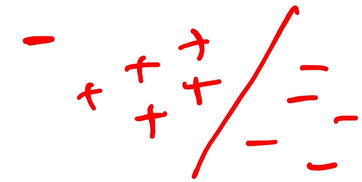
Ensemble Methods

Boosting [Schapire'89]



- **Idea:** given a weak learner, run it multiple times on (reweighted) training data, then let learned classifiers vote

- On each iteration *t*:
  - Learn a weak hypothesis – h_t
  - A weight for this hypothesis – α_t
  - weight $D_t(i)$ for each training example i, based on how incorrectly it was classified

$\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$

weighted training error of h_t

$D_{t+1}(i) \leftarrow \frac{D_t(i)}{Z_t} \begin{cases} e^{\alpha_t} \\ e^{-\alpha_t} \end{cases}$

- Final classifier: $\boxed{H(X) = \text{sign}(\sum \alpha_t h_t(X))}$

- **Practically useful**
- **Theoretically interesting**

Boosting and Logistic Regression

$P(y=1|x) = \frac{1}{1+\exp^{-f(x)}}$ ✓

Logistic regression equivalent to minimizing log loss

$$\sum_{i=1}^{m} \ln(1 + \exp(-y_i f(x_i))) \qquad f(x) = w_0 + \sum_j w_j x_j$$

Boosting minimizes similar loss function!!

$$\frac{1}{m}\sum_{i=1}^{m} \exp(-y_i f(x_i)) \qquad f(x) = \sum_t \alpha_t h_t(x)$$

Weighted average of weak learners

$y_i = 1$

Both smooth and convex approximations of 0/1 loss!

Boosting and Logistic Regression

Logistic regression:

- Minimize log loss

$$\sum_{i=1}^{m} \ln(1 + \exp(-y_i f(x_i)))$$

- Define

$$f(x) = \sum_j w_j x_j$$

where x_j predefined features

(linear classifier)

- Jointly optimize over all weights w_0, w_1, w_2…

Boosting:

- Minimize exp loss

$$\sum_{i=1}^{m} \exp(-y_i f(x_i))$$

- Define

$$f(x) = \sum_t \alpha_t h_t(x)$$

where $h_t(x)$ defined dynamically to fit data

(not a linear classifier)

- Weights α_t learned per iteration t incrementally

Hard & Soft Decision

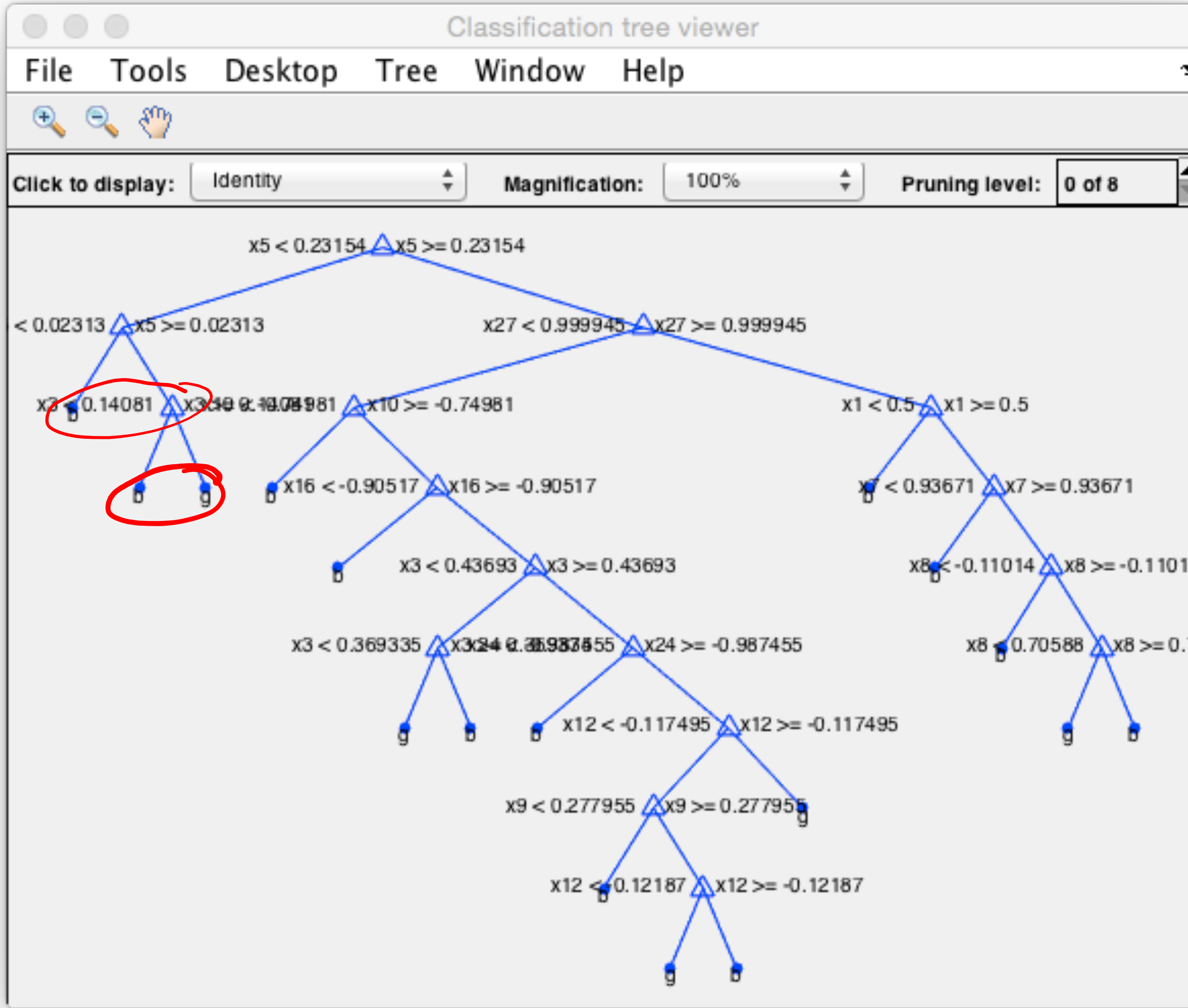
Weighted average of weak learners $$f(x) = \sum_t \alpha_t h_t(x)$$

Hard Decision/Predicted label: $$H(x) = sign(f(x))$$

Soft Decision:
(based on analogy with
logistic regression)

$$P(Y = 1|X) = \frac{1}{1 + \exp(-f(x))}$$

Matlab example – decision tree



load ionosphere

% UCI dataset

% 34 features, 351 samples

% binary classification

rng(100)

%Default MinLeafSize = 1

tc = fitctree(X,Y);

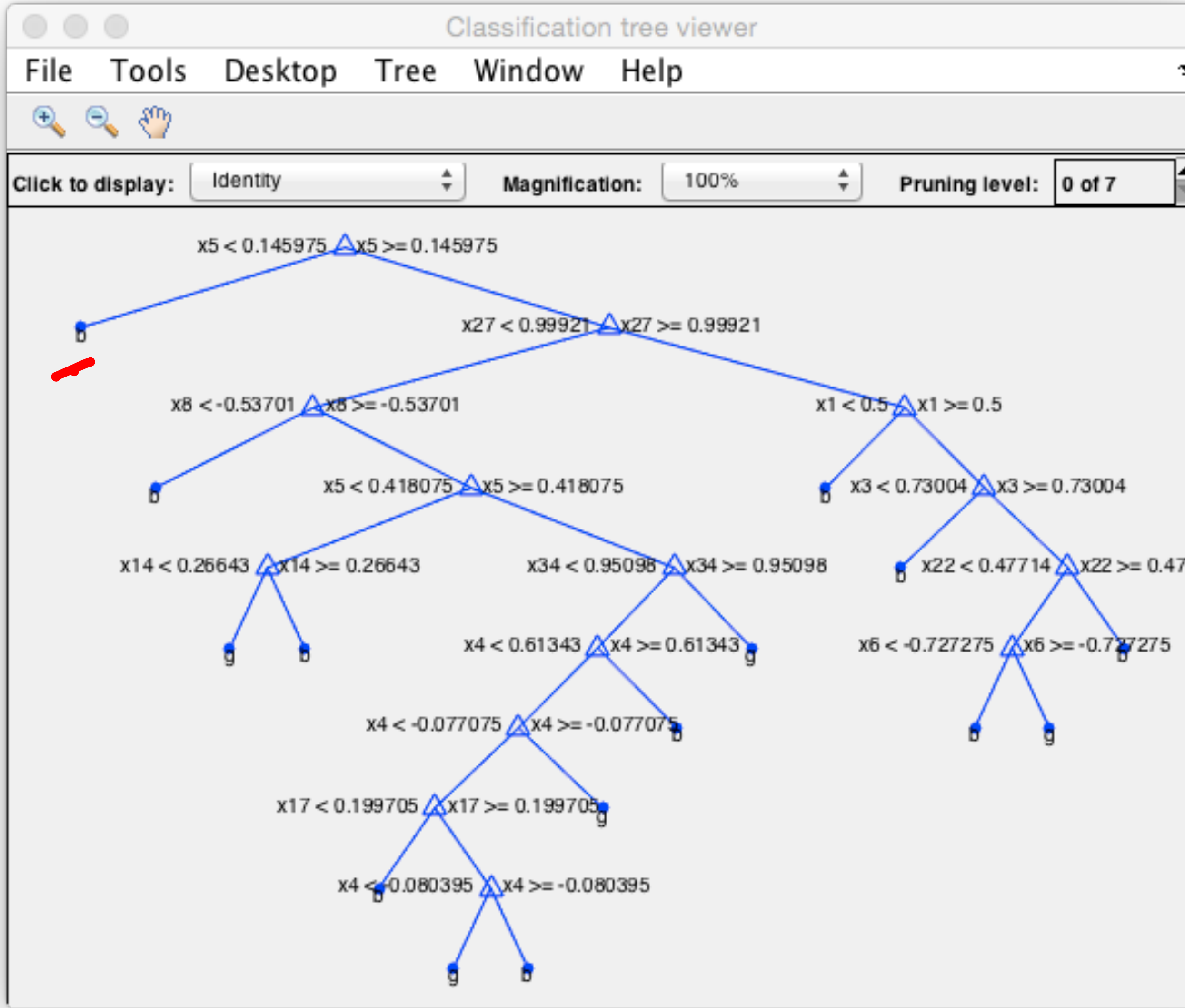
cvmodel = crossval(tc);

view(cvmodel.Trained{1},'Mode','graph')

kfoldLoss(cvmodel)

Validation error = 0.1254

Matlab example – decision tree

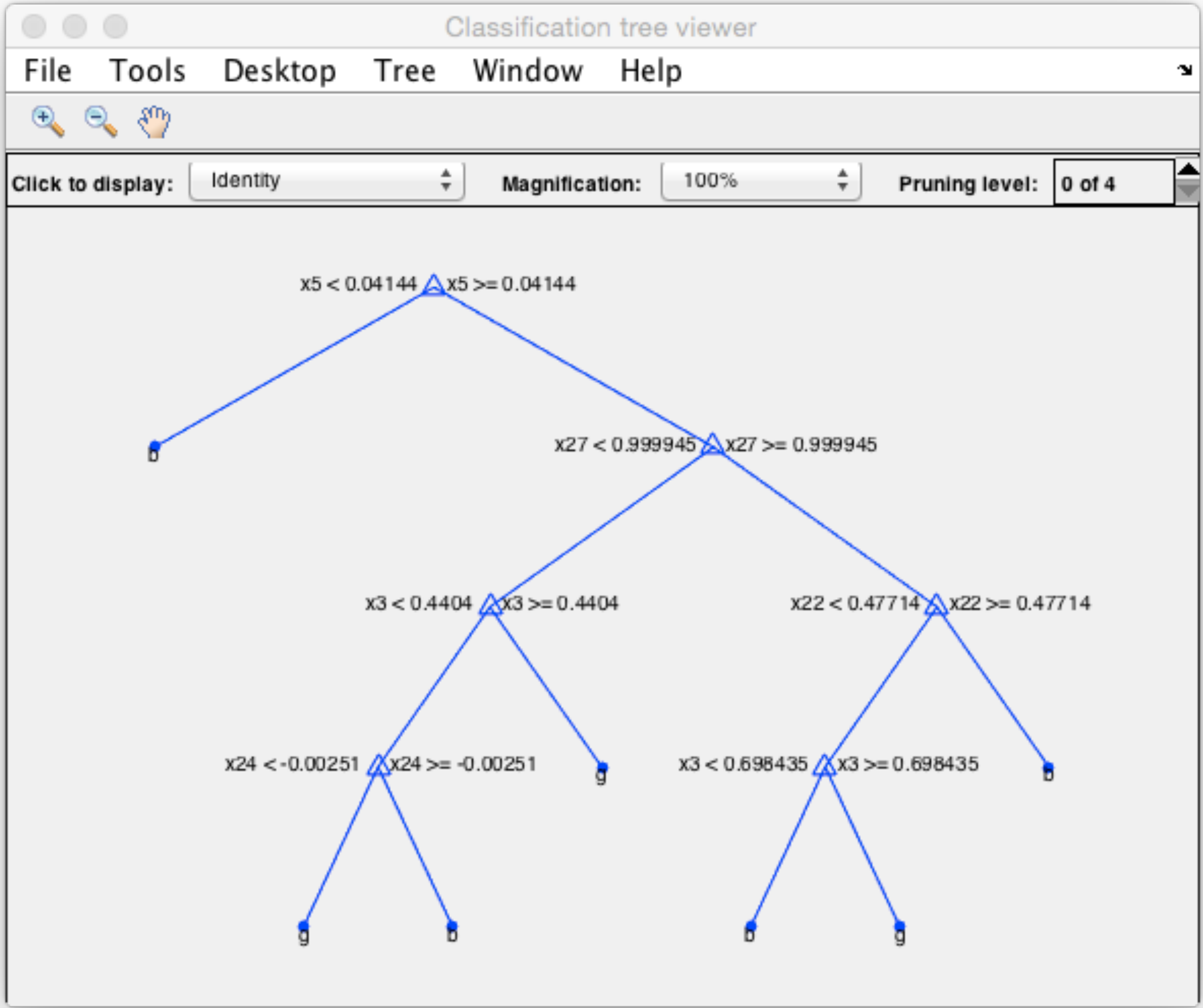



```
load ionosphere
% UCI dataset
% 34 features, 351 samples
% binary classification
rng(100)

%Default MinLeafSize = 1
tc = fitctree(X,Y, 'MinLeafSize',2);
cvmodel = crossval(tc);
view(cvmodel.Trained{1},'Mode','graph')
kfoldLoss(cvmodel)
```

Validation error = 0.1168

Matlab example – decision tree

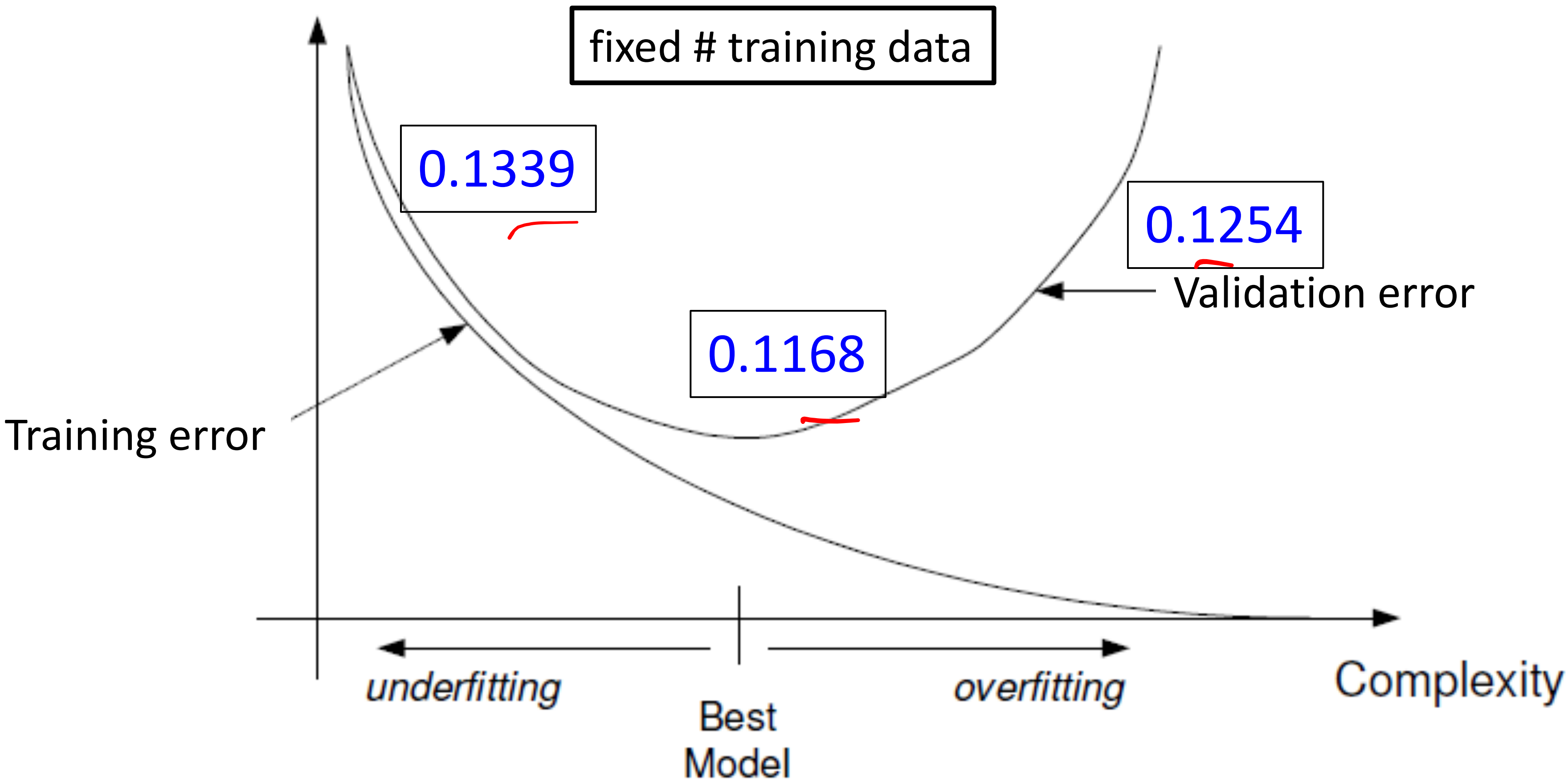



```
load ionosphere
% UCI dataset
% 34 features, 351 samples
% binary classification
rng(100)

%Default MinLeafSize = 1
tc = fitctree(X,Y, 'MinLeafSize',10);
cvmodel = crossval(tc);
view(cvmodel.Trained{1},'Mode','graph')
kfoldLoss(cvmodel)
```

$\boxed{\text{Validation error = 0.1339}}$

Matlab example – decision trees

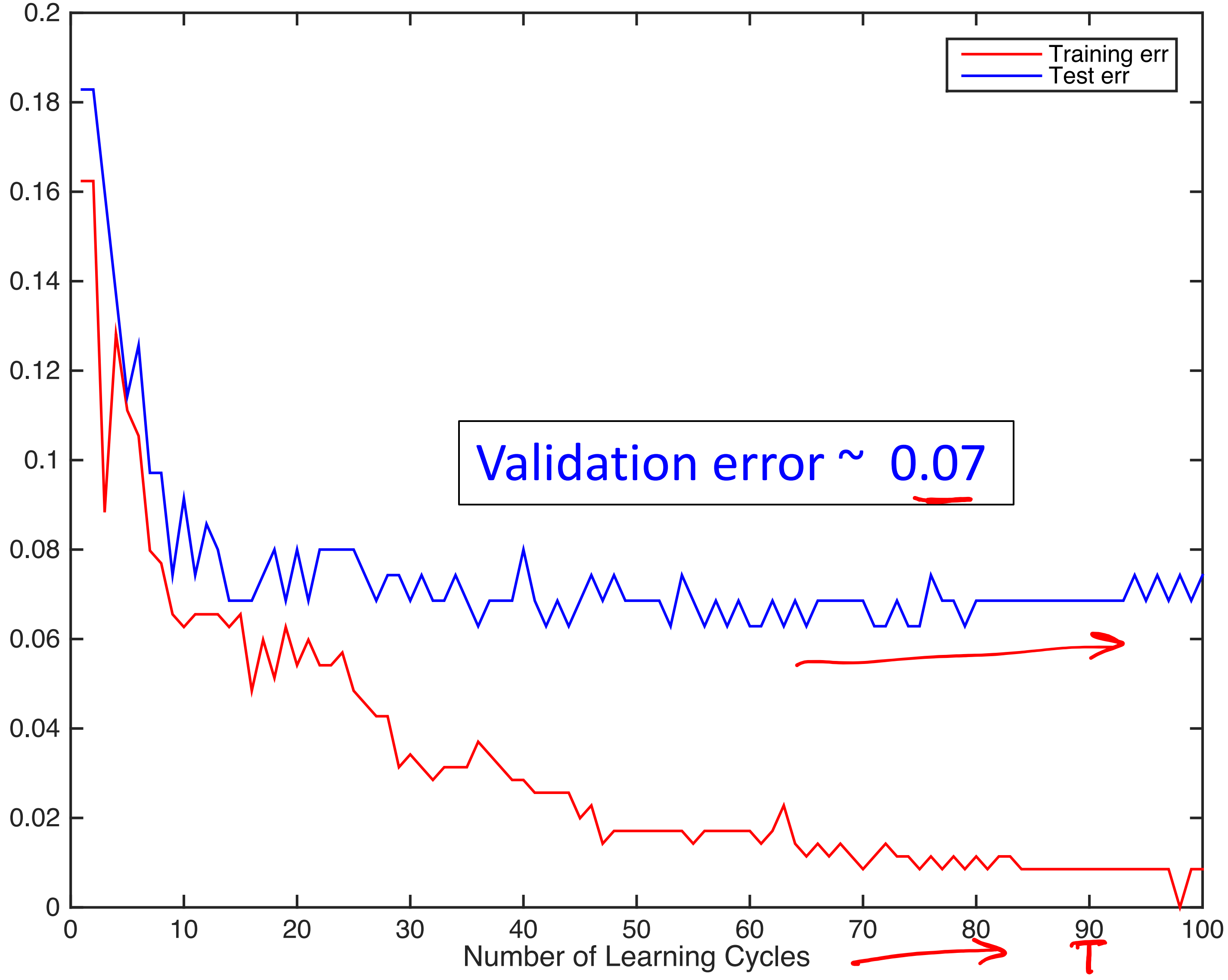


MinLeafSize 10 MinLeafSize 2 MinLeafSize 1

Matlab example - boosting

- % UCI dataset
- % 34 features, 351 samples
- % binary classification
- load ionosphere;
- rng(2); % For reproducibility
- ClassTreeEns = fitensemble(X,Y,'AdaBoostM1',100,'Tree');
- rsLoss = resubLoss(ClassTreeEns,'Mode','Cumulative');
- plot(rsLoss,'r');
- hold on
- ClassTreeEns = fitensemble(X,Y,'AdaBoostM1',100,'Tree',...
- 'Holdout',0.5);
- genError = kfoldLoss(ClassTreeEns,'Mode','Cumulative');
- plot(genError,'b');
- xlabel('Number of Learning Cycles');
- legend('Training err', 'Test err')

Matlab example - boosting



Bagging (Bootstrap aggregating)

[Breiman, 1996]

Related approach to combining classifiers:

1. Run independent weak learners on subsampled data (sample with replacement) from the training set
2. Average/vote over weak hypotheses

Bagging	vs.	Boosting
Resamples data points		Reweights data points (modifies their distribution)
Weight of each classifier is the same		Weight is dependent on classifier's accuracy
Only variance reduction		Both bias and variance reduced – learning rule becomes more complex with iterations
Can be trained in parallel		Trained sequentially

Random Forest

Related approach to combining decision trees:

1. Train decision trees on subsampled data (sample with replacement) from the training set + using **feature bagging** (random subset of features considered at each node)
2. Average/vote over decision trees

Random forest	vs. Boosted decision trees
Resamples data points	Reweights data points (modifies their distribution)
Weight of each classifier is the same	Weight is dependent on classifier's accuracy
Only variance reduction	Both bias and variance reduced – learning rule becomes more complex with iterations
Typically complex decision trees	Typically uses decision stumps
Can be trained in parallel	Trained sequentially

Boosting Summary

- Combine weak classifiers to obtain strong classifier
  - Weak classifier – slightly better than random on training data
  - Resulting very strong classifier – can eventually provide zero training error
- AdaBoost algorithm ✓ α_t, D_t
- Boosting v. Logistic Regression
  - Similar loss functions
  - Single optimization (LR) v. Incrementally improving classification (B)
- Most popular application of Boosting:
  - Boosted decision stumps!
  - Very simple to implement, very effective classifier

Comparison chart (classification)

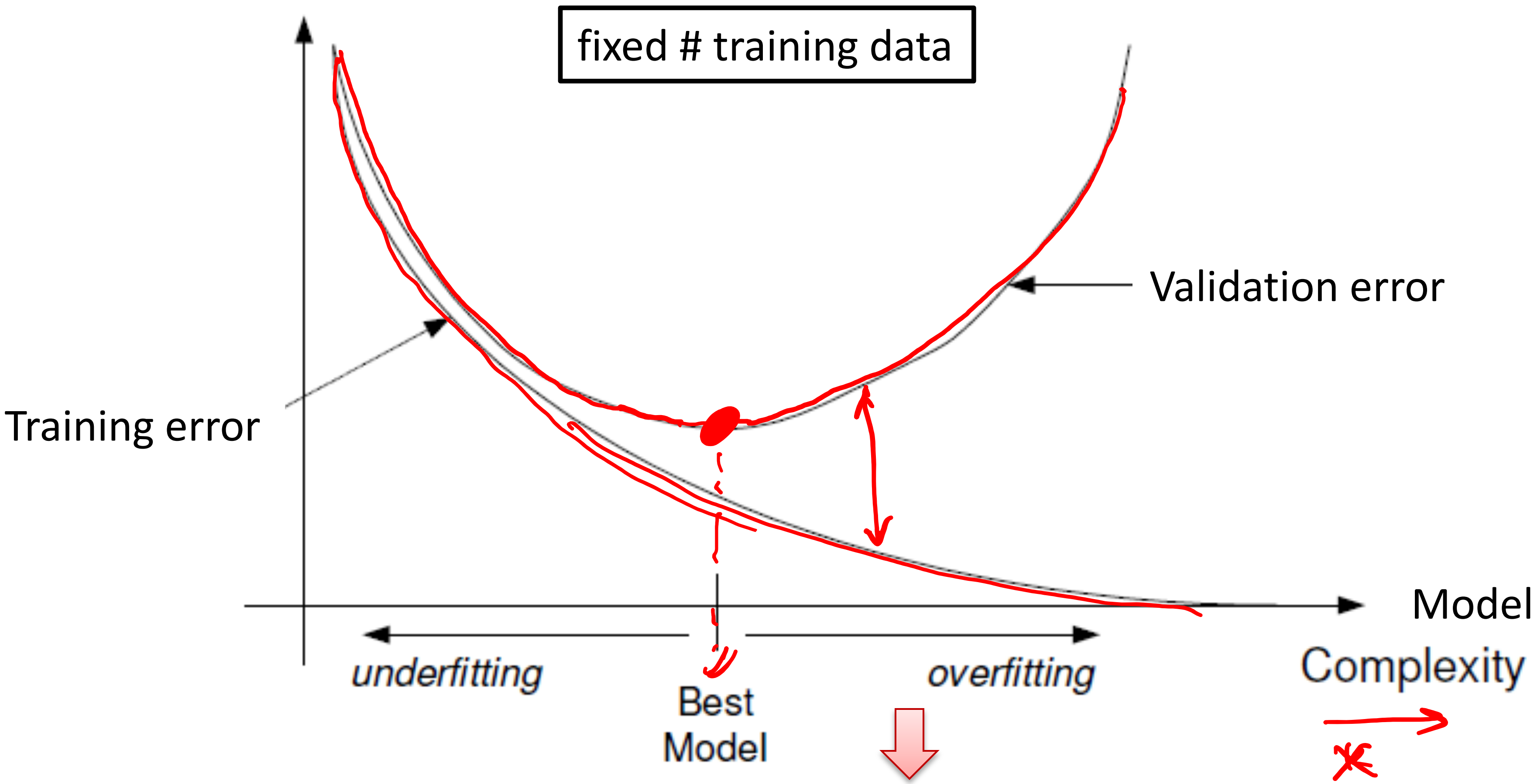
Algorithm	Generative/ Discriminative	Assumptions	Decision boundary	Loss function	Training
Naïve Bayes	G	$p(X,Y) = p(X\|Y)p(Y) = \prod_i p(X_i\|Y)p(Y)$	linear or quad. (Gaussian $p(x\|y)$)	max likelihood or MAP	closed-form / gradient ascent
Logistic Regression		$P(Y\|X) = \frac{1}{1+\exp(\ )}$	linear	max conditional likelihood / log	gradient ascent / descent
SVM Kernel SVM			linear non-linear	hinge	quad prog. primal/dual
Neural Networks	D		non-linear	ℓ_2/cross-entropy	SGD 'Adam'
k-Nearest Neighbors		non-parametric	"	plug-in Bayes	look up data
Decision Tree			"	Inf gain – ID3/C4.5 penalized $\ell_2/\|\cdot\|$ – CART	
Boosting			"	exp	Adaboost

Model selection

Aarti Singh

Machine Learning 10-701
Mar 15, 2023

Training vs. Test Error



Training error is no longer a good indicator of test error

Examples of Model Spaces

Model Spaces with varying complexity:

- Nearest-Neighbor classifiers with increasing neighborhood sizes $k = 1,2,3,\ldots$

  Large neighborhood => low complexity

- Decision Trees with increasing depth k or with k leaves

  Higher depth/ More # leaves => high complexity

- Neural Networks with increasing layers or nodes per layer

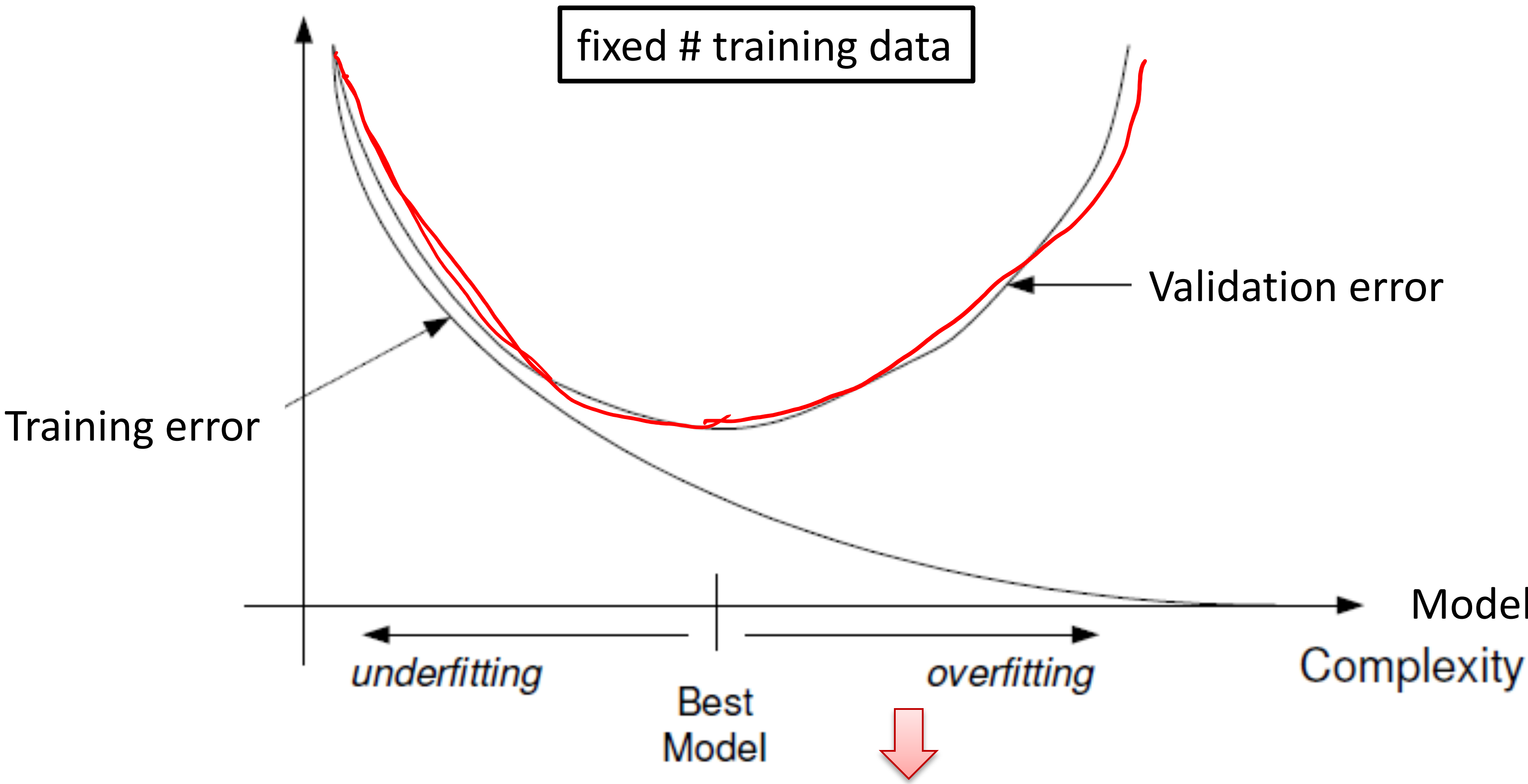
  More layers/Nodes per layer => high complexity

- MAP estimates with stronger priors (larger hyper-parameters β_H, β_T for Beta distribution or smaller variance for Gaussian prior)

  => low complexity

How can we select the right complexity model ?

Training vs. Test Error



Training error is no longer a good indicator of test error

Bias-Variance Tradeoff

$E[f_n] \simeq f^*$

$f_n \simeq E[f_n]$

$f_n \neq f^*$

- Why does test/validation error go down then up with increasing model complexity?

Two sources of error:

e.g. Regression

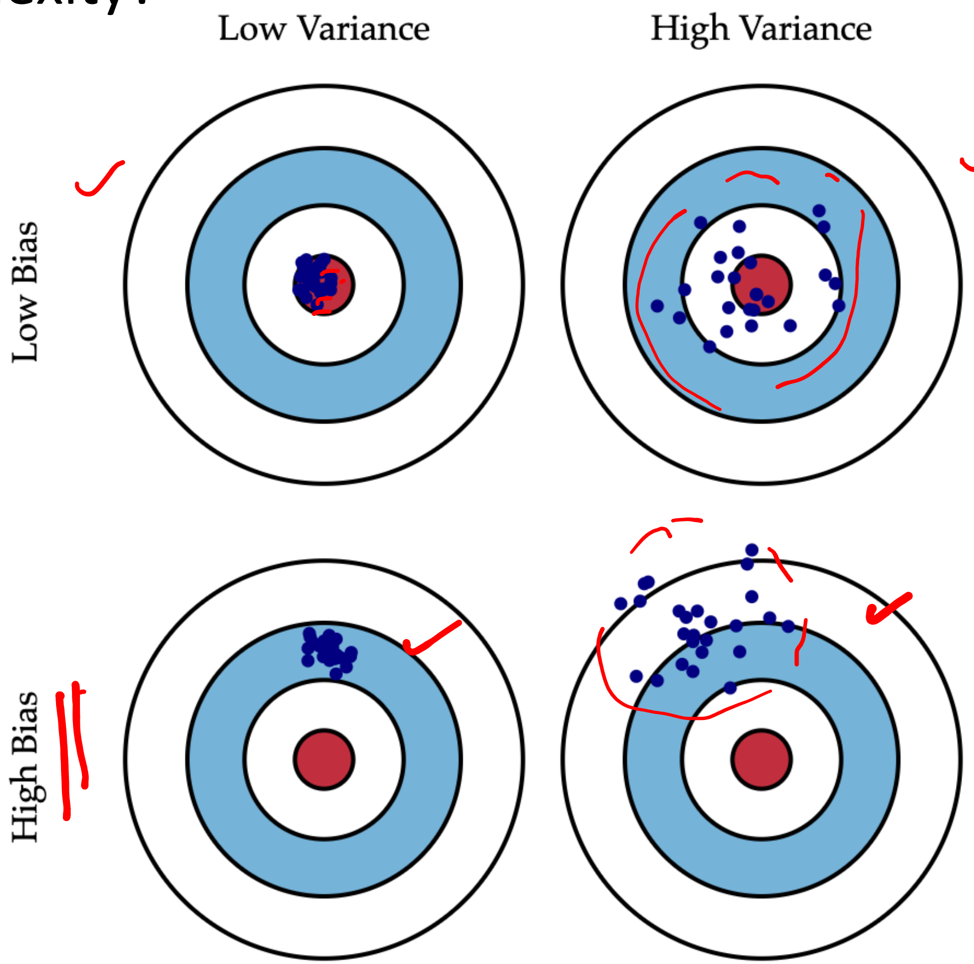
Bias^2

ground truth

$$|E[f_n] - f^*|^2$$

Variance

$$E[\,|f_n - E[f_n]|^2]$$

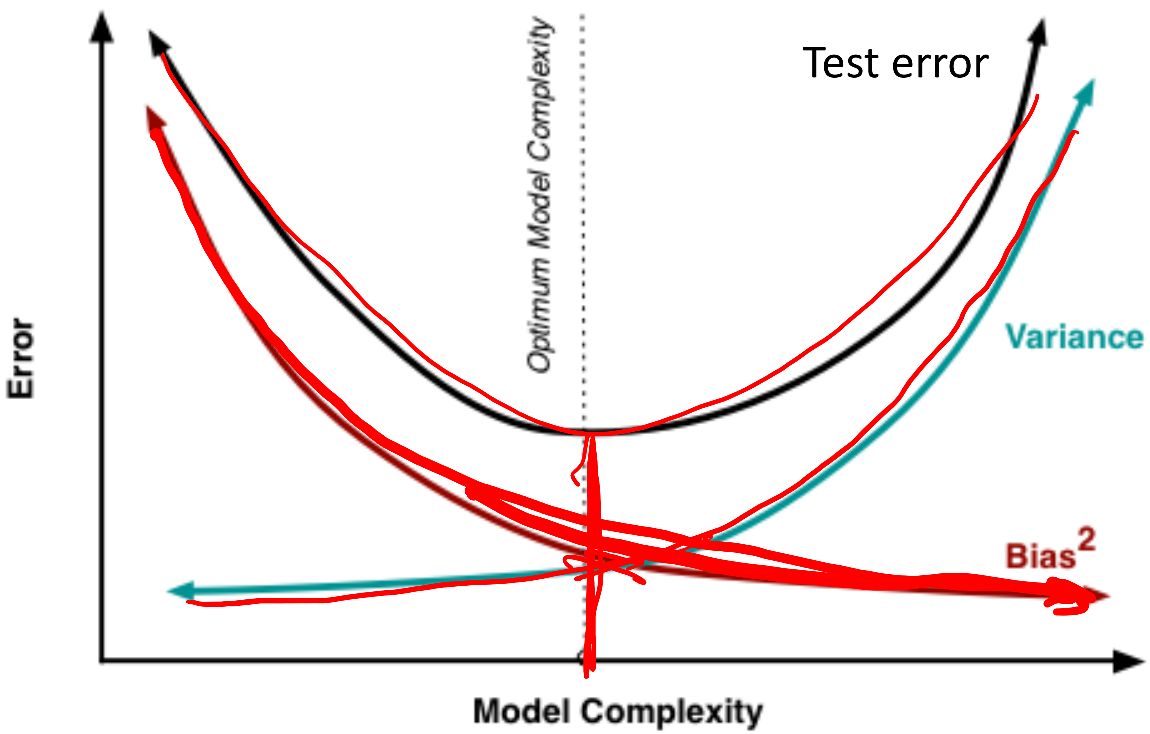


Bias-Variance Tradeoff

- Why does test/validation error go down then up with increasing model complexity?

Bayes error

Mean square test error = Variance + Bias^2 + Irreducible error



Judging Test error

- Training error of a classifier f

$$\frac{1}{n}\sum_{i=1}^{n} 1_{f(X_i)\neq Y_i}$$

Training Data

$$\{X_i, Y_i\}_{i=1}^{n}$$

- What about test error?

  Can't compute it.

- How can we know classifier is not overfitting?

  Hold-out or Cross-validation

Hold-out method

Can judge test error by using an independent sample of data.

Hold – out procedure:

n data points available $D \equiv \{X_i, Y_i\}_{i=1}^n$

1) Split into two sets (randomly and preserving label proportion):

Training dataset | Validation/Hold-out dataset

$D_T = \{X_i, Y_i\}_{i=1}^m$ | $D_V = \{X_i, Y_i\}_{i=m+1}^n$

often m = n/2

2) Train classifier on D_T. Report error on validation dataset D_V.
Overfitting if validation error is much larger than training error

Hold-out method

Drawbacks:

- May not have enough data to afford setting one subset aside for getting a sense of generalization abilities
- Validation error may be misleading (bad estimate of test error) if we get an “unfortunate” split

Limitations of hold-out can be overcome by a family of sub-sampling methods at the expense of more computation.