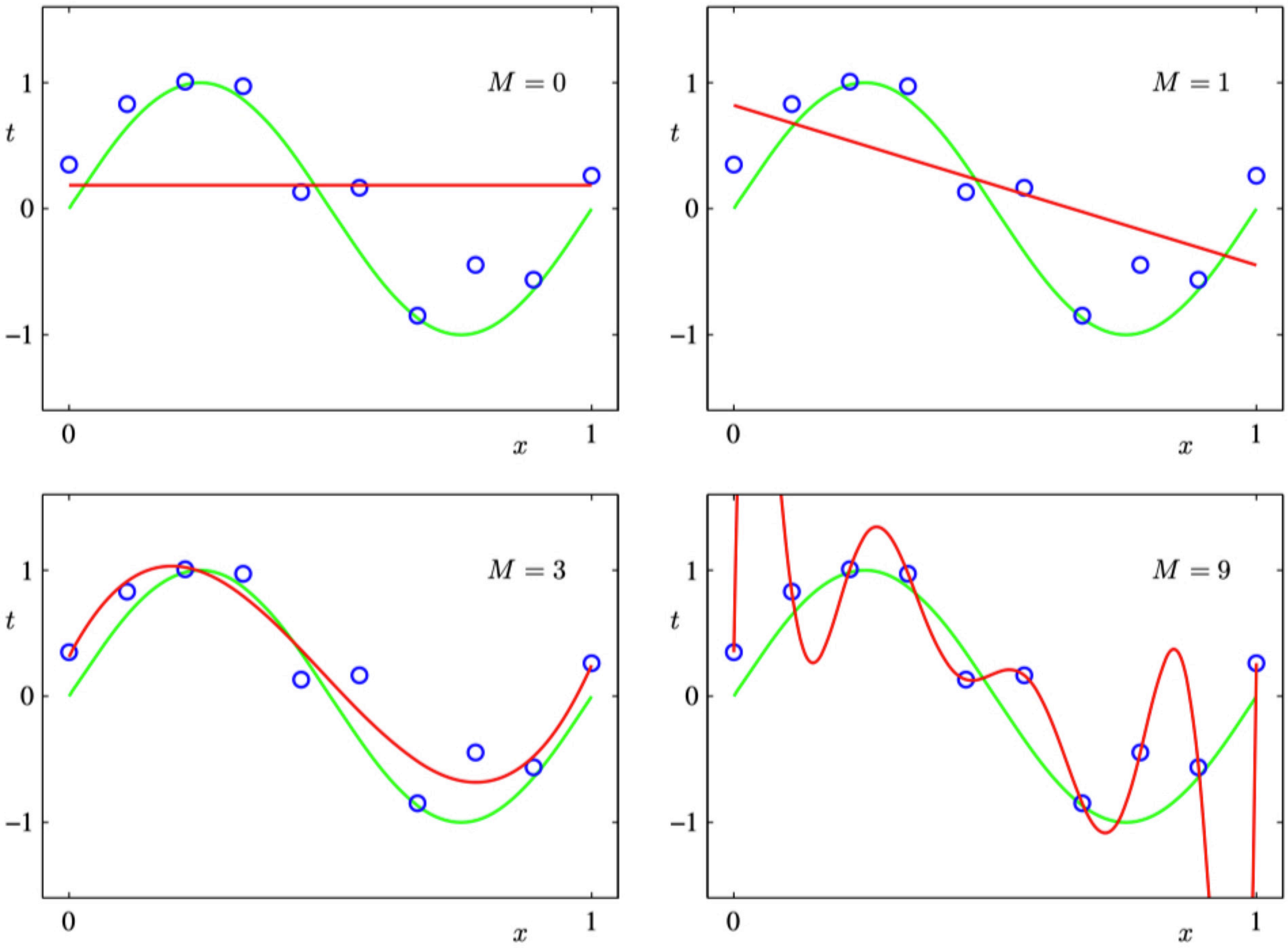
Data points **(x,t)** generated by adding noise to $sin(2\pi x)$



Errors in Regression

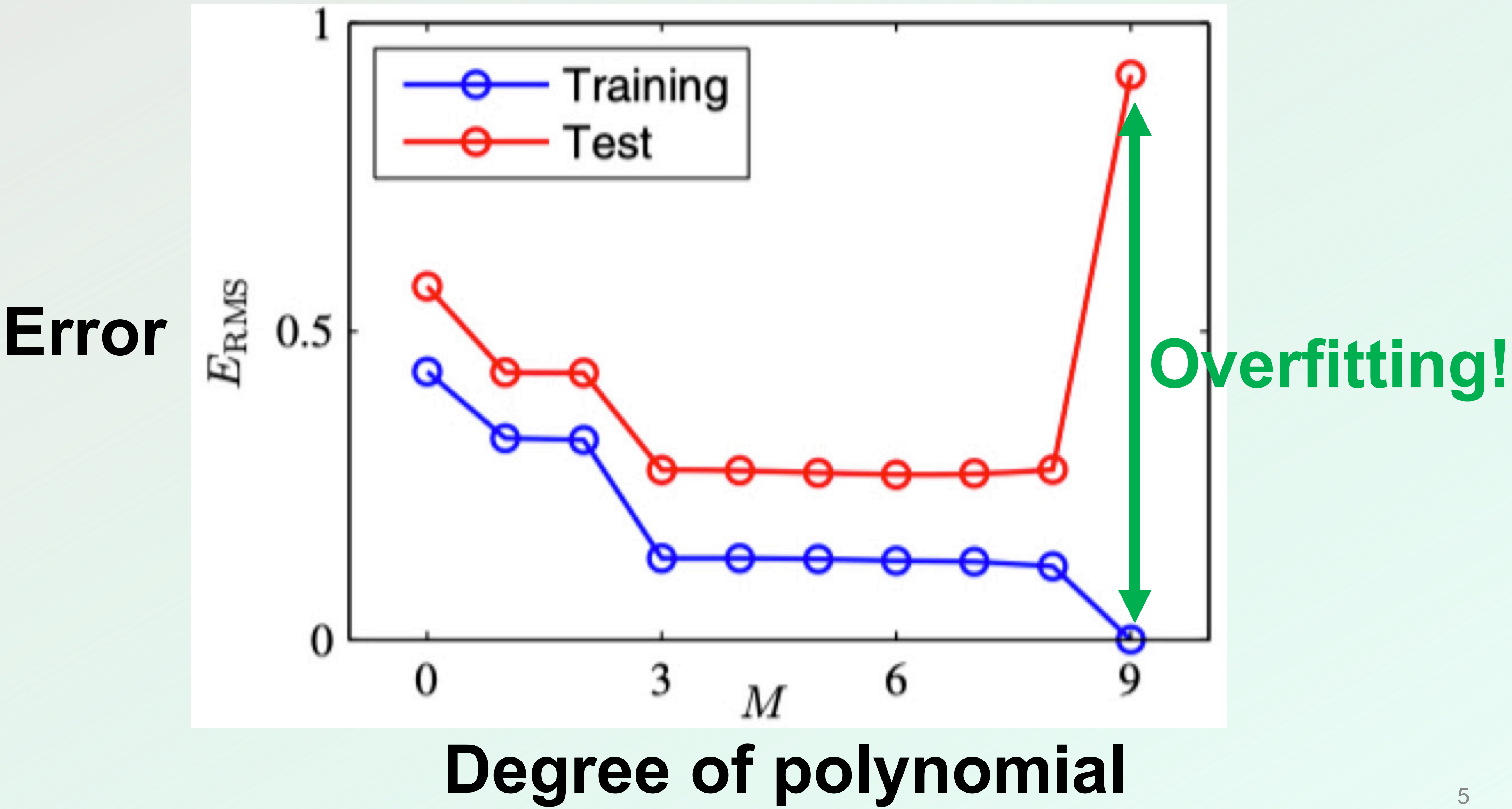
$$E(w) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, w) - t_n\}^2$$



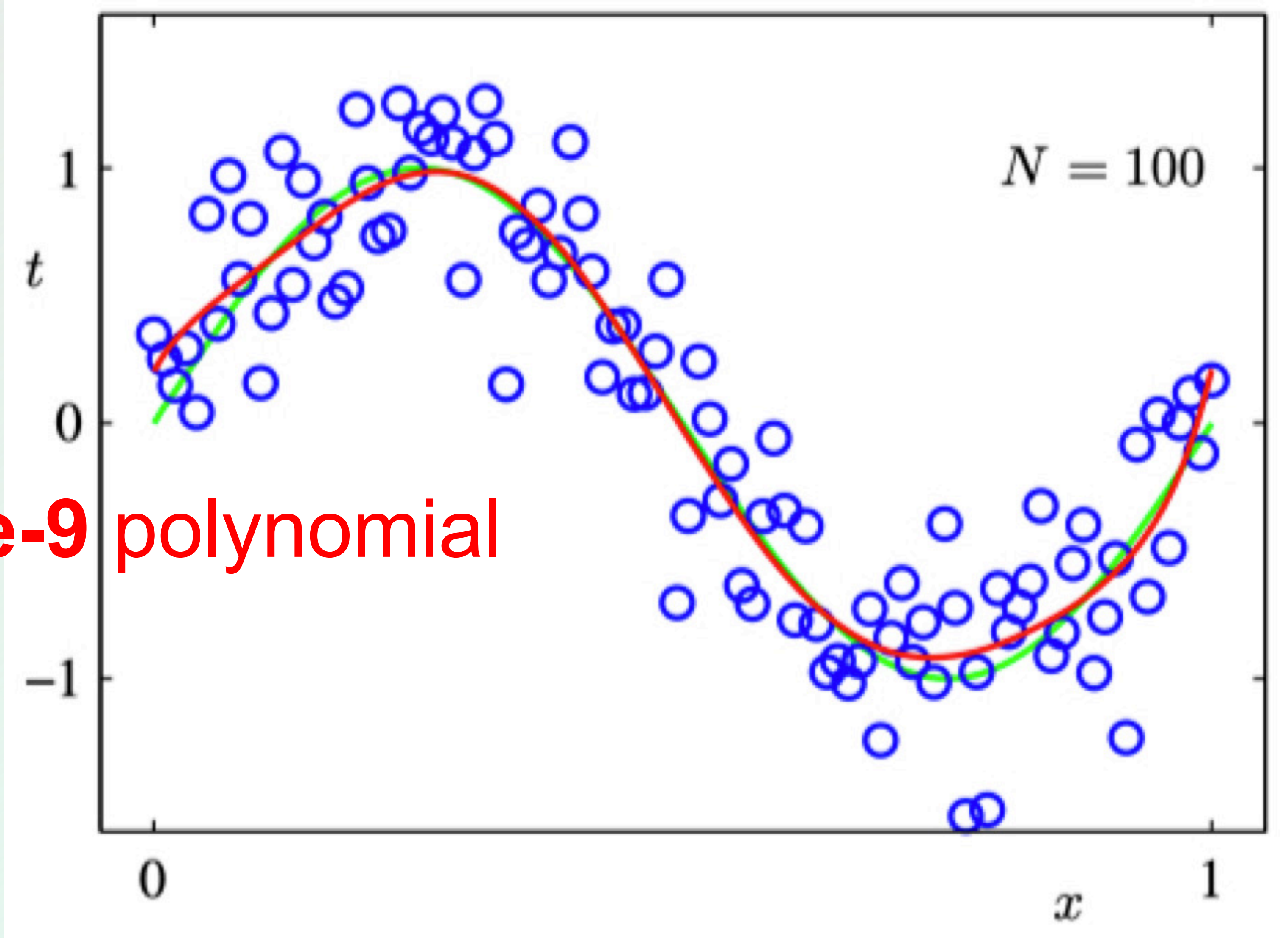


$M = 0$
$M = 1$
$M = 3$
$M = 9$
t
x

Error as a function of degree of polynomial



Dealing with Overfitting: (1) more data!



Degree-9 polynomial

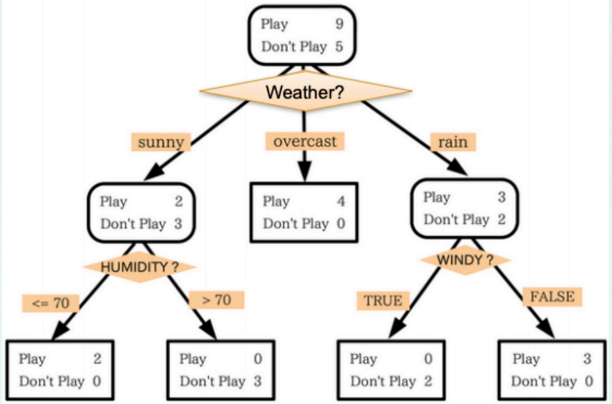
Dealing with Overfitting: (2) regularization

minimize:

$$E(w) = \underbrace{\frac{1}{2}\sum_{n=1}^{N}\{y(x_n, w) - t_n\}^2}_{\text{Model's squared error}} + \underbrace{\frac{\lambda}{2}||\mathrm{w}||^2}_{\text{Penalty on parameters}}$$

Regularization in Classification

- Decision Trees
  - Limit the depth of the tree
  - Prune subtrees after training

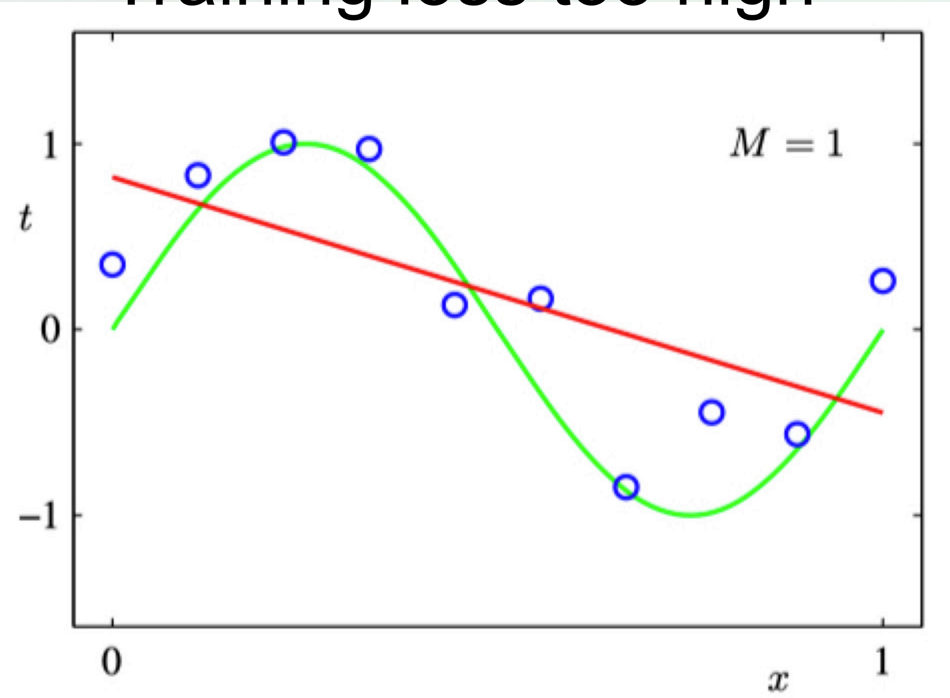


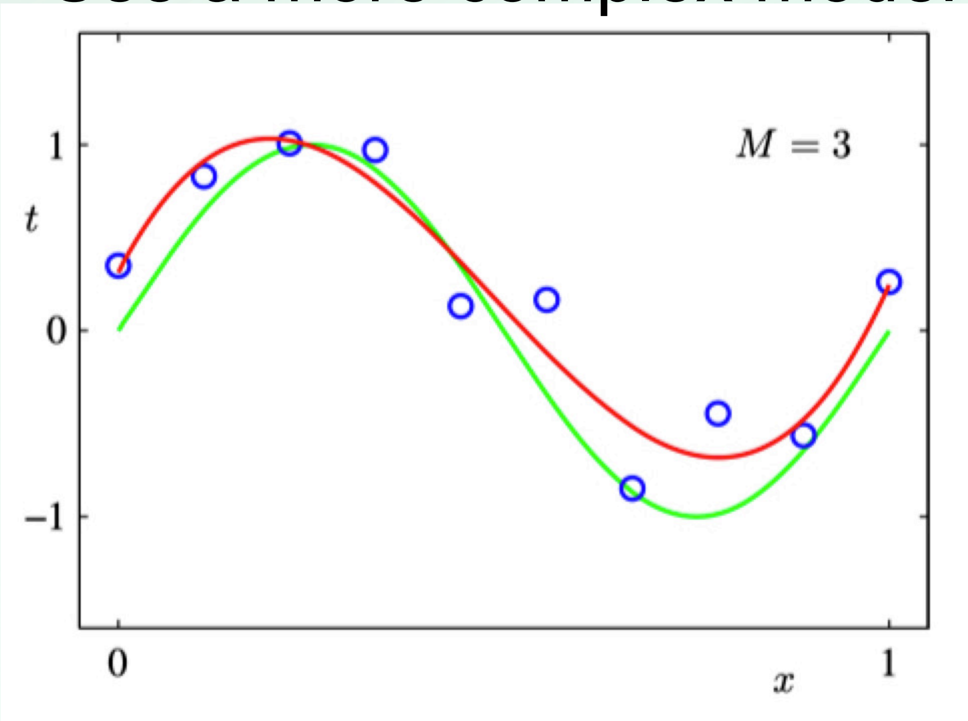
- Support Vector Machines
  - Built-in, tunable regularization term
- Logistic Regression
  - Can add a tunable regularization term to MLE objective

Penalize large parameter values!

Underfitting: model is too simplistic!

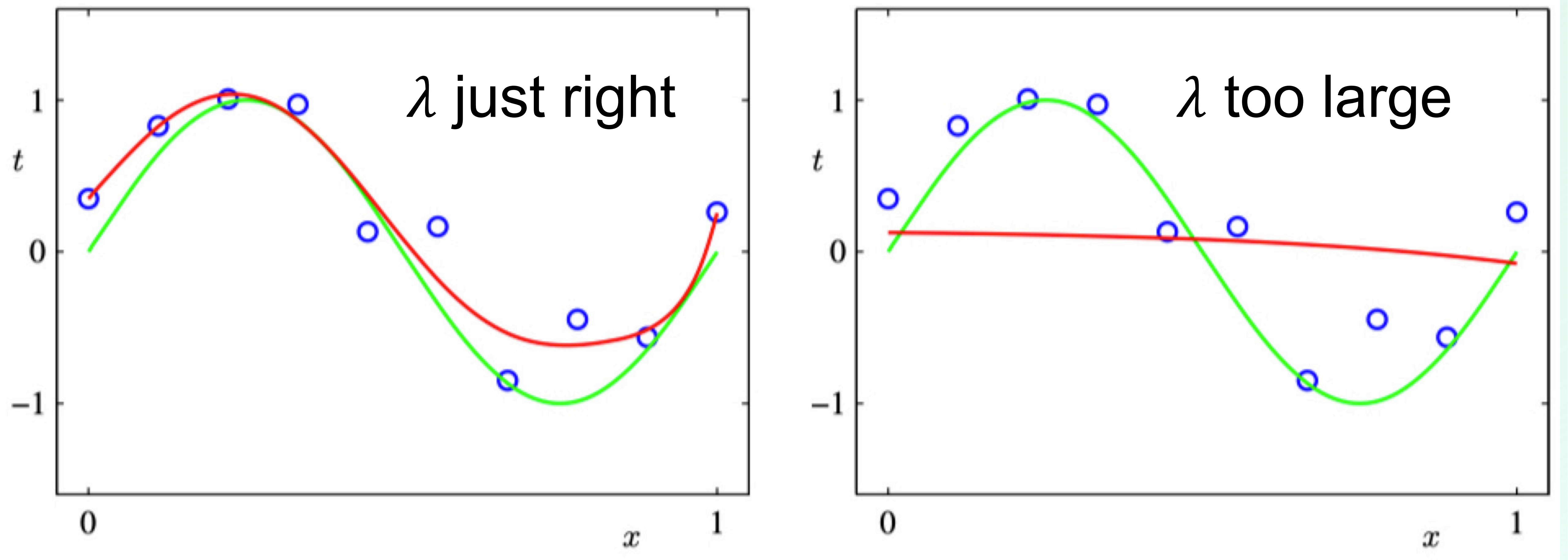
Training loss too high $\longrightarrow$ Use a more complex model





Dealing with Overfitting: (2) regularization

minimize: $E(w) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, w) - t_n\}^2 + \frac{\lambda}{2}||\mathrm{w}||^2$



Parameters vs Hyper-parameters

Example **hyper-parameters**

- Decision tree: maximum depth of the tree
- Polynomial regression: regularization coefficient

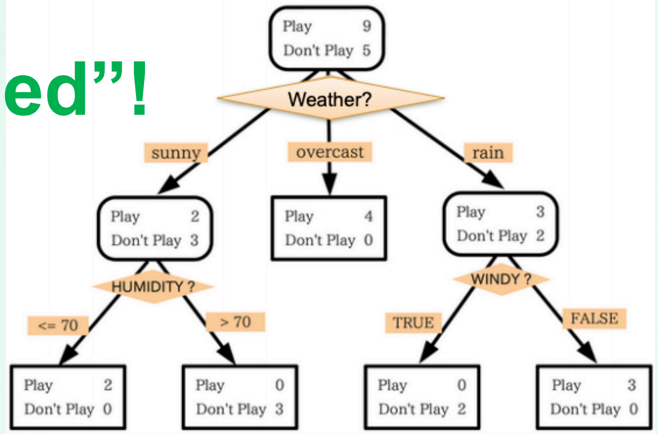
$$E(w) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, w) - t_n\}^2 + \frac{\lambda}{2}||\mathrm{w}||^2$$

- Hyper-parameters are **determined through trial-and-error / cross validation**

Example **parameters: directly optimized, "learned"!**

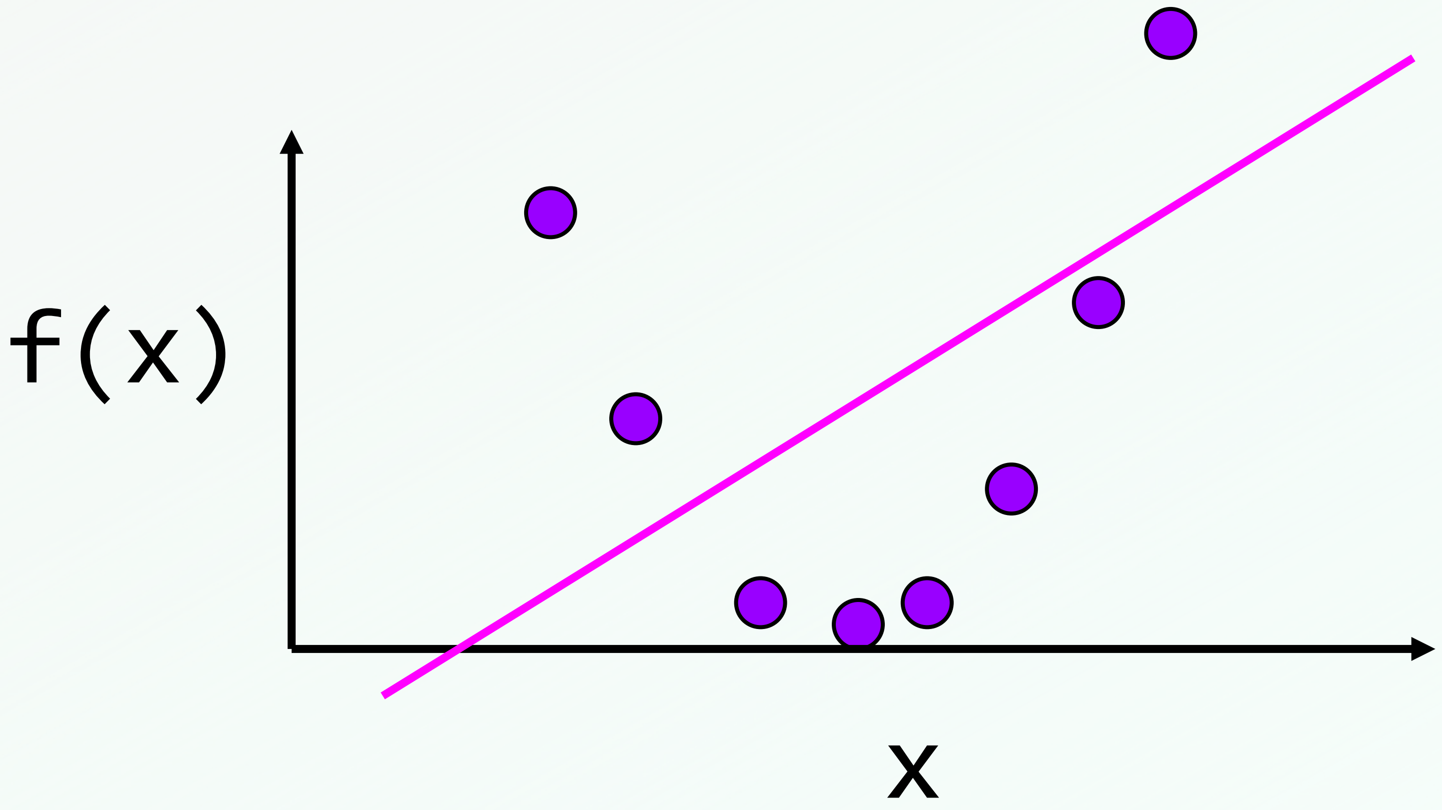
- Decision tree: attributes you split on
- Logistic regression: weights $\boldsymbol{\beta}$

$$p(x;\boldsymbol{\beta}) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\cdots+\beta_d x_d)}}$$



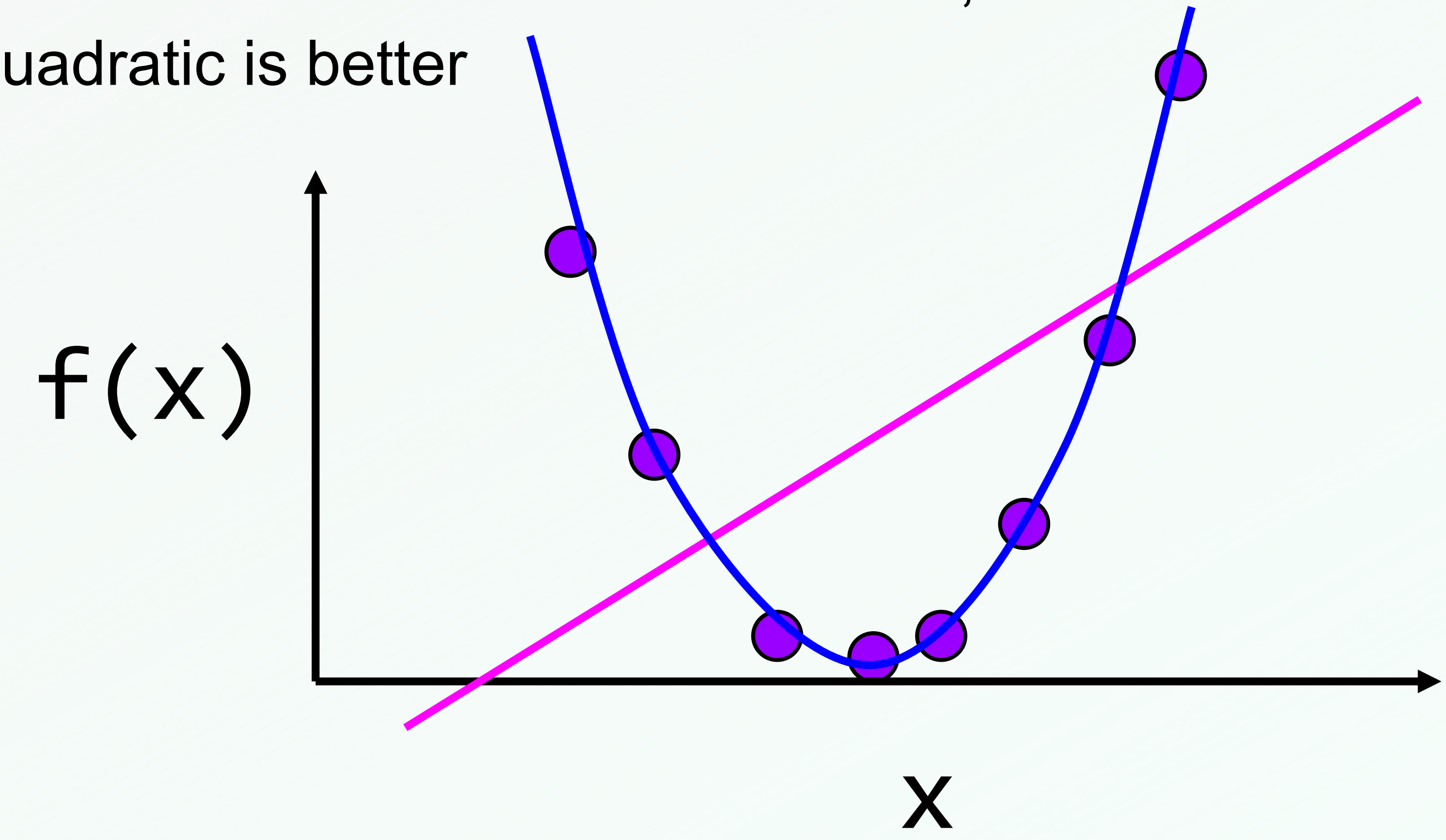
Bias / Variance tradeoff

- Bias
  - Error from assumptions model makes about data
  - Linear model assumes data is linear, bad for data that isn’t



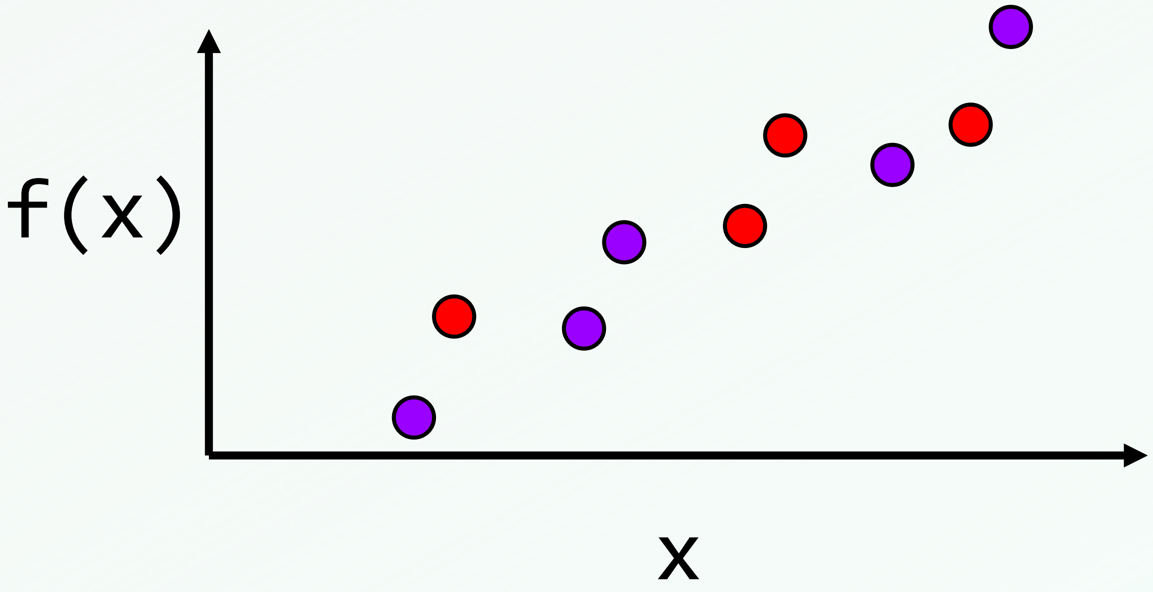
Bias / Variance tradeoff

- Bias
  - Error from assumptions model makes about data
  - Linear model assumes data is linear, bad for data that isn’t
  - Quadratic is better



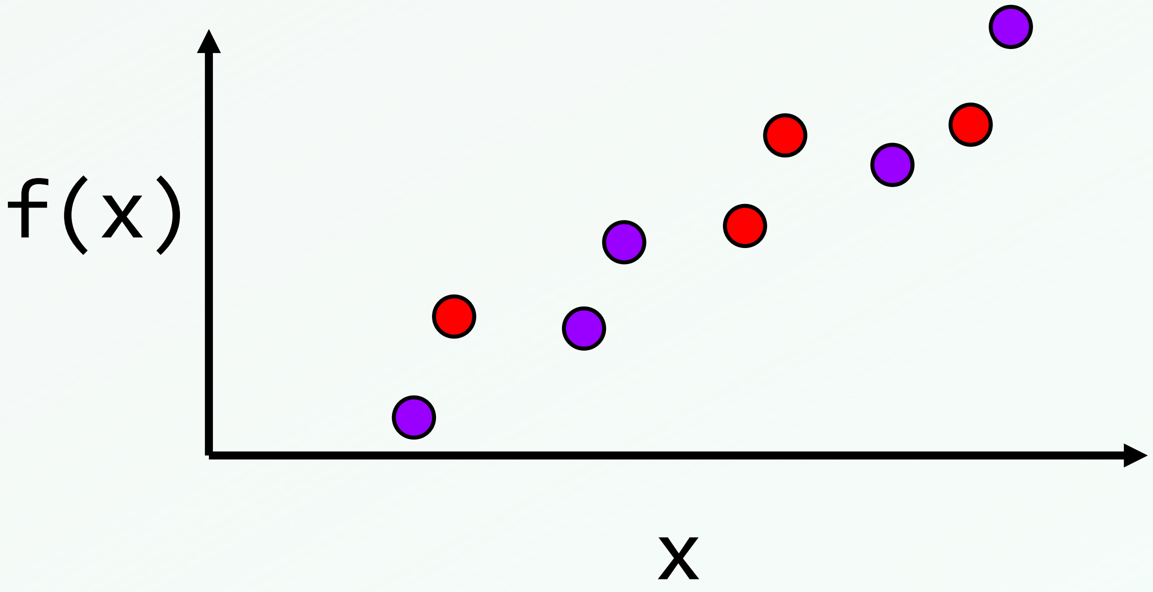
Bias / Variance tradeoff

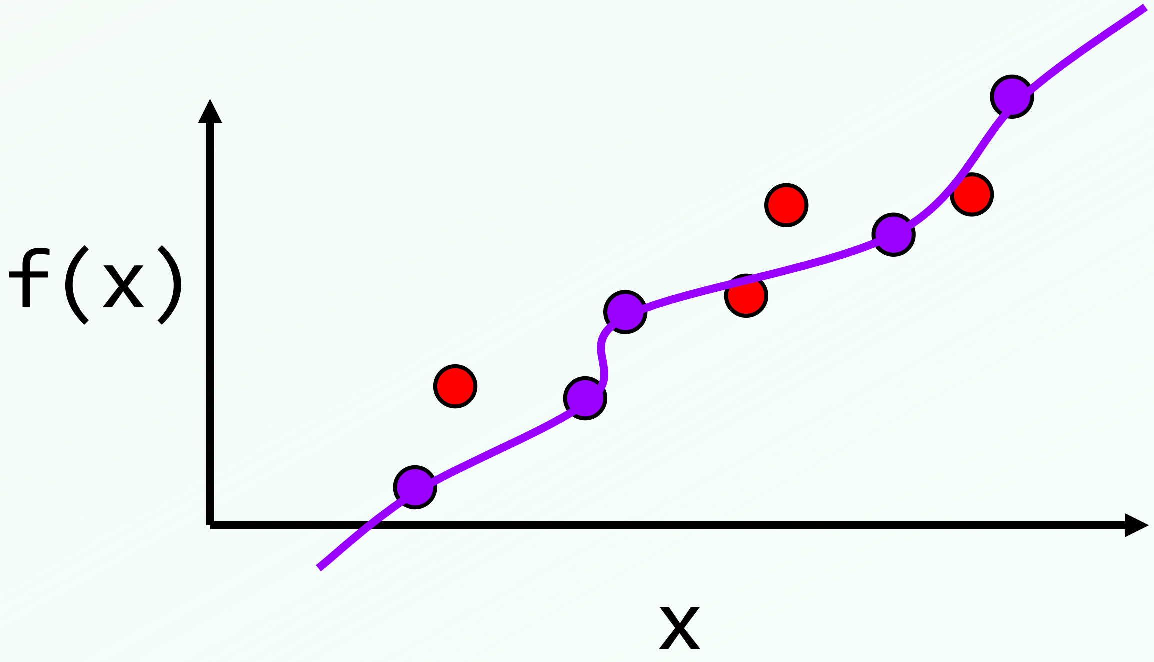
- Variance
  - Algorithm’s sensitivity to noise
  - More complex algorithms are more sensitive!



Bias / Variance tradeoff

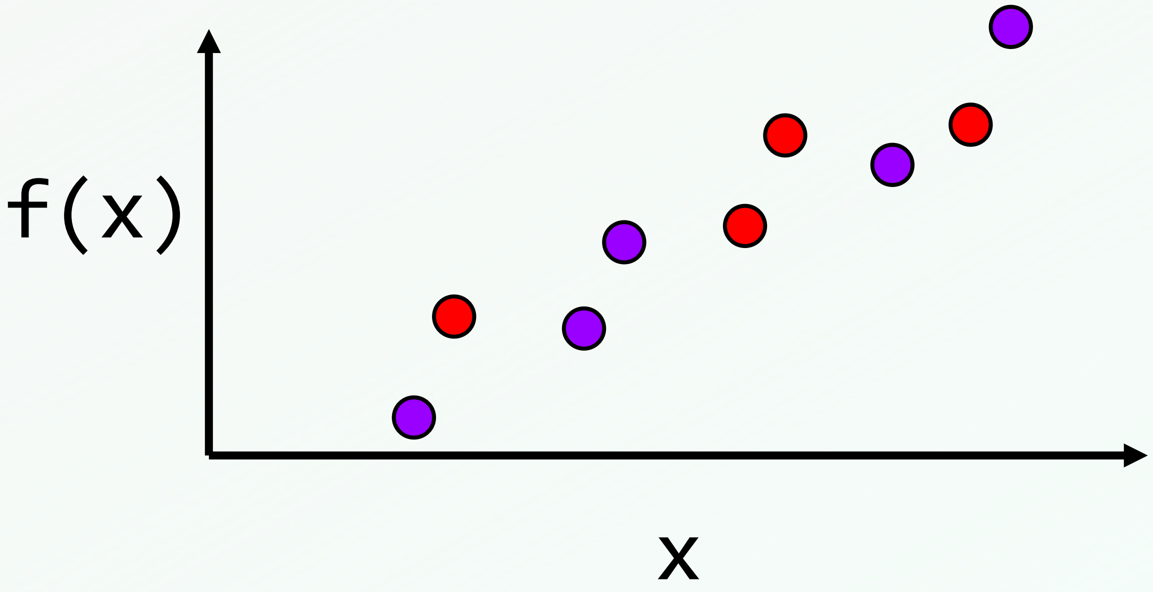
- Variance
  - Algorithm’s sensitivity to noise
  - More complex algorithms are more sensitive!

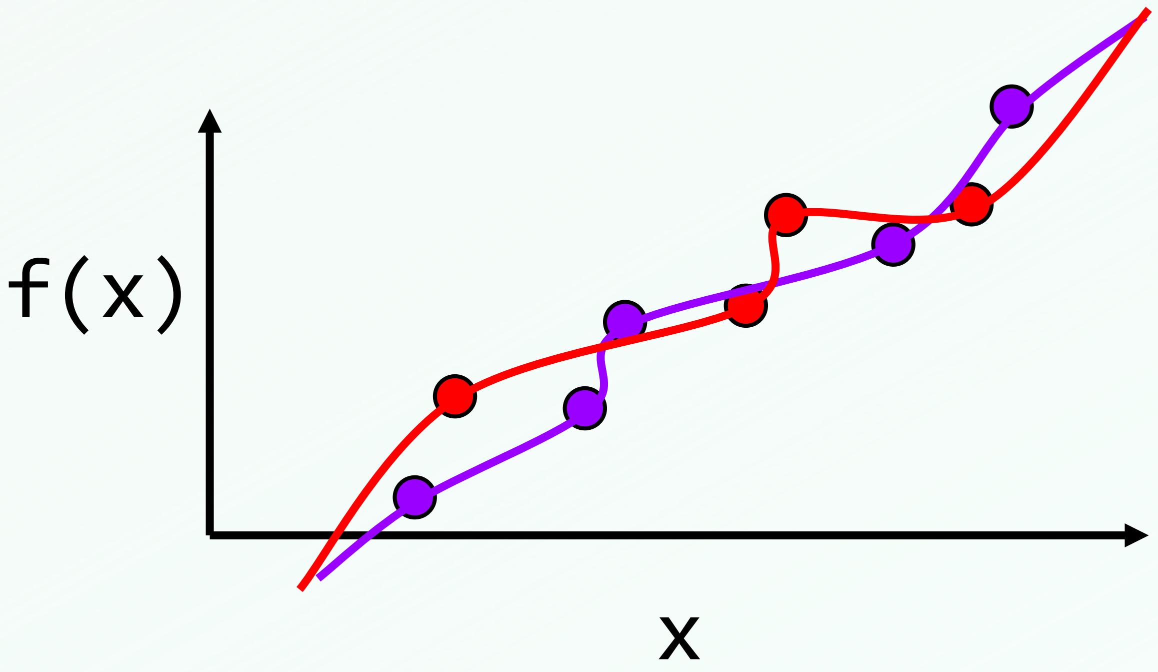




Bias / Variance tradeoff

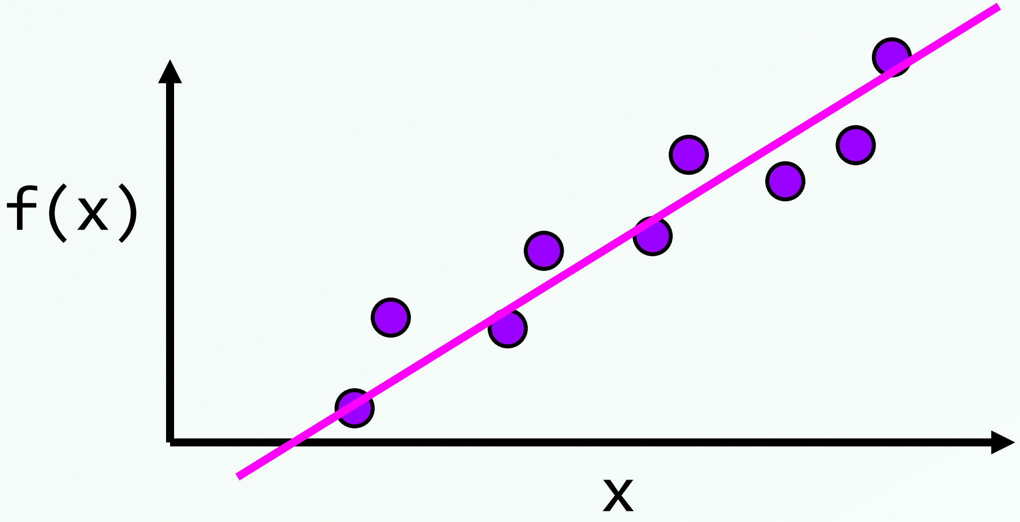
- Variance
  - Algorithm’s sensitivity to noise
  - More complex algorithms are more sensitive!
  - High variance hurts generalization, *overfitting*





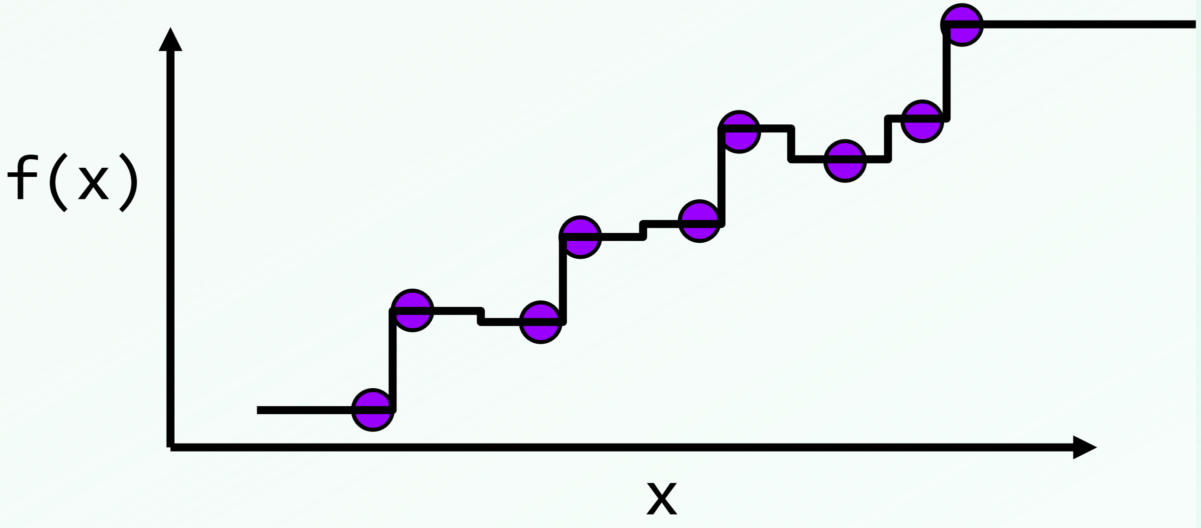
Linear regression

- $f^*(x) = ax + b$
- **High bias**: linear assumption
- **Low variance**
- Benefits:
  - Closed form solution
  - Fast to compute for new data
- Weaknesses:
  - Not very powerful, assumes linear
  - Underfit more interesting data

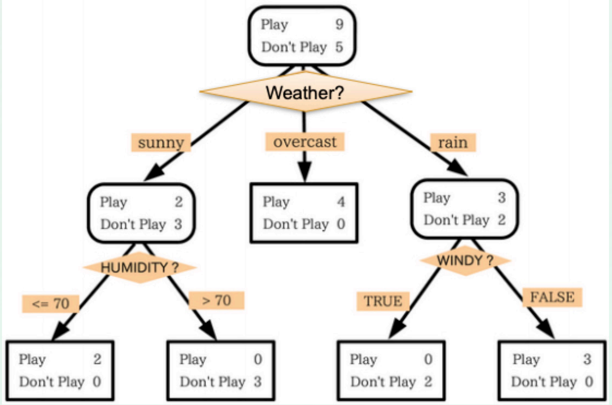


Nearest neighbor regression

- f*(x) = f(x') for nearest x' in training set
- **Low bias**: no assumptions about data
- **High variance**: very sensitive to training set
- Benefits:
  - Super easy to implement
  - Easy to understand
  - Arbitrarily powerful, esp with lots of data
- Weaknesses:
  - Hard to scale
  - Prone to **overfitting** to noise



Model Combination



Any single model may have **high variance** in its predictions

- Decision tree: predictions can be brittle, small perturbation in data can lead to misclassification

Creating more stable models:

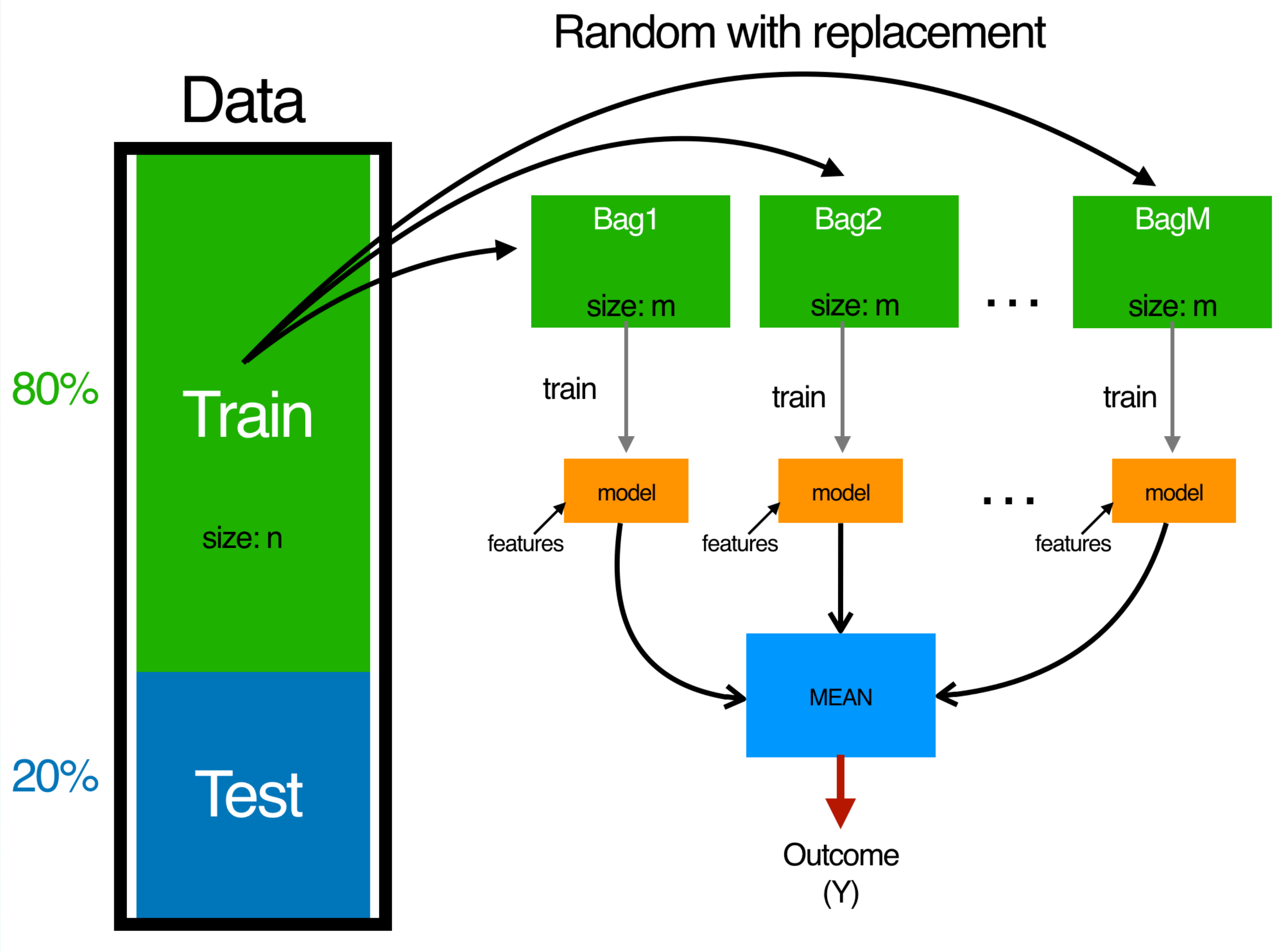
- Come up with a completely new model
- **Combine (unstable) models intelligently!**

Bootstrap Sampling

Data: $S = \{(x_i, y_i)\}_{i = 1,\ldots,n}$

Bootstrap Sample: subsample of S, drawn **with replacement**

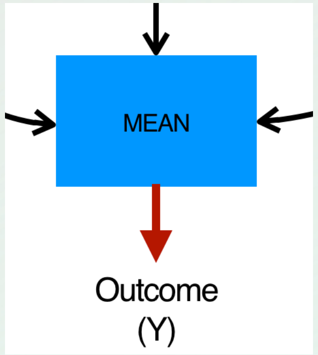
Bootstrap **Aggregating (Bagging)**



Bagging

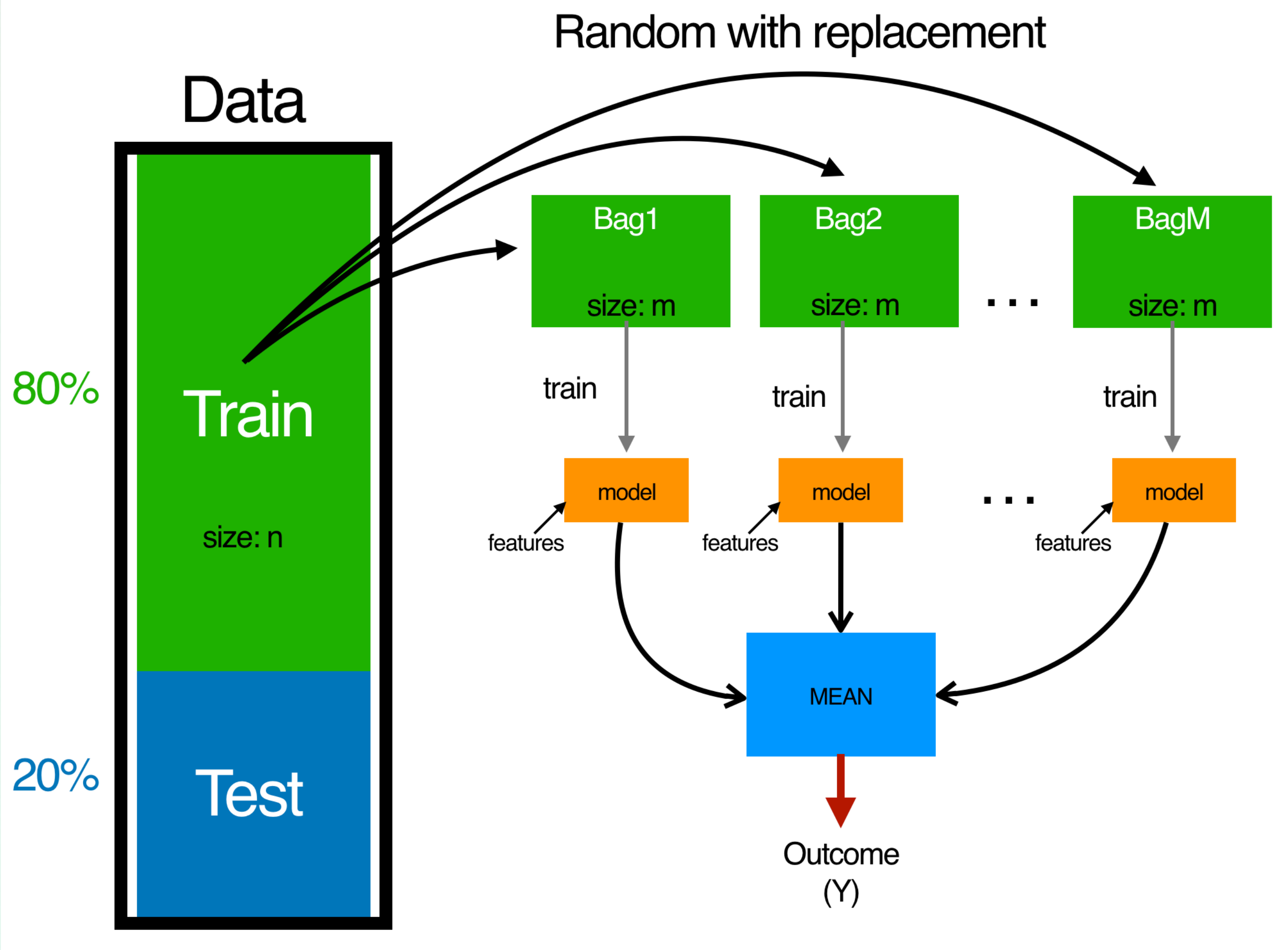
Aggregation:

- **Majority vote!**

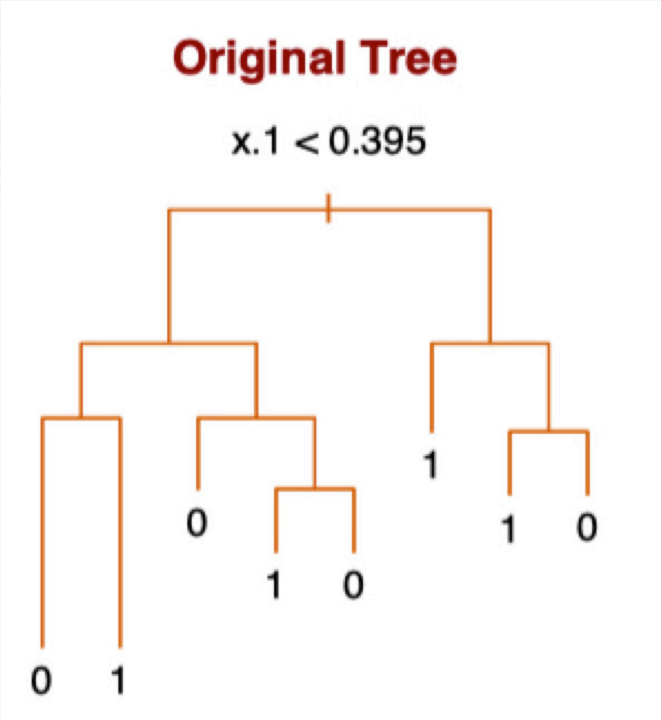


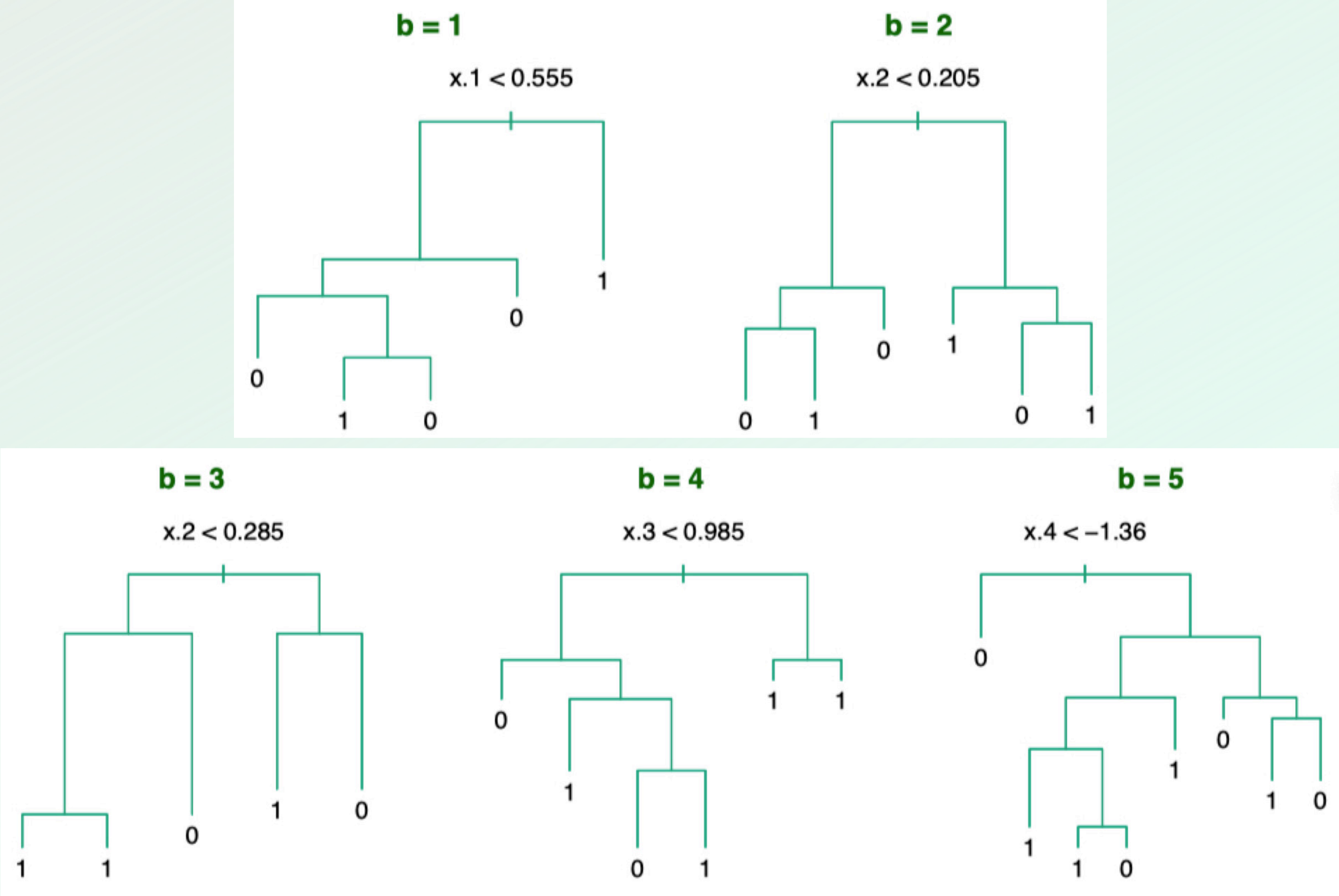
Benefits:

- Even if single model overfits, on average they will not

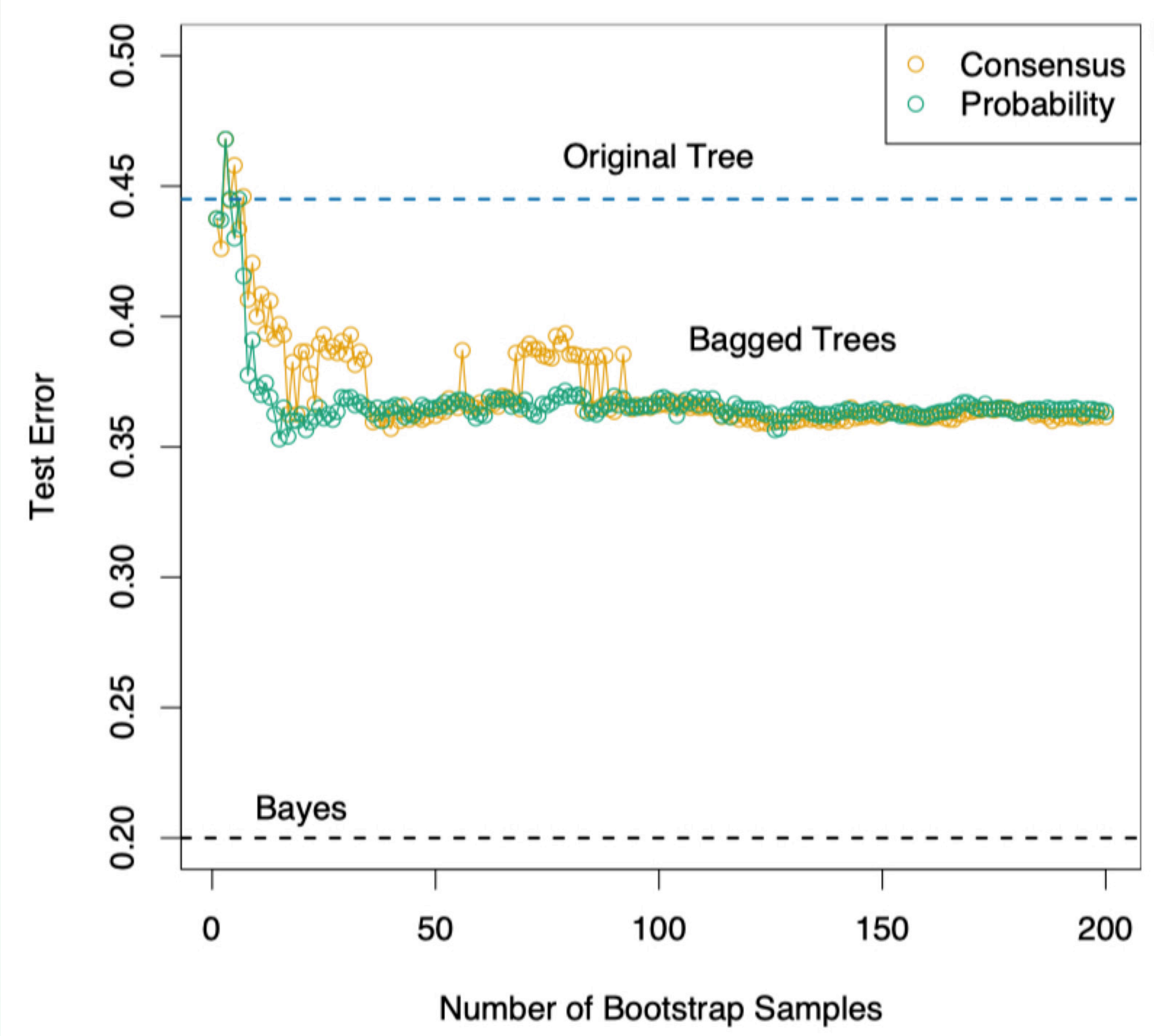


Bagging with Decision Trees





Predicting with bagged decision trees

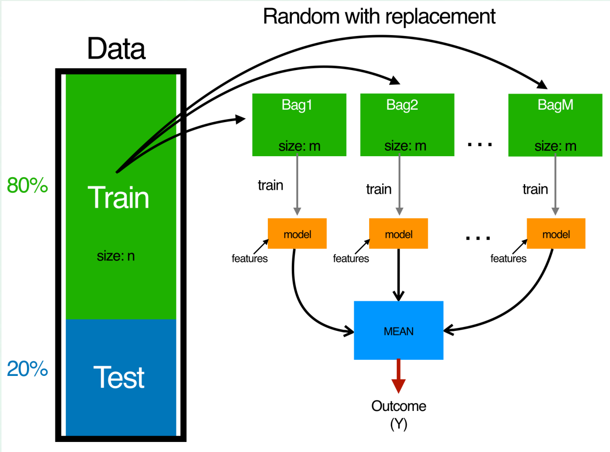


Consensus: proportion of trees that predict a given class

Probability:
- For b-th tree, look at the leaf corresponding to test sample
- What proportion of the training data had the same label as test sample?
- Average those proportions out

Prediction: in both cases, predict class with highest score

When should use Bagging?



High-variance classifiers, e.g., decision trees.

Bagging reduces variance by providing an alternative approach to **regularization**.

Even if each of the learned classifiers are individually overfit, they are likely to **overfit to different things**.

Through voting, we can overcome a significant portion of this overfitting.

In practice, bagging tends to **reduce variance** and **increase bias**.

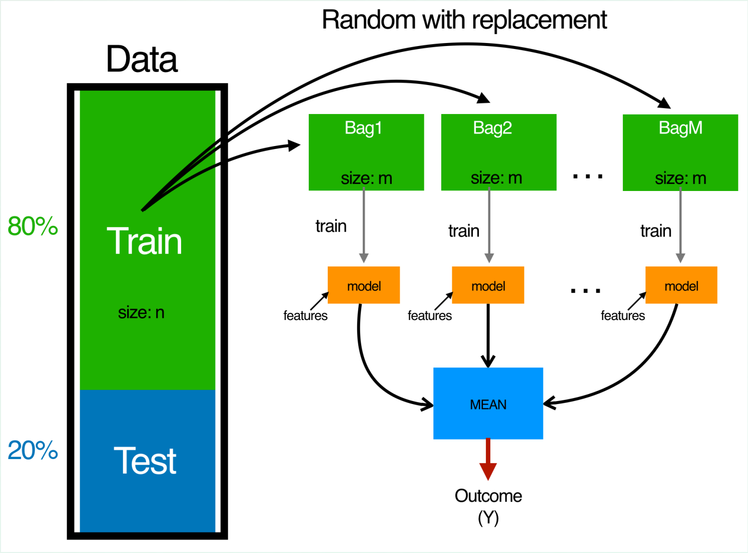
Final words on Bagging

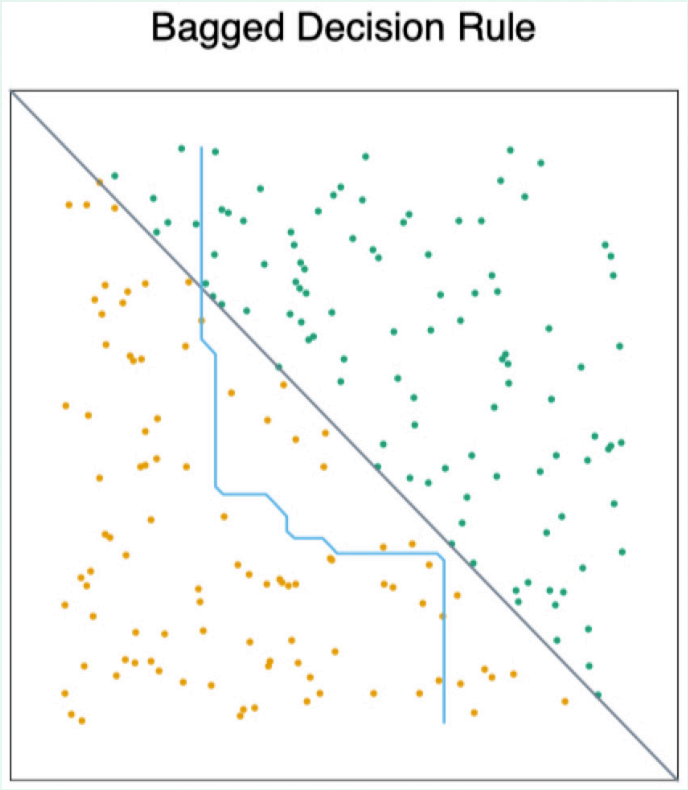
Advantages

- Reduces model variance
- Trivial to implement and use

Caveats

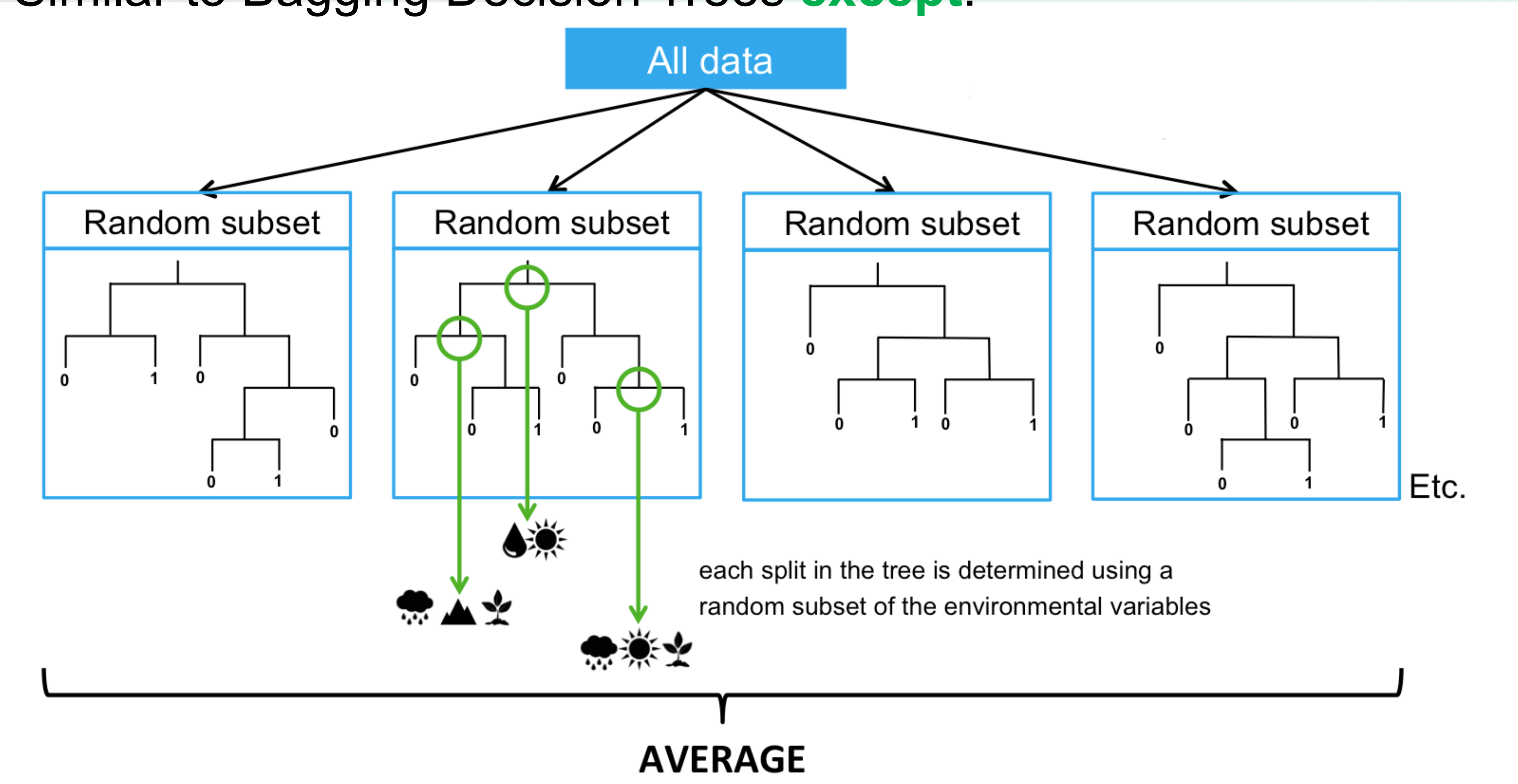
- **Destroys any interpretability** in the base model
- May not capture simple patterns



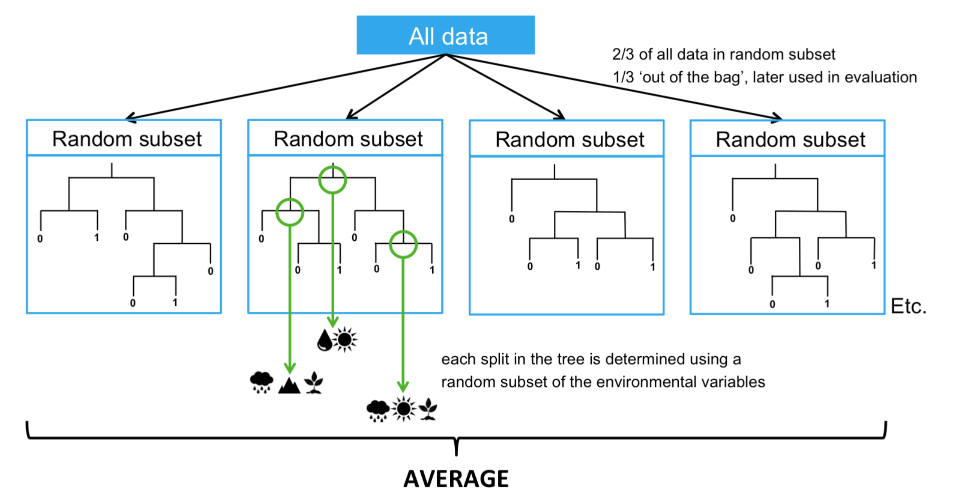


Random Forest

Similar to Bagging Decision Trees **except**:



Random Forest



- One of the most **popular** models!
- Usually **more effective** than Bagging Decision Trees
- Typically, use **100s of trees** (hyper-parameter to be tuned!)
- For data with d features, num. random features used for splitting:
  - Classification: $\sqrt{\boldsymbol{d}}$
  - Regression: $\boldsymbol{d/3}$

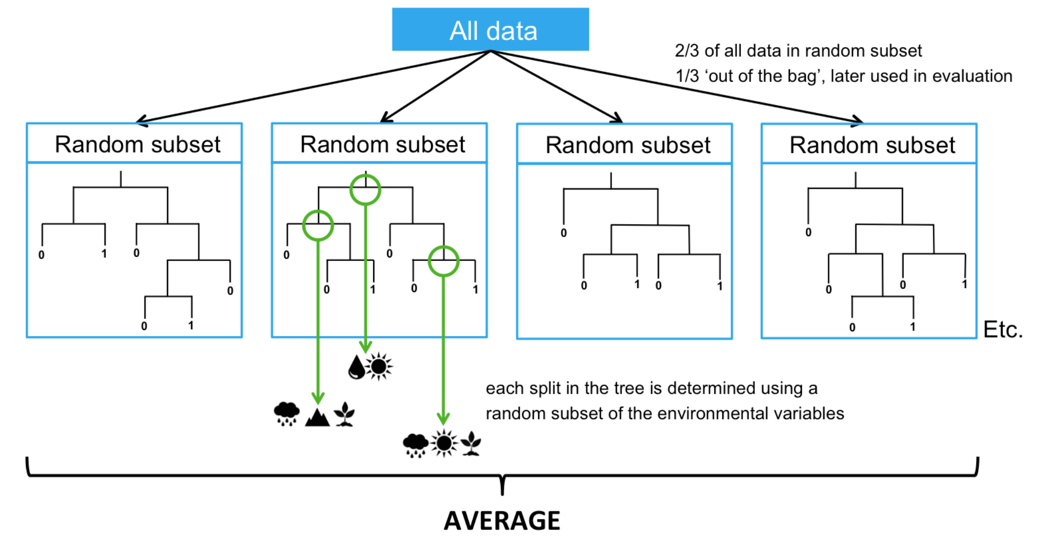
Final words on RF

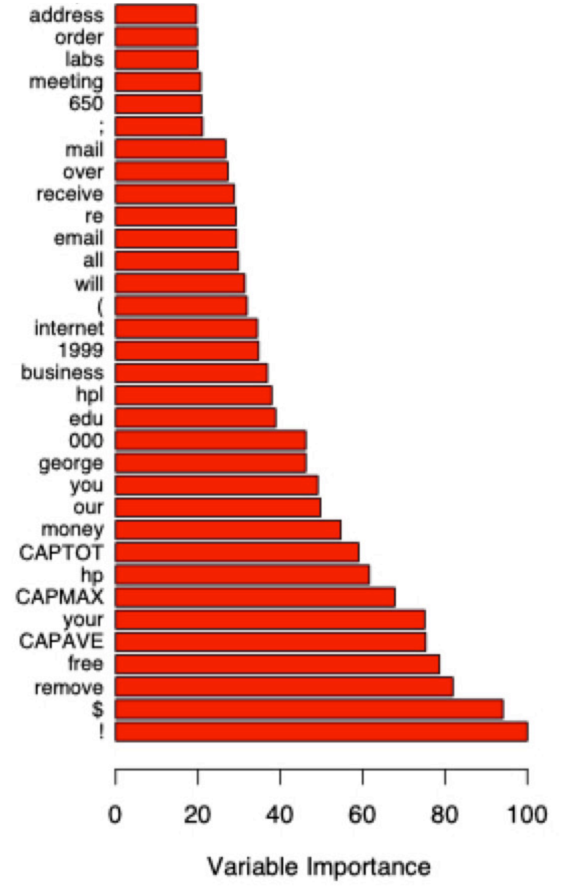
Advantages

- Feed it **data as is** without preprocessing
- **Fast training** because of feature sampling!
- Easy **parallel** training and prediction

Caveats

- **Not suitable for sparse data (why?)**





Recap

- Ensemble methods are a **must-try**

- Bagging: average models trained on bootstrap samples
- Good bagging: **Random Forests**