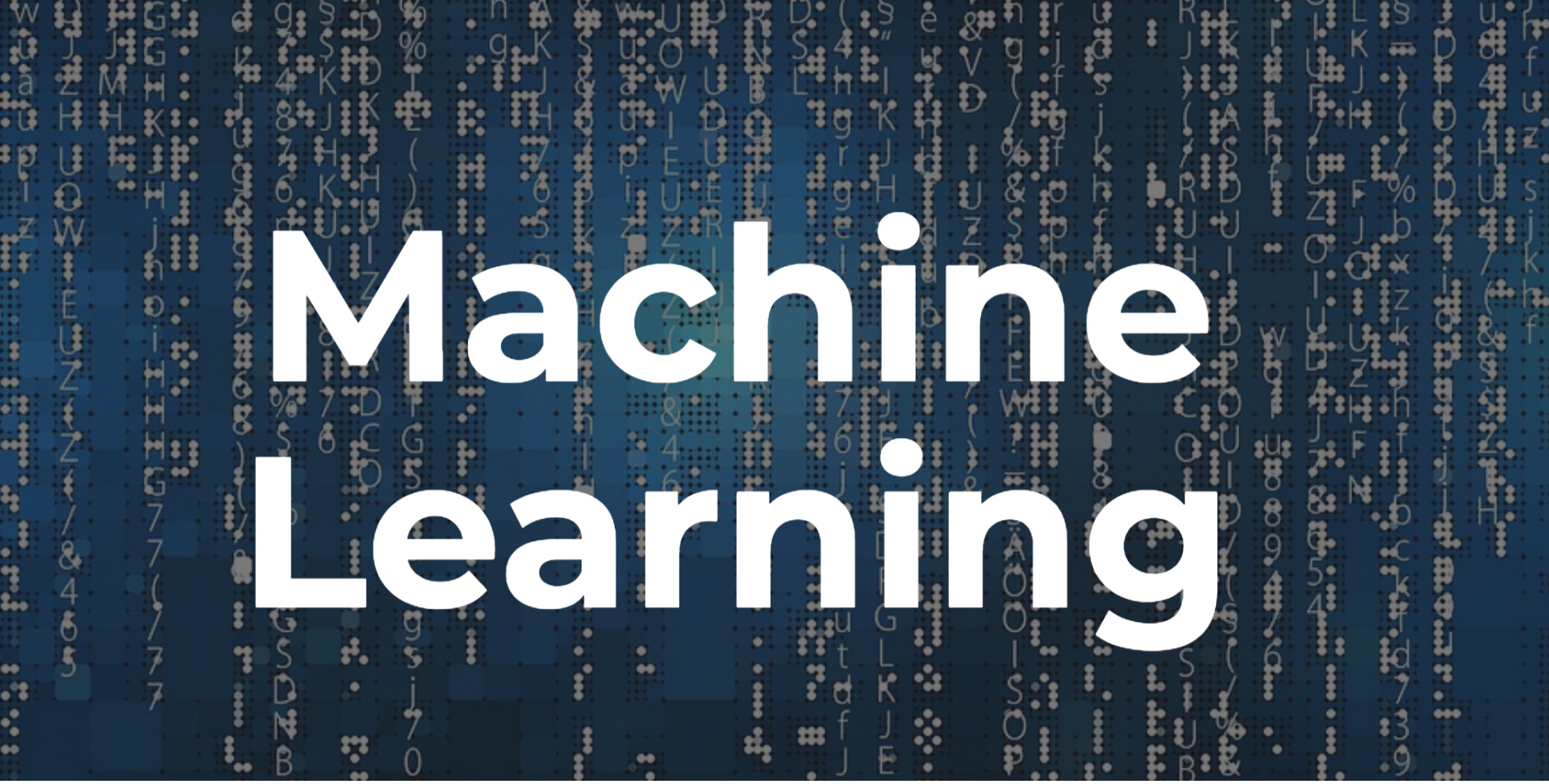


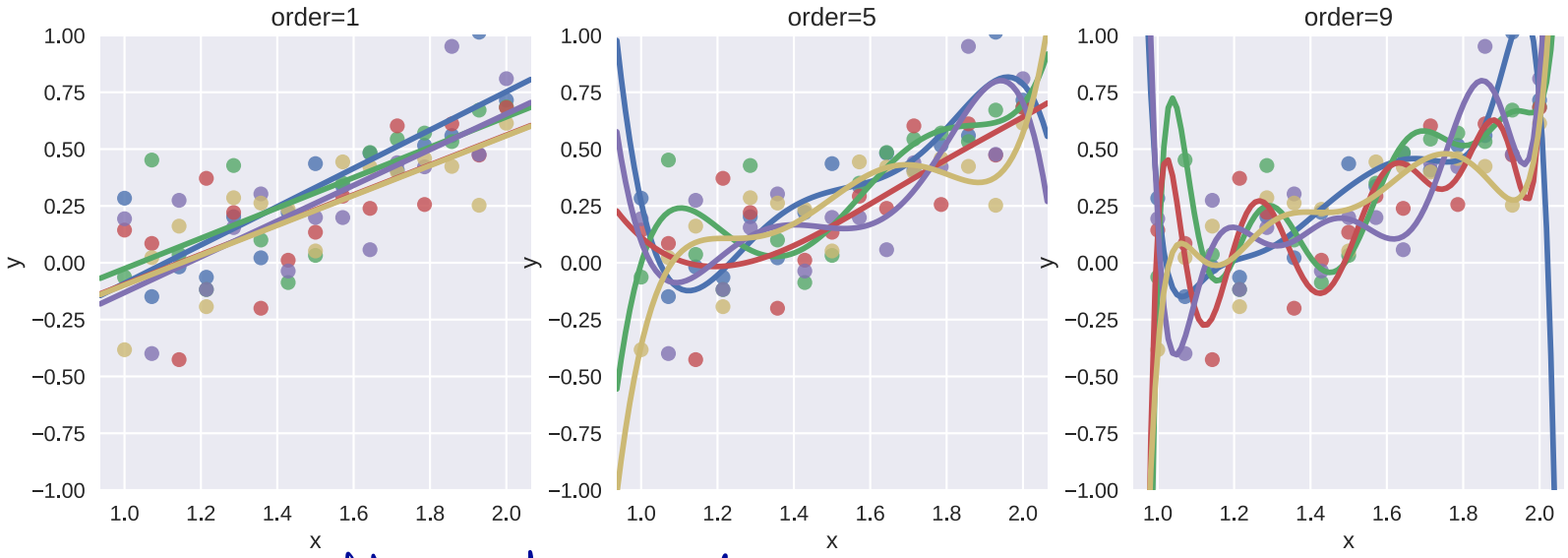
CSCI 4622 Fall 2019

Prof. Claire Monteleoni

Today: Lecture 7

- Bias/variance trade-off (continued)
- Validation, cross-validation (continued)
- Learning Theory
  - Complexity of classifiers
    - VC dimension
    - Margin
    - Sauer's lemma (if time)

Increasing model complexity



Data look better when order increase

Increasing the order of the polynomial model used to fit the data increases model complexity, eventually leading to overfitting.

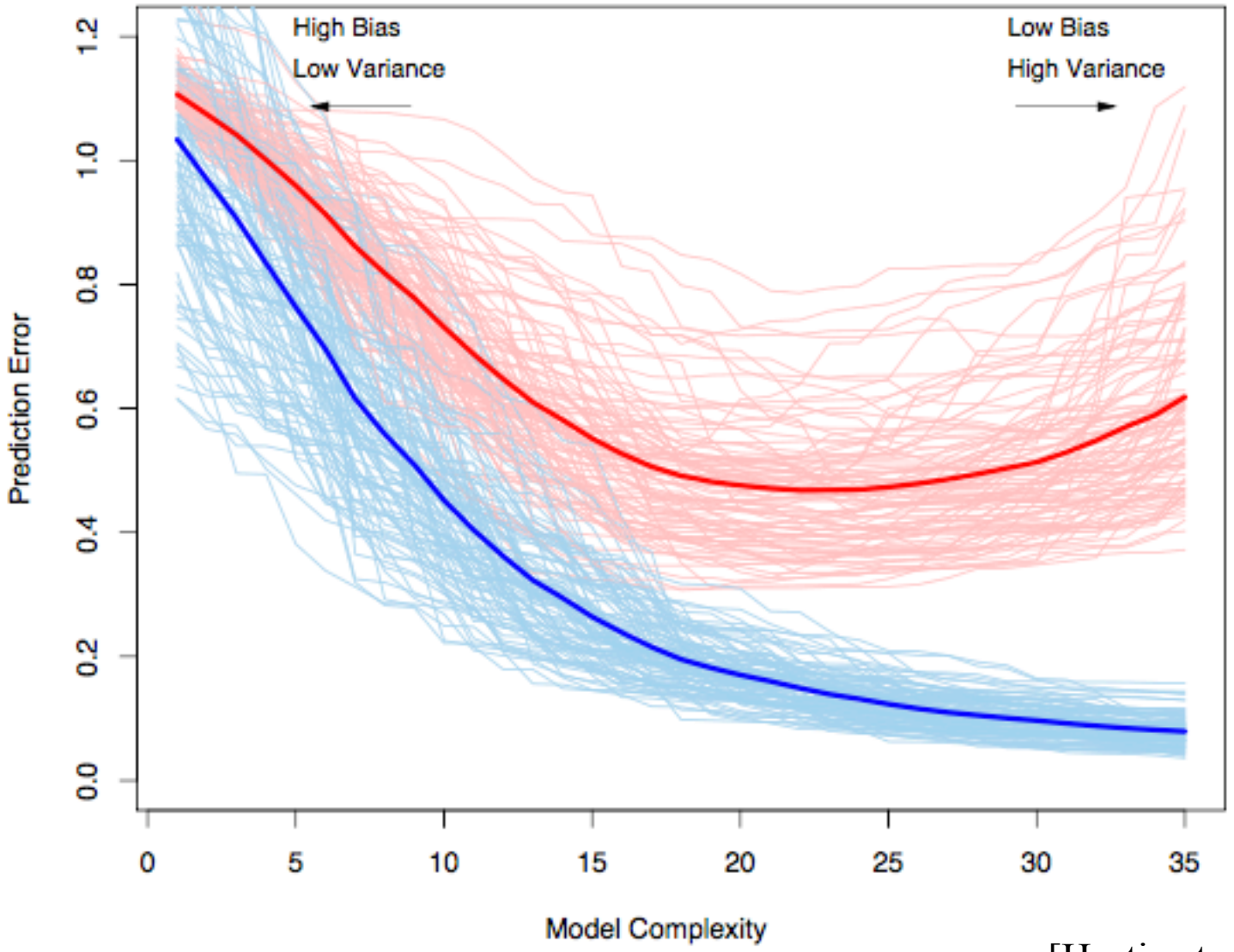
Credit: C. Tan

Bias-Variance Tradeoff

Allowing the model (the hypothesis class) to be more and more complex, the resulting classifier, *h*, will have a better and better fit to the training data, thus the bias of *h* reduces.

But as complexity increases, *h* will increasingly overfit to the chance structure of the particular training data set. So there will be higher variance of the true error of classifiers, *h'*, learned over different training sets.

Example



Test error

Training error

[Hastie et al., 2001. Fig. 7.1]

Bias-Variance Error Decomposition

If we assume all data points (x,y) obey: $y = f(x) + \epsilon$

where: $\mathrm{E}[\epsilon] = 0$

$\mathrm{Var}(\epsilon) = \sigma^2$

$E[y] = E[f(x) + z]$

$= E[f(x)] + E[z]$

$= f(x) + 0$

$= f(x)$

Then, $\mathrm{Err}_{\mathrm{h}}(x) = \mathrm{E}[(y - h(x))^2 | X = x]$

h is classifier from learning data

$= \sigma^2 + \mathrm{Var}[h] + (\mathrm{Bias}[h])^2$

Irreducible error (can't be changed by choice of h)

where: $\mathrm{Var}[h] = \mathrm{E}[h(x)^2] - \mathrm{E}[h(x)]^2$

$\mathrm{Bias}[h] = (f(x) - \mathrm{E}[h(x)])$

NOTE: All expectations are w.r.t. the random training set *h* is learned from, NOT *x*.

$\hat{f}$ is the output of learning algorithm. = evaluate at (x, y), a labeled data of that was not in our training set

generalization error of $\hat{f}$:

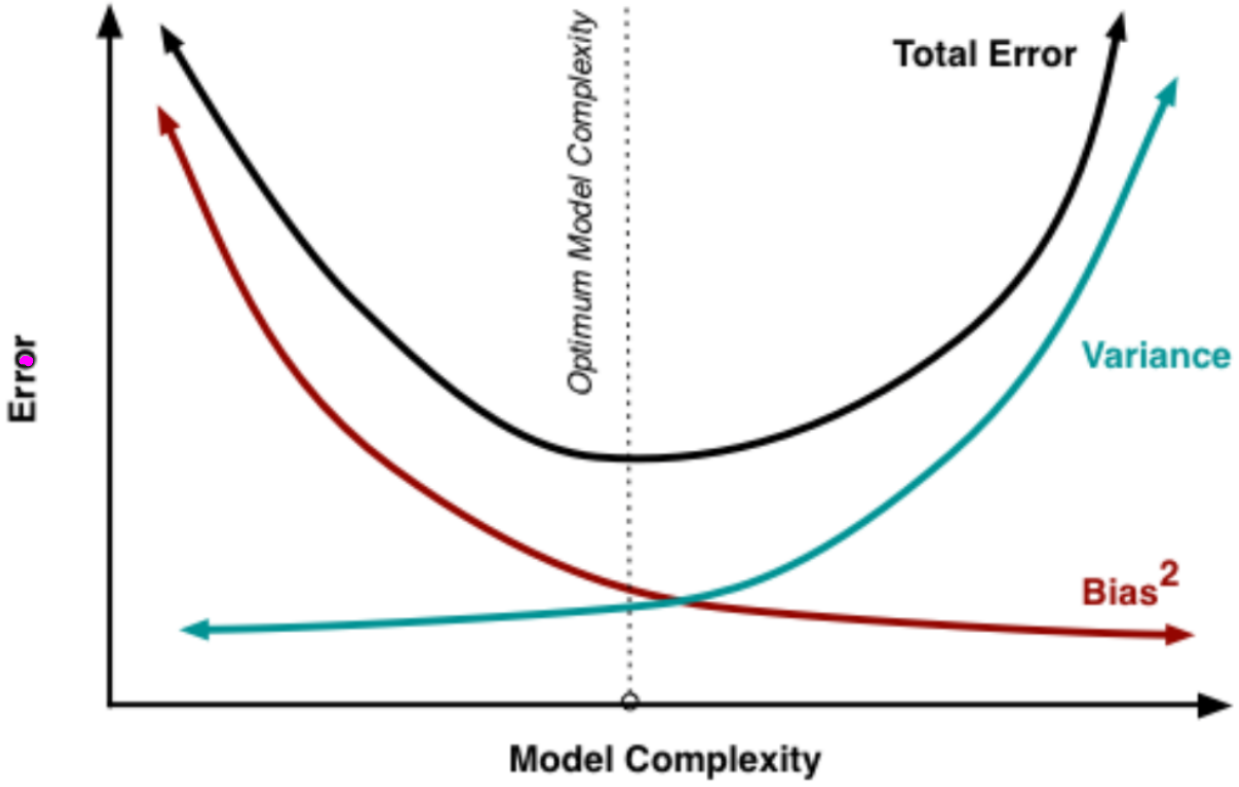
$$\text{Var}[a] = E[a^2] - (E[a])^2$$
$$E[a^2] = \text{Var}[a] + (E[a])^2$$

$$\begin{aligned}
\text{err}(\hat{f}(x)) &= E[(y - \hat{f}(x))^2] \\
&= E[y^2 + (\hat{f}(x))^2 - 2y\hat{f}(x)] \\
&= E[y^2] - E[\hat{f}(x)] - 2E[y\hat{f}(x)] \\
&= \text{var}[y] + (E[y])^2 + \text{var}(\hat{f}(x)) + (E[\hat{f}(x)])^2 \\
&\quad - 2\hat{f}(x)E[y] \\
&= \text{var}[y] + (f(x))^2 + \text{var}[\hat{f}(x)] + (E[\hat{f}(x)])^2 - 2f(x)\hat{f}(x) \\
&= \text{var}(y) + \text{var}(\hat{f}(x)) + (f(x) - \hat{f}(x))^2 \\
&= \sigma^2 + \text{var}(\hat{f}) + (\text{Bias})^2
\end{aligned}$$

because $E[y] = f(x)$

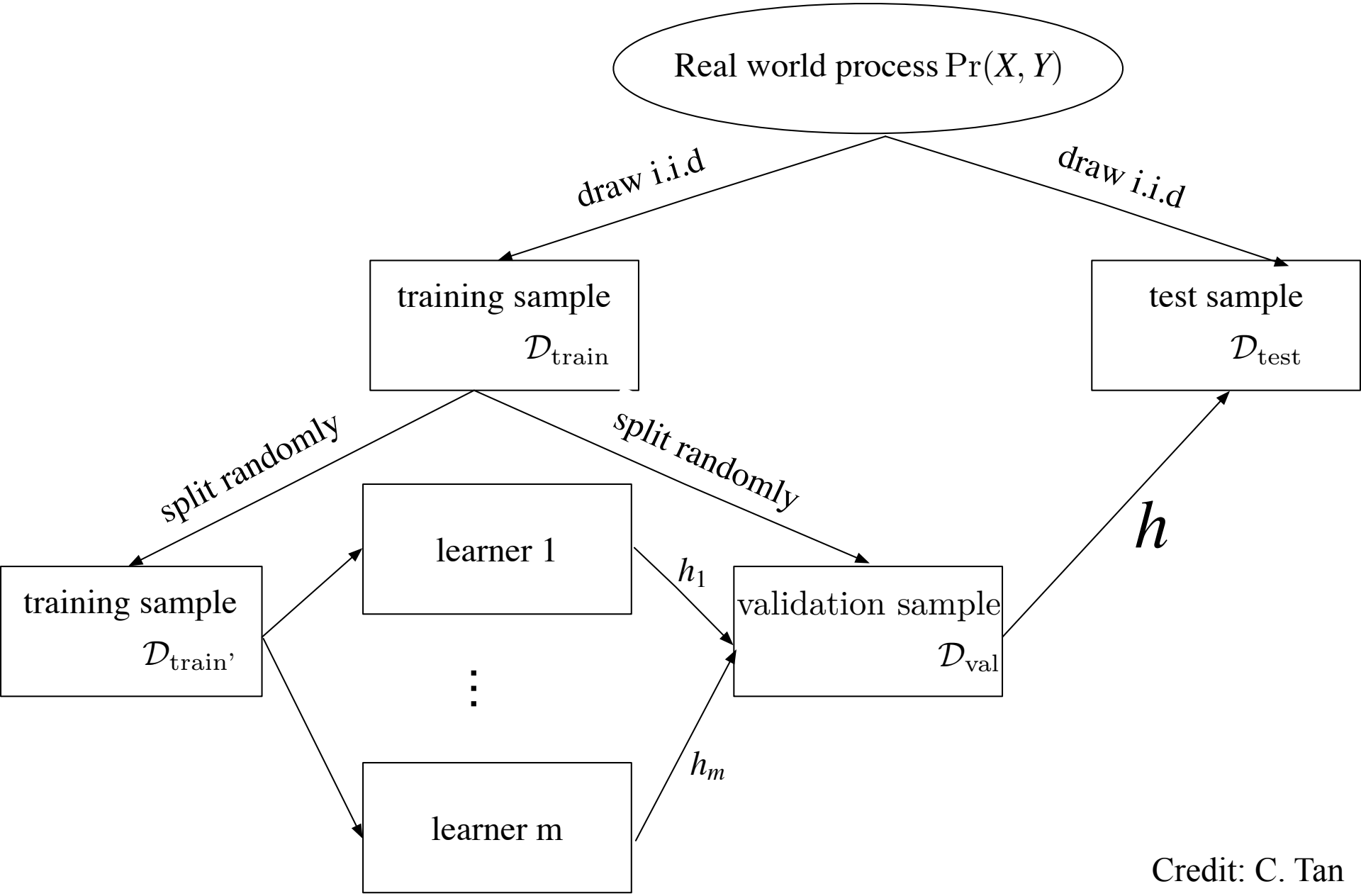
Bias-Variance Tradeoff

Generalization error = Irreducible error + Bias^2 + Variance



Credit:
http://scott.fortmann-roe.com/docs/BiasVariance.html

Model Selection



Credit: C. Tan

Managing data sets: Train v. Test

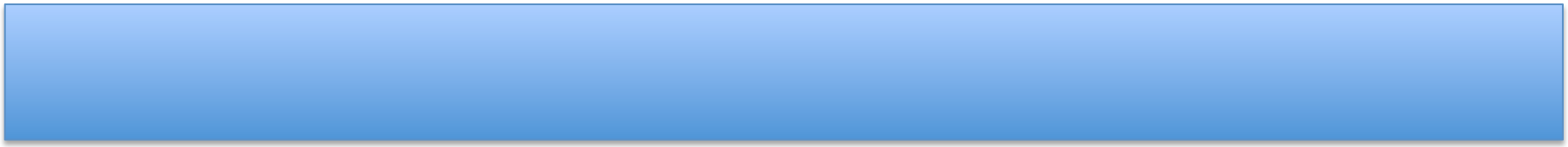
- In general, you should have a separate training set and test set.
- Ideally they will be drawn from the same distribution.
- E.g. randomly permute your entire input data set, then fix T, and use all (x_t,y_t) for $t < T$ for Train, and the remaining data for Test.

Managing data sets: Train v. Test

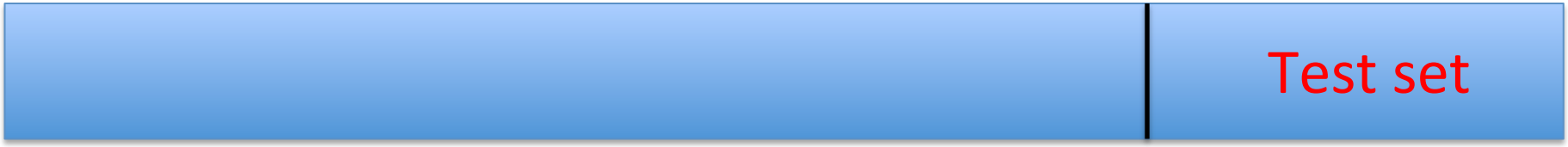
- First create the holdout data set, the Test set, for computing the final test error. For best performance it will be a (labeled) dataset from the same distribution as the Training set.
  - E.g. technique on previous slide.

- With the remaining data set, you can either divide it into two parts for Train and Validation, as specified on the previous slide, or you can run Cross-Validation on it.
  - This is for model selection / parameter tuning

- Whole data set:



- Separate the test set, and hold it out:



- From the remaining data, use part for Validation:

- Or run cross validation to train and fit parameters:

Cross validation: motivation

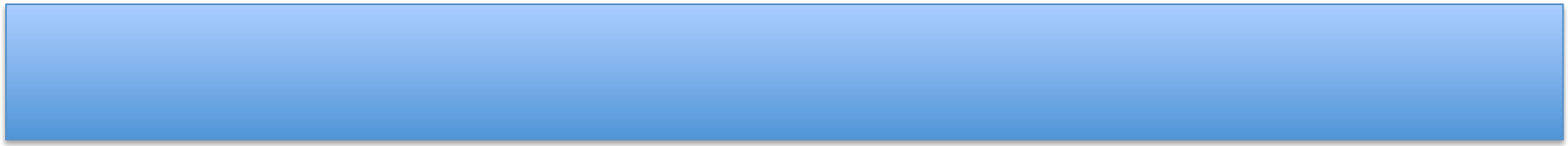
- To get a more robust estimate of the error rate, we should run more experiments, and average the error rates over the experiments.

- If we have a huge amount of labeled data, we can run several disjoint experiments (e.g. further split Train and Test into more data sets) and average the test error over the results. This is usually not the case.
  - Instead we do Cross-Validation.

N-fold Cross-validation

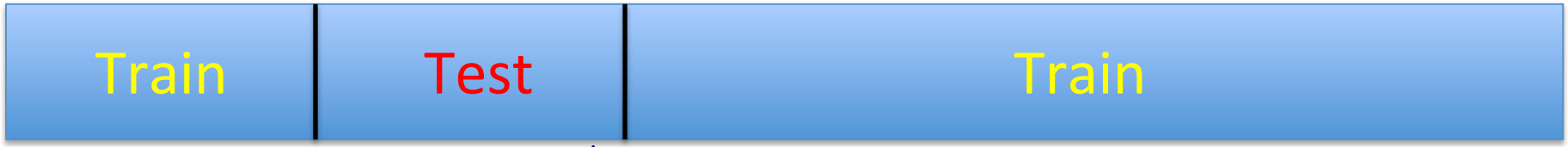
fold is not greater than data point.

- Cross validation data set:



- Fold 1:

- Fold 2:



five fold = mean we do cross validation 5 five.

- …
- Fold n

N-fold cross-validation

- Fix N, for example 5 or 10.
- Run N experiments. In each experiment, 1/N fraction of the data is used for Test and the remaining data is used for Train. The Test sets over the N experiments are disjoint, and the Train sets are overlapping.
- Average the Test error over the N experiments.
- Note:, different classifiers will likely be learned in each experiment as the Train sets differ slightly.

Leave-one-out cross-validation

- N is set to the size of the data set. For each experiment, Test consists of the i-th point, and Train consists of all points exept the i-th point.
- Average the test error over the N experiments.

Model selection / Parameter tuning

- For each meta-parameter, e.g. *k* in *k*-NN, explore a range of parameter values. Log-search can be performed by doubling, or halving the value of the parameter.

- On the tuning data set (a.k.a. Validation set, or CV on the training data), try to find the "knee" in the curve: the parameter setting such that the tuning test error is minimized, before increasing again.

Complexity of classifiers

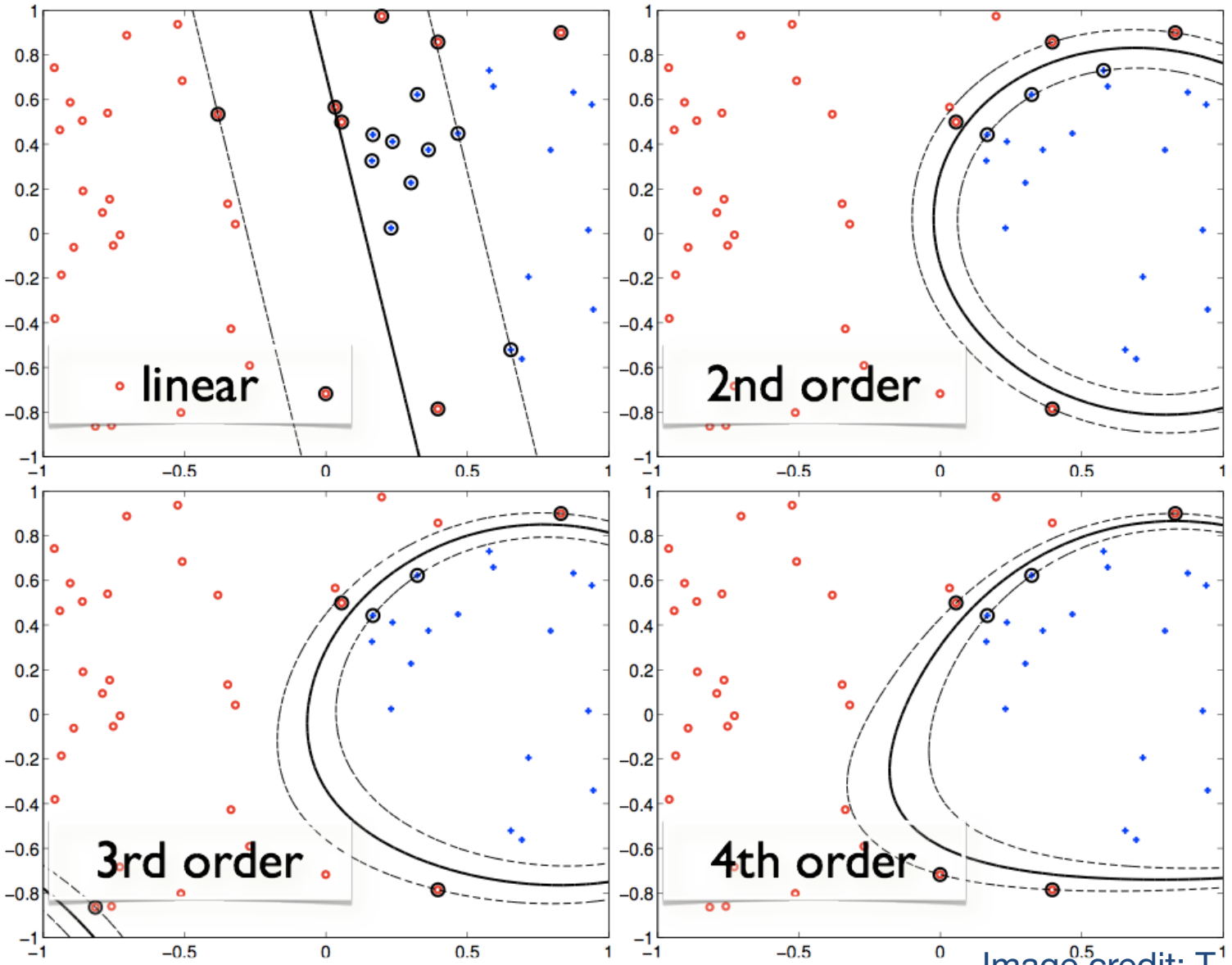


Image credit: T. Jaakkola

VC dimension

what we learn over training data

We have seen various different hypothesis classes, e.g. linear separators, polynomial separators, decision trees, nearest-neighbor classifiers, etc.

How do we compare them? One way is to define a notion of the level of complexity of a hypothesis class H.

Complexity can be defined in various ways.

Complexity and the ability to generalize to future data are inversely related (as we have mentioned repeatedly!)

→ i.e. as complexity of the classifier increases, it can lead to overfitting.

VC dimension = is the whole dimension of hypothesis class

Definition: Vapnik-Chervonenkis (VC) dimension.

The VC-dimension of a hypothesis class H is the maximum number of points that a classifier can shatter.

This is a measure of the complexity of H.

To prove that the VC dimension of a class H is V, you must prove both of the following:

- H can shatter V points.
- H cannot shatter V+1 points.

To show $VC(H) = d$

need to prove both:

① H can shatter d points

② H cannot shatter $d+1$ points

H can shatter n pts $\Leftrightarrow$

$\exists$ a set of n pts, S

$\forall$ possible labeling of S

$\exists$ $h \in H$ that obtain that labeling of S

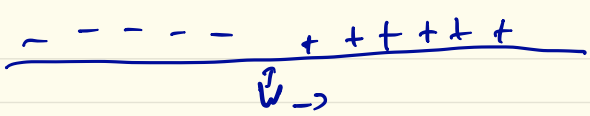
H cannot shatter pts

$\forall$ sets of n pts, S

$\exists$ a labeling of S

$\forall$ $h \in H$, h can't obtain that labeling

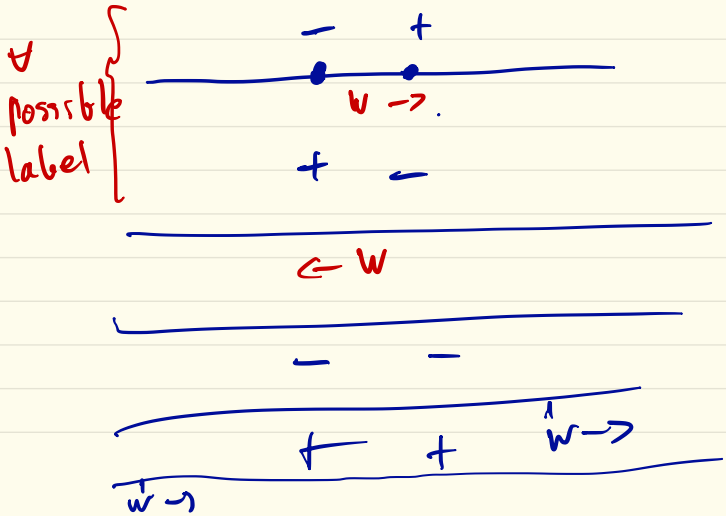
H = Decision stump in $\mathbb{R}^1$



w: $w(x) = \begin{cases} +1 & x \geq w \\ -1 & x < w \end{cases}$

$w(x) = \begin{cases} +1 & x < w \\ -1 & x \geq 0 \end{cases}$

wts $VC(H) \geq 2$ [wt.s H can shatter 2 pts



To show wts $VC(H) < 3$ [wts H cannot shatter 3 pts]

$\Rightarrow \underline{VC(H) = 2}$

Shattering

Definition: A hypothesis class H can shatter n points if there exists a set of n points such that for every possible labeling of the points, there exists some h in H that can attain that labeling.

To prove that a hypothesis class H can shatter n points, you must:

$\exists$ give a point set S of size n

$\forall$ and show that for all possible labelings of S

$\exists$ there exists some h in H which achieves that labeling.

Shattering

To prove that a hypothesis class H cannot shatter n points you need to you need to show that:

- $\forall$ for any set of n points
  - $\exists$ there exists some labeling of that configuration
    - $\forall$ such that no h in H can achieve that labeling of that configuration (i.e. for all h in H, h cannot)

VC dimension

- Example: what is the VC dimension of decision stumps in R^1?

- Example: what is the VC dimension of decision stumps in R^2?

- Example: what is the VC dimension of 1-nearest-neighbor classifiers?

VC dimension

- Example: What is the VC dimension of linear classifiers in the plane (R^2)?

- Example: What is the VC dimension of linear classifiers in R^d?

VC dimension

- The VC dimension of Linear Classifiers in R^d is:

- If the data has margin r, and B upper bounds the norm of all points, then the VC dimension of Linear Classifiers in R^d is

  $V \leq (B/r)^2$ [like perceptron mistake bound!]

- This means even if $d = \infty$ we can use linear separators and not suffer from high complexity, if there's a margin!
- Margin is therefore another measure of complexity.

Sauer's Lemma

Suppose you are given m (unlabeled) points.

Before we introduce Sauer's lemma, what's an upper bound on the number of possible (binary) labelings of the m points?

If we know that the VC dimension, V, of a class H is finite, then we can apply Sauer's lemma which says that the number of possible labelings of the m points that can be achieved by classifiers in H is $O(m^V)$.

As long as we have more points than the VC dimension (i.e. $m > V$), this is a much tighter bound, i.e. $O(m^V) \leq O(2^m)$

Sauer's Lemma

Formally, given a hypothesis class H define H(m) as the maximum number of ways to label any set of m points using hypotheses in H. Let V = VC-dimension(H) < ∞.

Sauer's Lemma:

$$H(m) \leq \sum_{i=1}^{V} \binom{m}{i} = O(m^V)$$

Effective size of H

Given a hypothesis class H, the size of H, |H|, is the number of classifiers in H.

This can be infinite, for example if H = {Linear Classifiers in R^d}

However, as soon as we fix a set of m (unlabeled) data points, M, the "effective" size of H becomes finite.

> Group the hypotheses into equivalence classes, where each class contains all hypotheses in H that output the same labeling on M.

A (potentially loose) upper bound on the effective size of H is:

If H has finite VC dimension, V, what's a tighter upper bound on the effective size of H?