

# Applied Machine Learning

Generalization

Reihaneh Rabbany

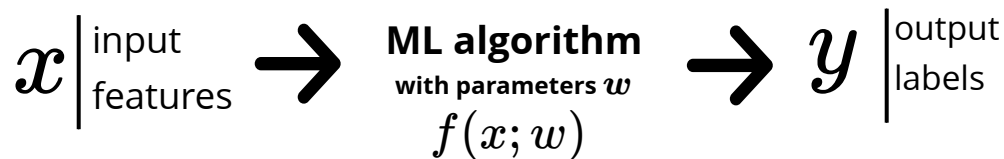


COMP 551 (winter 2023)

# Learning objectives

- generalization
- bias and variance trade-off
- validation and cross-validation
- evaluation

# Generalization



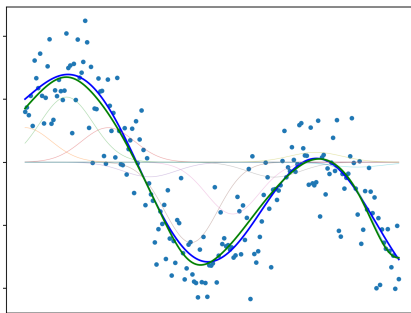
training: parameter estimation

$$\mathcal{D} = \{(x^{(n)}, y^{(n)})\}_{n=1}^N \quad \left| \quad \begin{aligned} w^* &= \arg \min_w J(w) \\ J(w) &= \frac{1}{N} \sum_{n=1}^N l(y^{(n)}, f(x^{(n)}; w)) \end{aligned} \right.$$

## Example: Linear regression

model:  $\hat{y} = f_w(x) = w^\top x : \mathbb{R}^D \rightarrow \mathbb{R}$

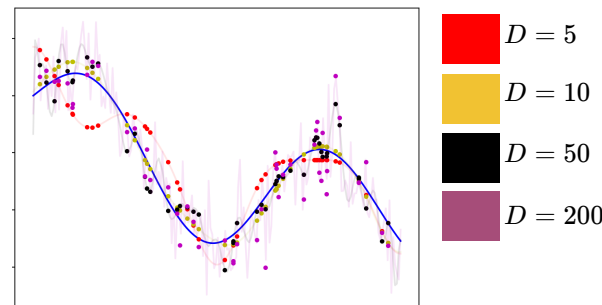
cost function:  $J_w = \frac{1}{N} \sum_n \frac{1}{2} (y^{(n)} - \hat{y}^{(n)})^2$



who the trained model generalizes to unseen data?  
how accurate is the model in general?

inference:  $\hat{y} = f(x; w^*)$

on unseen data for which we haven't seen the label

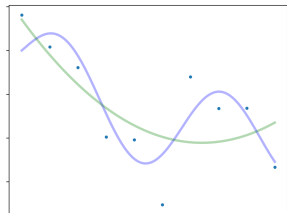


# Generalization and model complexity

simple models cannot fit the data

- large training error due to underfitting

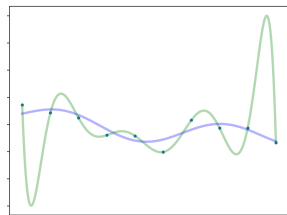
bias



expressive models can overfit the data

- small training error
- large test error due to overfitting

variance



**regularization** can help us trade-off between bias and variance

we want to see how these two terms contribute to the **generalization error**

# Generalization and model complexity

## example

**columns:** a different type of model  $g(x)$

**rows:** different datasets

datasets are from the same distribution

$$x^{(n)}, y^{(n)} \sim p(x, y)$$

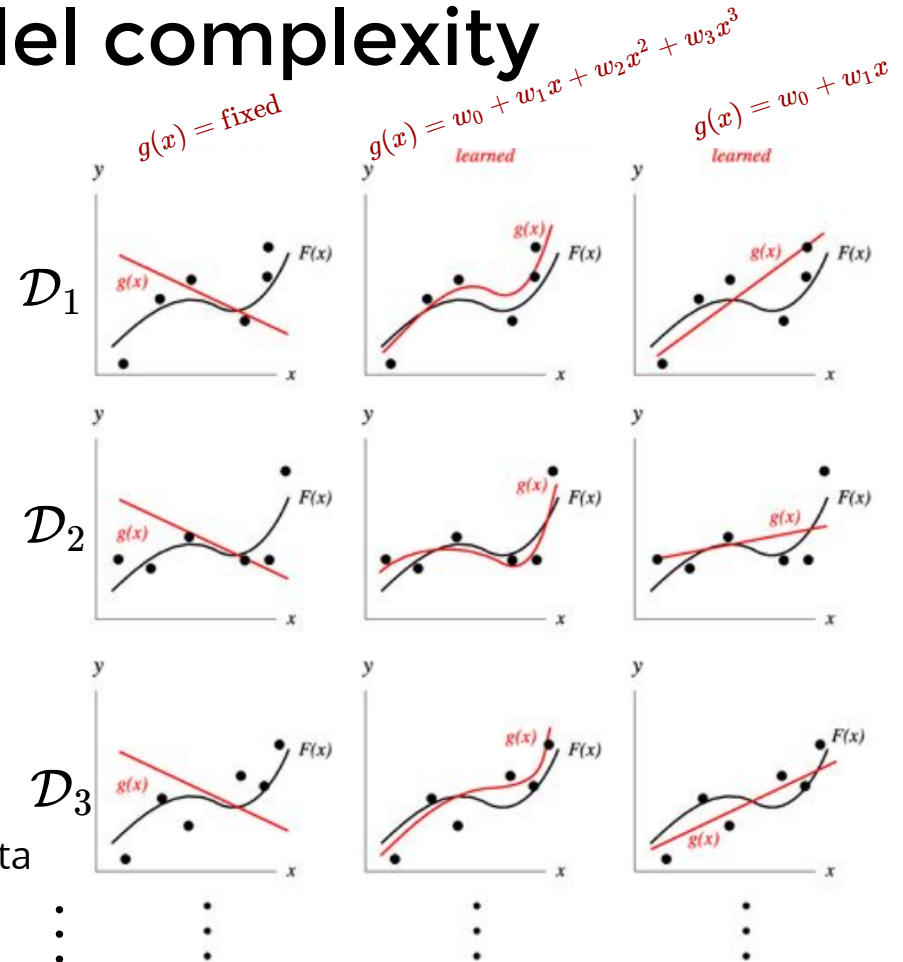
$F(x)$  the best possible model

## higher variance

the complex model varies more with the dataset  
it may not generalize well for this reason

## higher bias

the simple model is biased to a particular type of data  
it underfits, but it has a low variance



# Bias-variance decomposition: Setup

decompose the generalization error to see the effect of bias and variance (for L2 loss)

assume a true distribution  $p(x, y)$

best prediction given L2 loss  $f(x) = \mathbb{E}_p[y|x]$

assume that a dataset  $\mathcal{D} = \{(x^{(n)}, y^{(n)})\}_n$  is sampled from  $p(x, y)$

let  $\hat{f}_{\mathcal{D}}$  be our model based on the dataset

what we care about is the **generalization error** (aka expected loss, expected risk)

$$\mathbb{E}[(\hat{f}_{\mathcal{D}}(x) - y)^2]$$

all blue items are random variables

# Bias-variance decomposition

what we care about is the generalization error

$$\mathbb{E}[(\hat{f}_{\mathcal{D}}(x) - y)^2] = \mathbb{E}[(\hat{f}_{\mathcal{D}}(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)] - y + \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)])^2]$$

$\hat{f}_{\mathcal{D}}(x)$        $y$   
|                      |  
 $\hat{f}_{\mathcal{D}}(x)$        $f(x) + \epsilon$

$\hat{f}_{\mathcal{D}}(x) + \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)] - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)]$  add and subtract a term

above simplifies to the following (the remaining terms are going to be zero)

$$= \underbrace{\mathbb{E}[(\hat{f}_{\mathcal{D}}(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)])^2]}_{\text{variance}} + \underbrace{\mathbb{E}[(f(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)])^2]}_{\text{bias}^2} + \underbrace{\mathbb{E}[\epsilon^2]}_{\text{unavoidable noise error}}$$

# Bias-variance decomposition

the expected loss is decomposed to:

$$= \mathbb{E}[(\hat{f}_{\mathcal{D}}(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)])^2] + \mathbb{E}[(f(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)])^2] + \mathbb{E}[\epsilon^2]$$

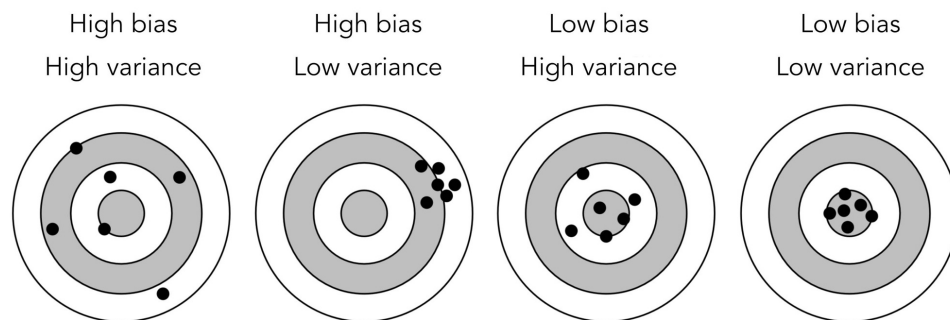
**variance:** how change of dataset affects the prediction

**bias:** how average over all datasets differs from the regression function

**noise error:** the error even if we used the true model  $f(x)$

different models vary in their trade off between error due to bias and variance

- simple models: often more biased
- complex models: often have more variance





# Bias vs. variance

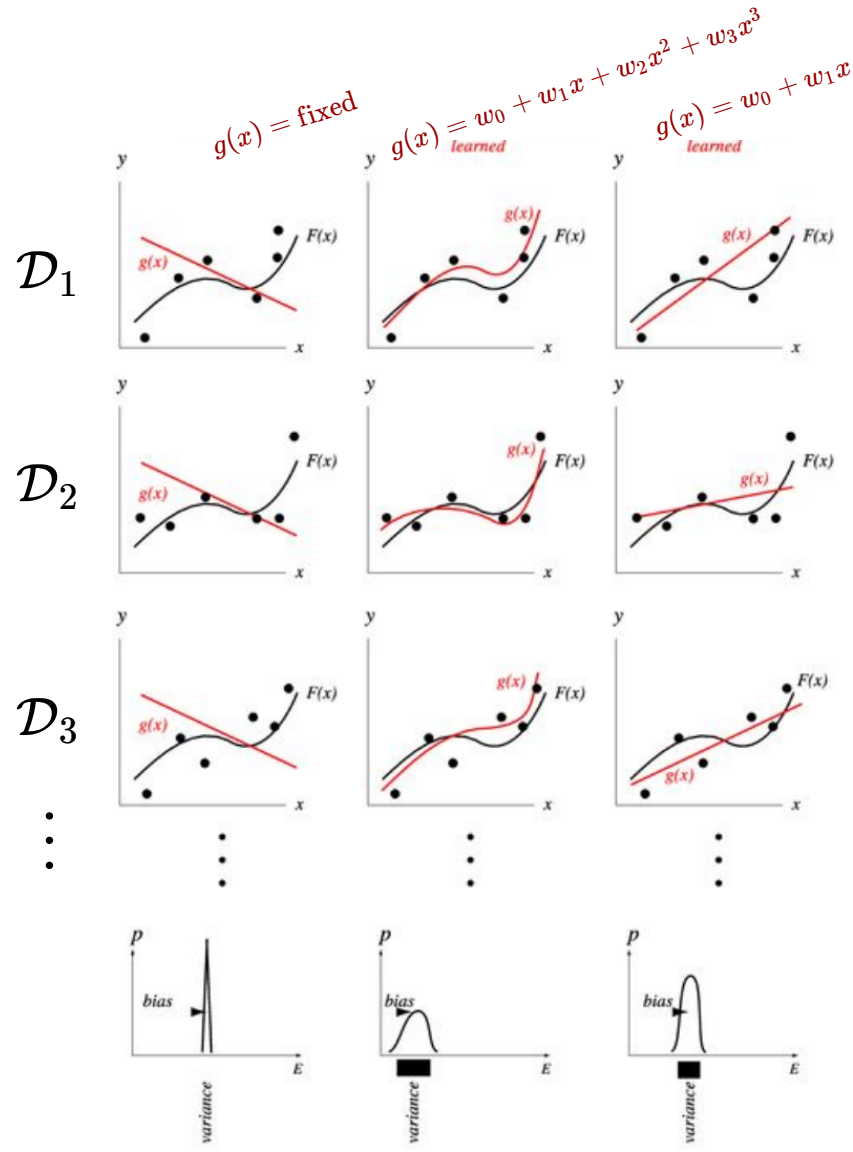
example

distribution of error (cost) due to randomness of dataset

we care about the expected error

bias causes a high error for all choices of dataset

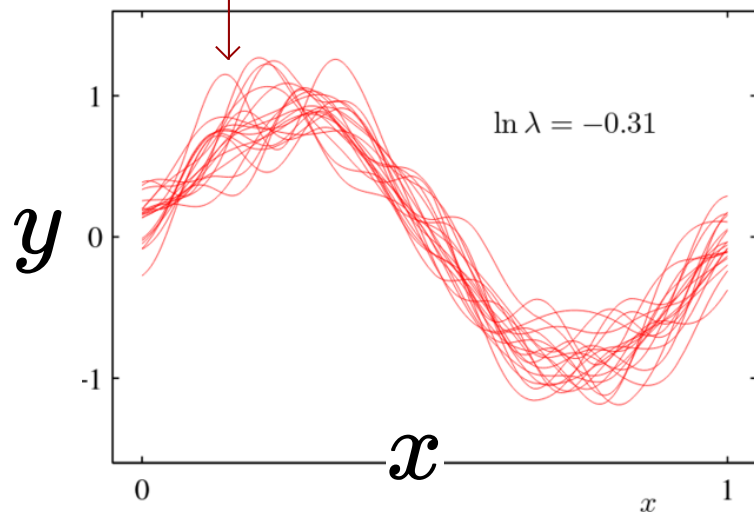
higher variance also increases the expected error



# Example: bias vs. variance

models for different datasets  $\hat{f}_{\mathcal{D}}$   
using Gaussian bases

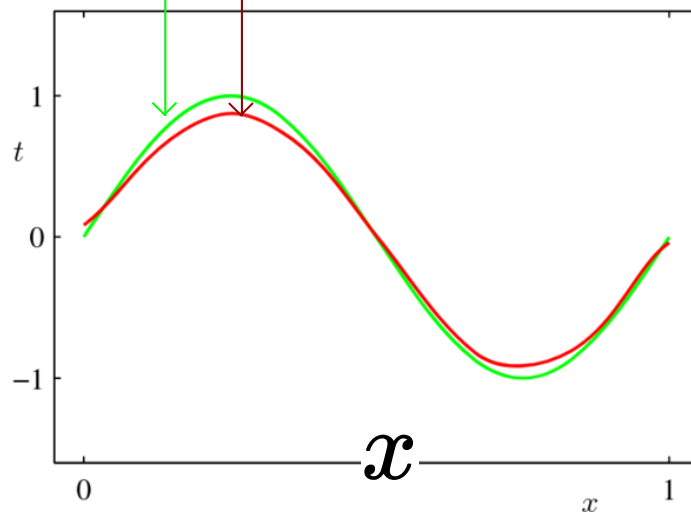
random datasets of size  $N=25$  instances are not shown



**variance** is the average difference (in squared L2 norm) between these curves and their average

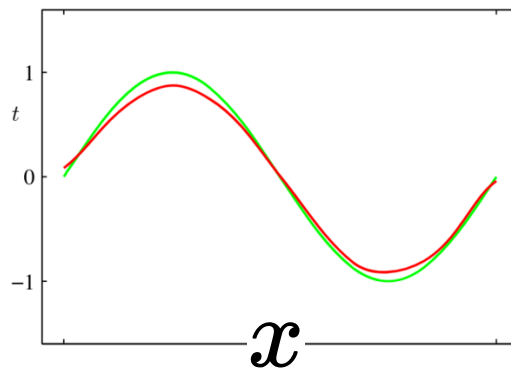
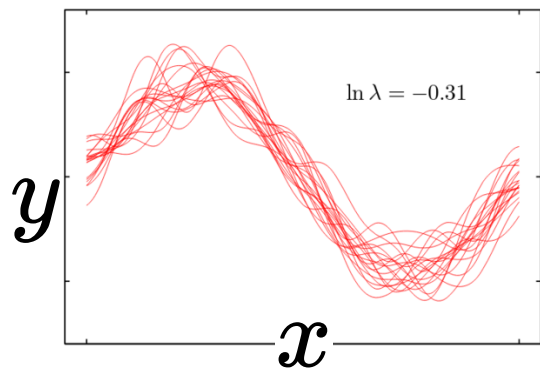
true model  $f$

their average  $\mathbb{E}[\hat{f}_{\mathcal{D}}]$



**bias** is the difference (in L2 norm) between two curves

# Example: bias vs. variance

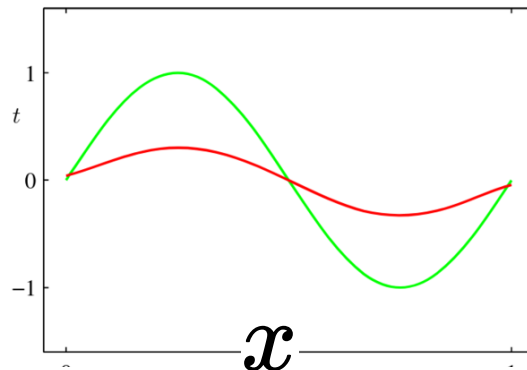
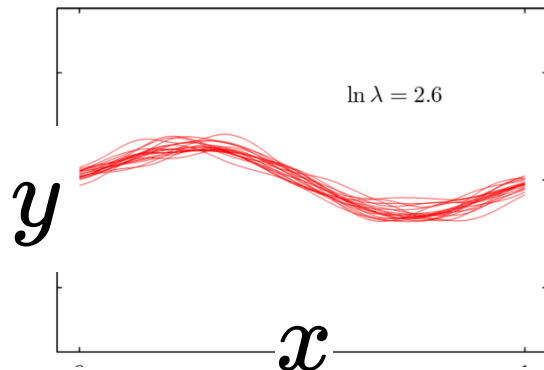


side note

the average fit is very good, despite high variance

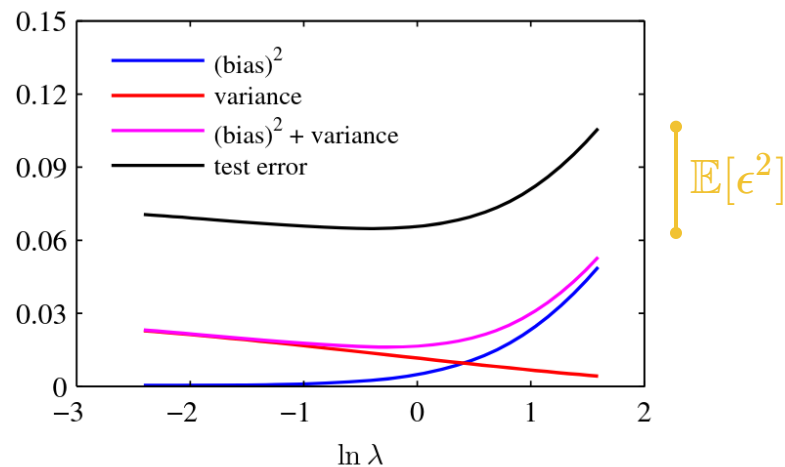
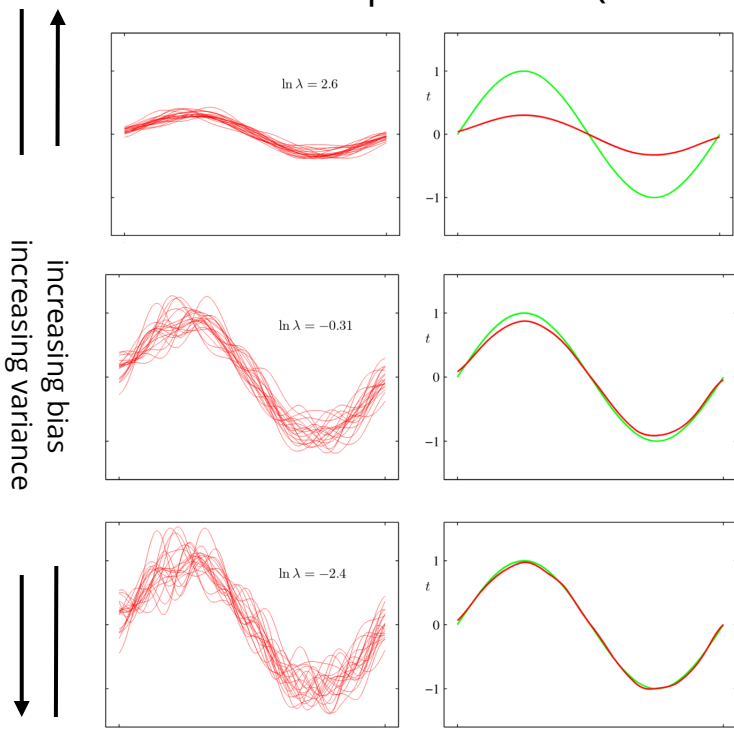
**model averaging:** uses "average" prediction of expressive models to prevent overfitting

using larger regularization penalty: **higher bias - lower variance**



# Example: bias vs. variance

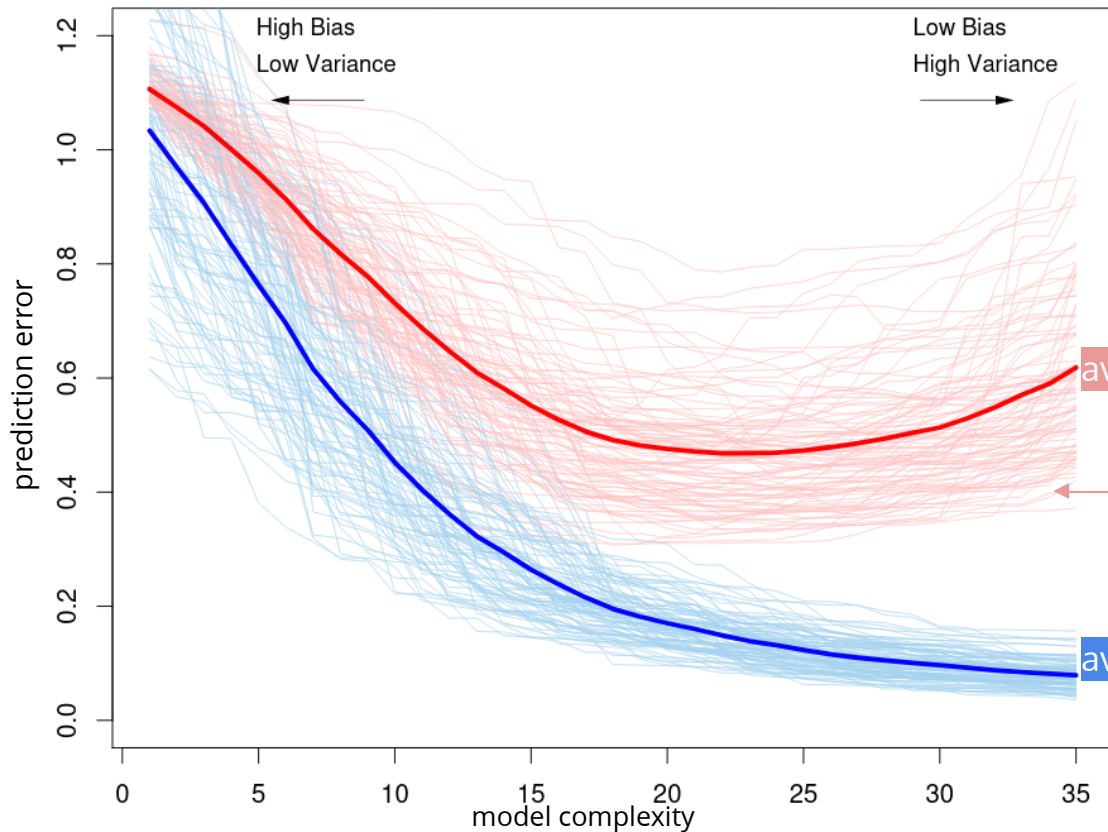
the lowest expected loss (test error) is somewhere between the two extremes



in practice, how to decide which model to use?

# Effect on training and test error

high bias in  
simplistic models  
means that **training  
error can be high**



high variance in more  
complex models means  
that **test and training error  
can be very different**

average test error

error for random dataset  $\mathcal{D}$

average training error

# Model selection

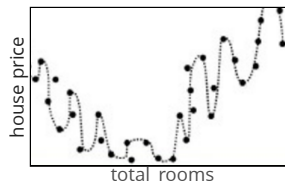
many ML algorithms have hyper-parameters

(e.g., the number of nonlinear basis to use)

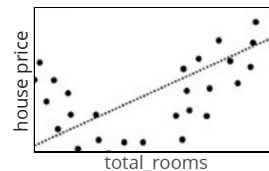
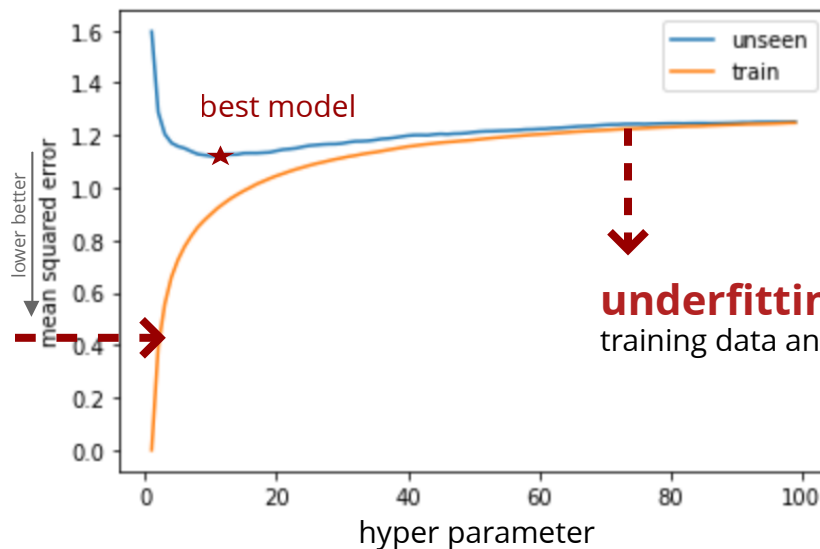
how should we select the best hyper-parameter?

example

performance of a regression model on *California Housing Dataset*



**overfitting** to the training data  
bad performance on unseen data



**underfitting** the model can more closely fit the training data and still get good test error

# Model selection

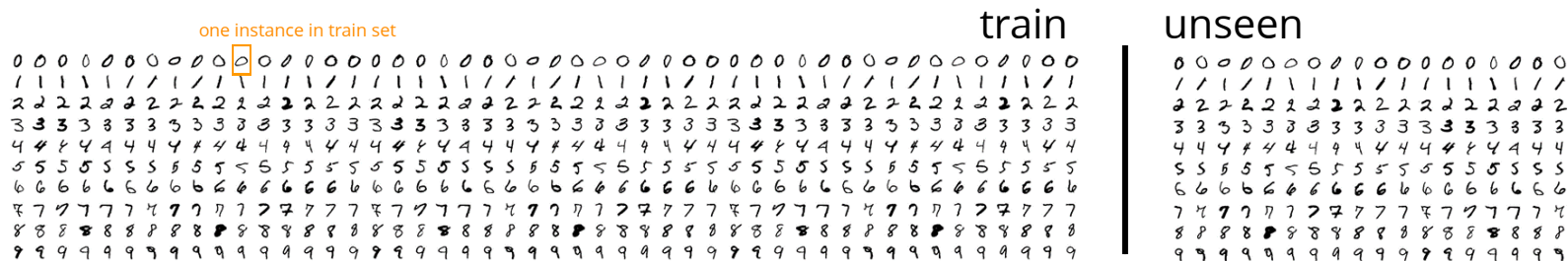
what if unseen data is completely different from training data?

no point in learning!

**assumption:** training data points are samples from an unknown distribution  
*independent identically distributed (IID)*

$$x^{(n)}, y^{(n)} \sim p(x, y)$$

unseen data comes from the same distribution.



# Loss, cost and generalization

assume we have a **model**  $f : x \mapsto y$  for example  $f : \boxed{3} \mapsto 3$

and we have a **loss function** that measures the error in our prediction  $\ell : y, \hat{y} \rightarrow \mathbb{R}$

$$\text{for example } \begin{cases} \ell(y, \hat{y}) = (y - \hat{y})^2 & \text{for regression} \\ \ell(y, \hat{y}) = \mathbb{I}(y \neq \hat{y}) & \text{for classification} \end{cases}$$

we train our models to minimize **the cost function**:

$$J = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{x, y \in \mathcal{D}_{\text{train}}} \ell(y, f(x))$$

We can drop this, why?

how to estimate this?

what we really care about is the **generalization error**:  $\mathbb{E}_{x, y \sim p} \ell(y, f(x))$

we can not measure this, why?

we can set aside part of the given data and use it to estimate generalization error





# Cross validation

how to get a better estimate of generalization error?

increase the size of the validation set? *this reduces the training set*

training										validation										test									
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9

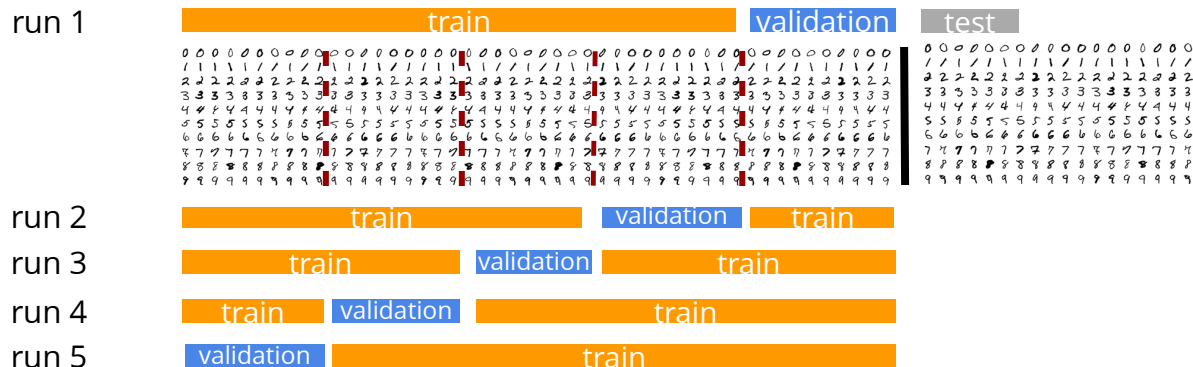
**Cross-validation** helps us in getting better estimates + uncertainty measure

- divide the (training + validation) data into  $L$  parts
- use one part for validation and  $L-1$  for training

$L = 5$										test									
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9

# Cross validation

- divide the (training + validation) data into L parts
- use one part for validation and L-1 for training



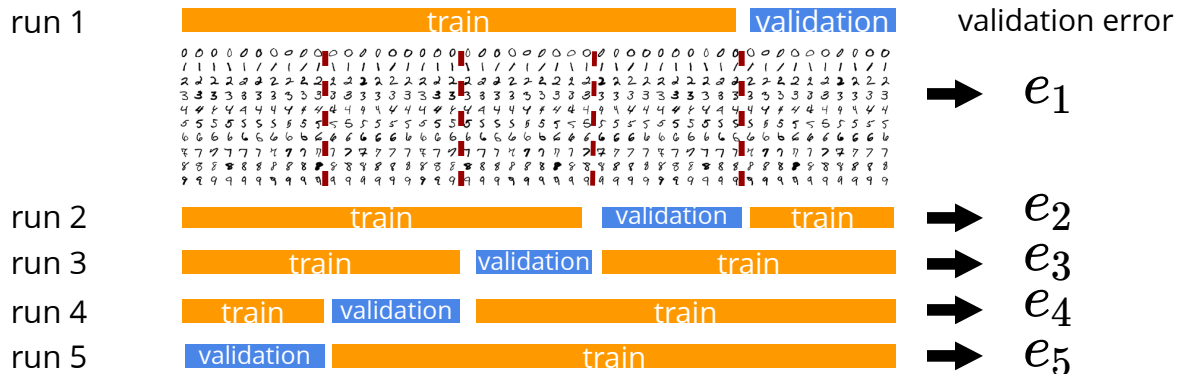
- use the **average** validation error and its variance (uncertainty) to pick the best model
- report the test error for the final model

this is called **L-fold** cross-validation

in **leave-one-out** cross-validation  $L=N$  (only one instance is used for validation)

# Cross validation

- divide the (training + validation) data into L parts
- use one part for validation and L-1 for training



- use the **average** validation error and its variance (uncertainty) to pick the best model

$$\bar{e} = \frac{1}{5} \sum_{i=1}^5 e_i$$

- report the test error for the final model



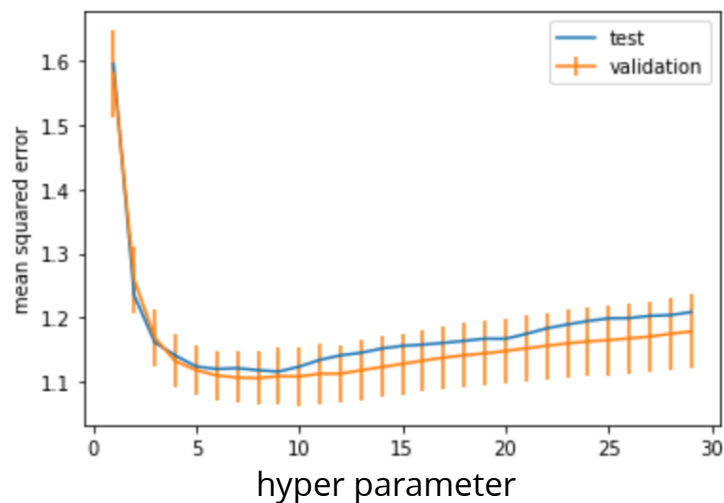
$\rightarrow e_t$

this is called **L-fold** cross-validation

in **leave-one-out** cross-validation L=N (only one instance is used for validation)

# Cross validation

**example** the plot of the mean and standard deviation in 10 fold cross-validation



test error is plotted only to show its agreement with the validation error; in practice we don't look at the test set for hyper-parameter tuning

**a rule of thumb:** pick the simplest model within one std of the model with lowest validation error

# Performance metrics for classification

Not all errors are the same

In particular in classification, we have different **types of mistakes**

**false positive (type I)** and **false negative (type II)**

**example:**

patient does not have disease but received positive diagnostic (**Type I error**)

patient has disease but it was not detected (**Type II error**)

a message that is not spam is assigned to the spam folder (**Type I error**)

a message that is spam appears in the regular folder (**Type II error**)

# Performance metrics for classification

binary classification results:

*FP* false positive (type I)

*FN* false negative (type II)

*TP* true positive

*TN* true negative

confusion matrix	Truth		$\Sigma$
	TP	FP	RP
Result	FN	TN	RN
$\Sigma$	P	N	

**marginals**

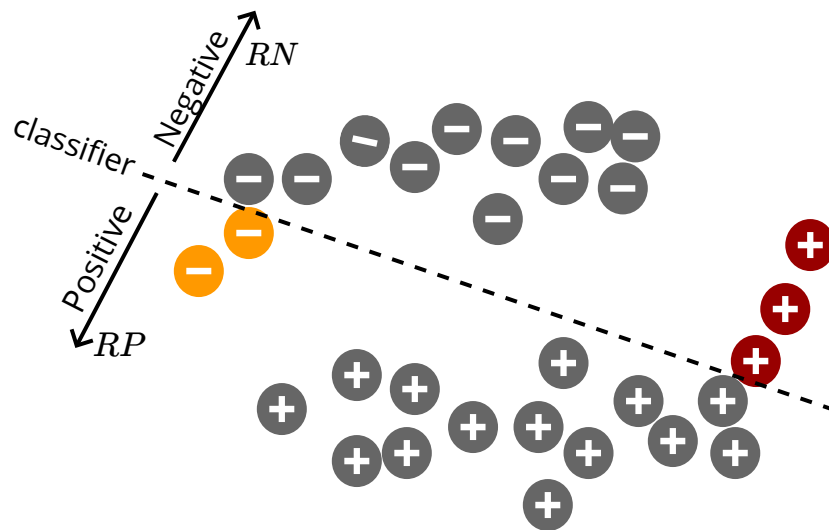
$$RP = TP + FP$$

$$RN = TN + FN$$

$$P = TP + FN$$

$$N = TN + FP$$

$TN + TP + FN + FP = ?$



example:	Truth		$\Sigma$
Result	14	2	16
	3	11	14
$\Sigma$	17	13	

# Performance metrics for classification

confusion matrix	Truth		$\Sigma$
	TP	FP	RP
	FN	TN	RN
Result			
$\Sigma$	P	N	

example:	Truth		$\Sigma$
	14	2	16
	3	11	14
Result			
$\Sigma$	17	13	

$$Precision = \frac{TP}{RP} = \frac{14}{16}$$

$$Recall = \frac{TP}{P} = \frac{14}{17}$$

$$\frac{1}{F_{\beta} score} = \frac{1}{1+\beta^2} \frac{1}{Precision} + \frac{\beta^2}{1+\beta^2} \frac{1}{Recall}$$

$$Accuracy = \frac{TP+TN}{P+N}$$

$$Precision = \frac{TP}{RP}$$

$$Recall = \frac{TP}{P} \quad \text{sensitivity}$$

$$F_1 score = 2 \frac{Precision \times Recall}{Precision + Recall} \quad \{\text{Harmonic mean}\}$$

$$F_{\beta} score = (1 + \beta^2) \frac{Precision \times Recall}{\beta^2 Precision + Recall}$$

recall is  $\beta$  times more important compared to precision

$$Miss rate = \frac{FN}{P}$$

$$Fallout = \frac{FP}{N} \quad \text{false positive rate}$$

$$False discovery rate = \frac{FP}{RP}$$

$$Selectivity = \frac{TN}{N} \quad \text{specificity}$$

$$False omission rate = \frac{FN}{RN}$$

$$Negative predictive value = \frac{TN}{RN}$$

less common



# Trade-off between precision and recall

How many false positives do we tolerate?

How important are false negatives?

e.g. spam in inbox v.s. negative test for cancer test

We can often control the trade-off between type I & type II error

e.g. by changing the threshold of  $p(y = 1|x)$  if we produce class score (probability)

**goal:** evaluate class scores/probabilities (independent of choice of threshold)

Receiver Operating Characteristic **ROC curve**, a function of threshold  $t$

**TPR(t)** = TP(t)/P (**recall**, sensitivity at  $t$ , hit rate)

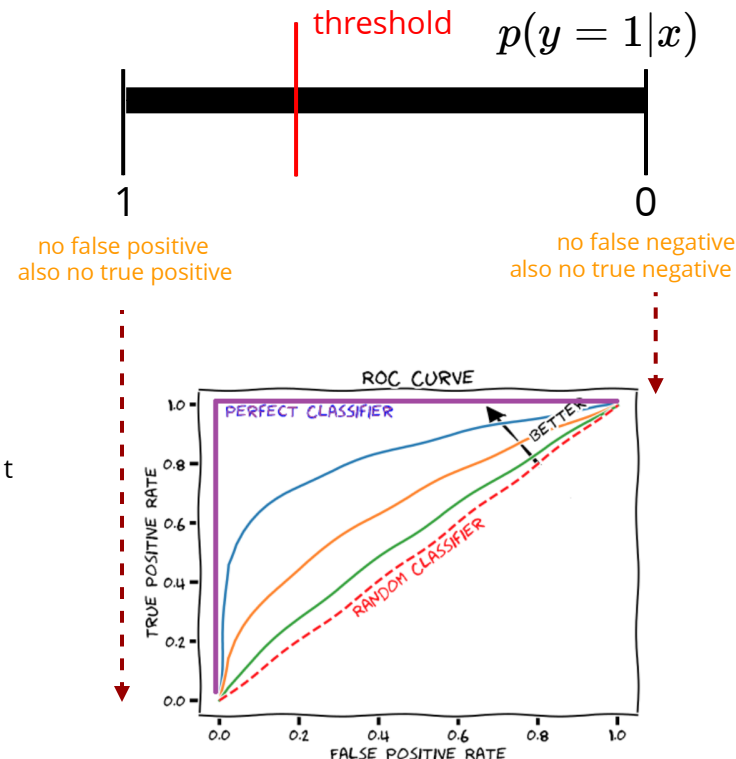
**FPR(t)** = FP(t)/N (**fallout**, false alarm at  $t$ , type I error rate)

Area Under the Curve (**AUC**) is used as a threshold independent measure of quality of the classifier

$$AUC = \sum_t TPR(t)(FPR(t) - FPR(t-1)) \text{ , box-rule approximation}$$

what is the maximum value for AUC? what is AUC of a random classifier?

Most ML algorithm produces class score or probability



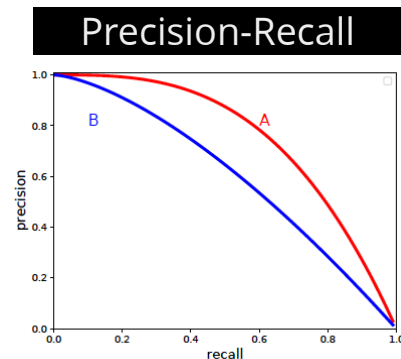
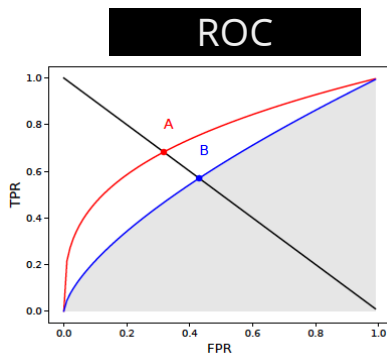
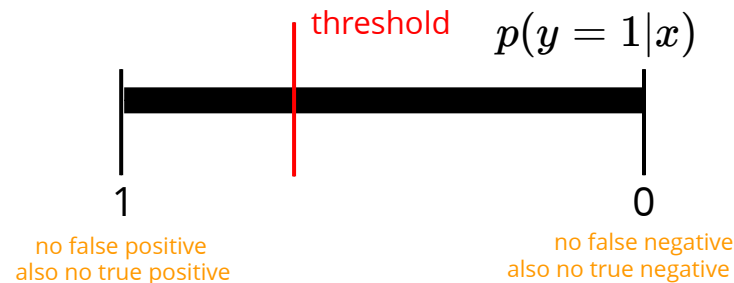
# Precision-recall curves

Similar to ROC curve but more helpful in some situation

- when size of negative set is also a model parameter, e.g. in information retrieval
- when there is class imbalance, e.g. in fraud detection when  $P \gg N$ ), since *ROC curves are insensitive to class imbalance*

Instead we curve Precision vs Recall for different thresholds

Most ML algorithm produces class score or probability



# Confusion Matrix for multiclass classification

A  $C \times C$  table that shows how many samples of each class are classified as belonging to another class

$$M_{rc} = N\{\hat{y} = r, y = c\}$$

**CIFAR-10 Confusion Matrix**

airplane	923	4	21	8	4	1	5	5	23	6
automobile	5	972	2					1	5	15
bird	26	2	892	30	13	8	17	5	4	3
cat	12	4	32	826	24	48	30	12	5	7
deer	5	1	28	24	898	13	14	14	2	1
dog	7	2	28	111	18	801	13	17		3
frog	5		16	27	3	4	943	1	1	
horse	9	1	14	13	22	17	3	915	2	4
ship	37	10	4	4		1	2	1	931	10
truck	20	39	3	3			2	1	9	923

per class recalls:

88.0%	93.9%	85.8%	79.0%	91.4%	89.7%	91.6%	94.1%	94.8%	95.0%
12.0%	6.1%	14.2%	21.0%	8.6%	10.3%	8.4%	5.9%	5.2%	5.0%

per class precisions:

92.3%	7.7%
97.2%	2.8%
89.2%	10.8%
82.6%	17.4%
89.8%	10.2%
80.1%	19.9%
94.3%	5.7%
91.5%	8.5%
93.1%	6.9%
92.3%	7.7%

True Class

Predicted Class

airplane automobile bird cat deer dog frog horse ship truck

sample images from Cifar-10 dataset

airplane

automobile

bird

cat

deer

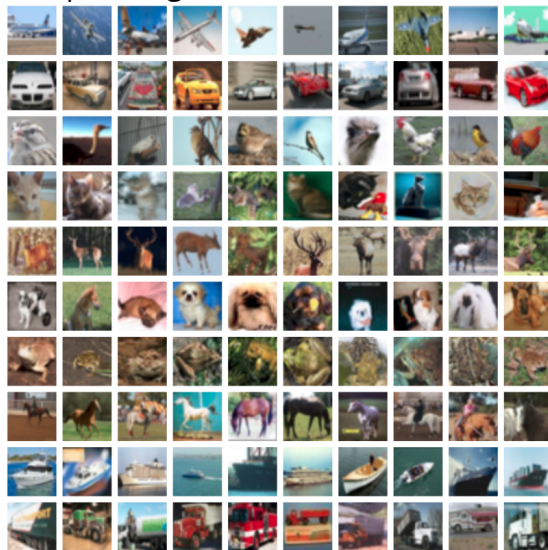
dog

frog

horse

ship

truck



classifier's accuracy is the sum of diagonal divided by the sum-total of the matrix, you can also report the average of the F<sub>1</sub> scores per class (macro), or weight the average by class sizes

when evaluating a classifier it is useful to look at the confusion matrix

# Bias and Fairness Challenge

The model learns from the distribution of the input data

{train, validation, test are still sampled based on some process}

the demographic and phenotypic composition of training and benchmark datasets are important

## Growing use, growing concerns

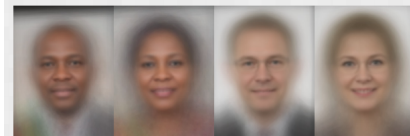
- Amazon's hiring algorithm decides not to invite women to interview, [read it here](#)
- Google's online ad algorithm decides to show high-income jobs to men much more often than to women, [read about it here](#)
- A machine learning algorithm denies you credit based on race or gender, [read it here](#)
- Health care algorithm offers less care to black patients, [read it here](#), and [here](#)
- Florida risk score algorithm used in courts assign higher risk to black defendants, [read it here](#)

How can we factor these in the evaluation of models?

Many recent works, for example see this book on fairness & ML, [here](#), read this article on [bias detectives](#), or [this course](#) on data ethics

## Face-recognition software is perfect – if you're a white man

Gender Classifier	Darker Male	Darker Female	Lighter Male	Lighter Female	Largest Gap
Microsoft	94.0%	79.2%	100%	98.3%	20.8%
FACE++	99.3%	65.5%	99.2%	94.0%	33.8%
IBM	88.0%	65.3%	99.7%	92.9%	34.4%



nature

UPDATE 26 OCTOBER 2015

### Millions of black people affected by racial bias in health-care algorithms

Study reveals rampant racism in decision-making software used by US hospitals – and highlights ways to correct it.

Heidi Ledford



PDF version

RELATED ARTICLES  
A failure way forward for AI in health care

Bias detectives: the researchers striving to make algorithms fair

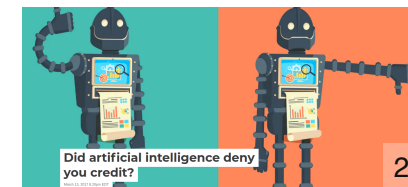
Can we open the black box of AI?

Black people with complex medical needs were less likely than equally ill white people to be referred to programmes that provide more personalized care. Credit: Ed Kaush/VA/Reuters/Science

SUBJECTS

Computer science Health care  
Policy Society

An algorithm widely used in US hospitals to allocate health care to patients has been systematically discriminating against



# Inductive bias

learning algorithms make implicit assumptions **learning or inductive bias**

e.g., we are often biased towards **simplest explanations** of our data

**Occam's razor** between two models (explanations) we should prefer the simpler one

## example

both of the following models perfectly fit the data

x1	x2	x3	f
1	0	1	0
1	1	0	1
1	1	1	1

$\hat{f}(x) = x_2$  this one is simpler

$\hat{f}(x) = x_1 \wedge x_2$



why does it make sense for learning algorithms to be biased?

- the world is not random
- there are regularities, and induction is possible (why do you think the sun will rise in the east tomorrow morning?)

what are some of the inductive biases in using linear regression?



# Curse of dimensionality

learning in **high dimensions** can be difficult since the volume of space grows exponentially fast with the dimension

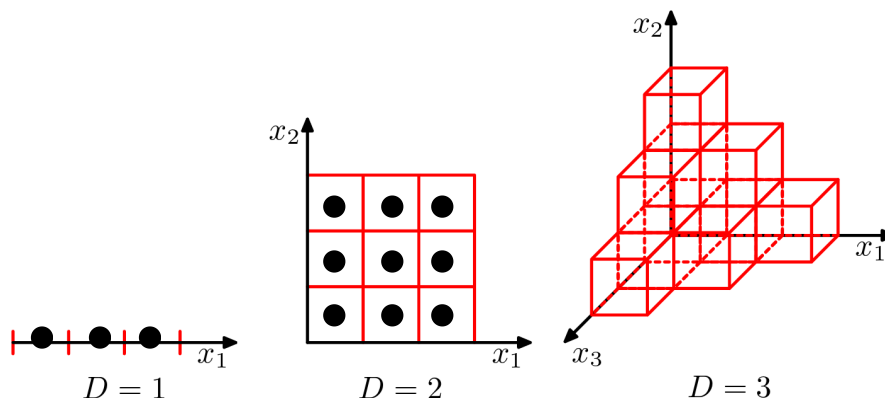
example:

suppose our data is uniformly distributed in some range, say  $x \in [0, 3]^D$

predict the label by counting labels in the same unit of the grid

to have at least one example per unit, we need  $3^D$  training examples

for  $D=180$  we need more training examples than the number of particles in the universe

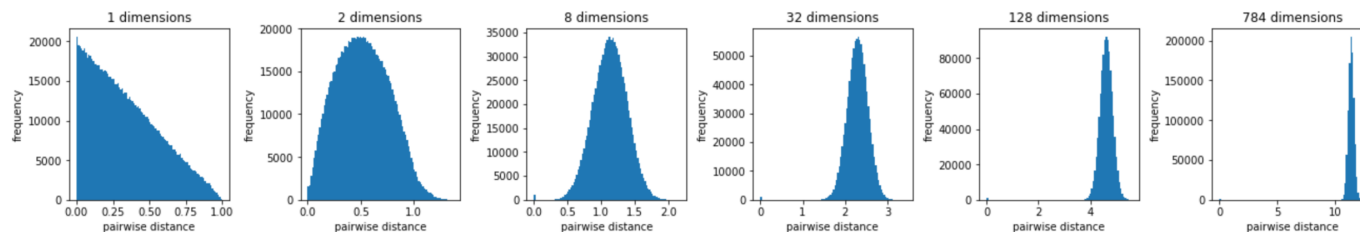




# Curse of dimensionality

in high dimensions most points have similar distances!

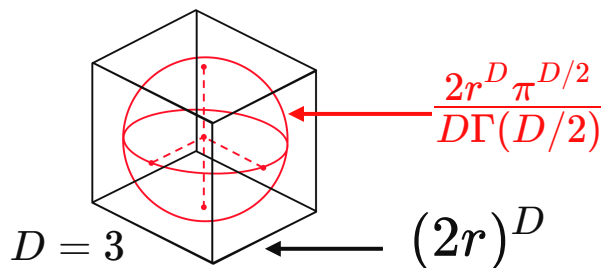
histogram of pairwise distance of 1000 points with random features of D dimensions



as we increase dimension,  
distances become "similar"!

**Q.** why are most distances similar?

**A.** in high dimensions most of the volume is close to the corners!



$$\lim_{D \rightarrow \infty} \frac{\text{volum}(\circ)}{\text{volum}(\square)} = 0$$

a "conceptual" visualization of the same idea  
# corners and the mass in the corners grow quickly with D

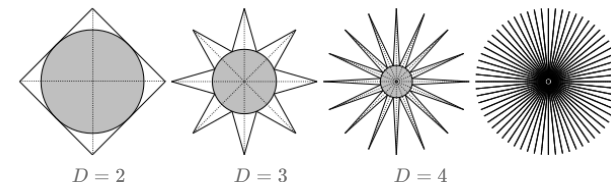
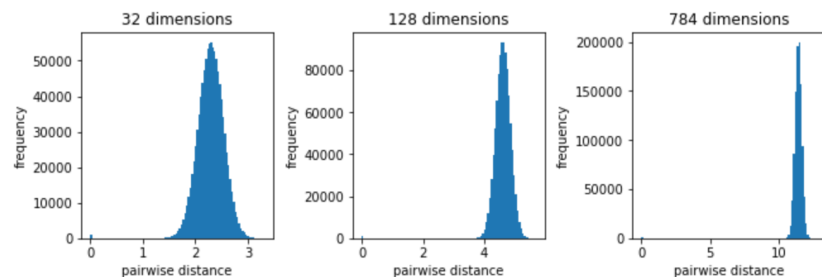


image: Zaki's book on Data Mining and Analysis

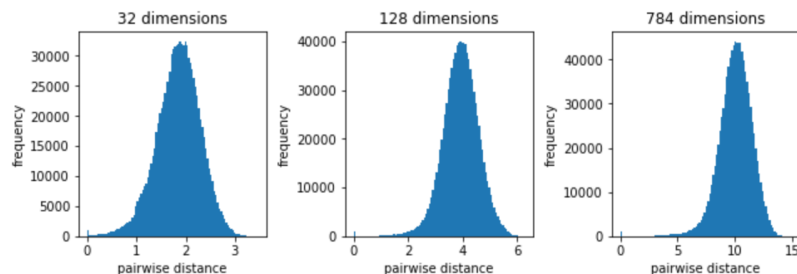
# Real-word vs. randomly generated data

how come ML methods work for image data ( $D$ =number of pixels)?

pairwise distance for random data



pairwise distance for  $D$  pixels of MNIST digits



0 1 2 3 4 5 6 7 8  
0 1 2 3 4 5 6 7 8  
0 1 2 3 4 5 6 7 8

the statistics do not match that of random high-dimensional data!

in fact KNN works well for image classification

	Test Error Rate (%)
Linear classifier (1-layer NN)	12.0
K-nearest-neighbors, Euclidean	5.0
K-nearest-neighbors, Euclidean, deskewed	2.4
K-NN, Tangent Distance, 16x16	1.1
K-NN, shape context matching	0.67
1000 RBF + linear classifier	3.6
SVM deg 4 polynomial	1.1
2-layer NN, 300 hidden units	4.7
2-layer NN, 300 HU, [deskewing]	1.6
LeNet-5, [distortions]	0.8
Boosted LeNet-4, [distortions]	0.7



# Manifold hypothesis

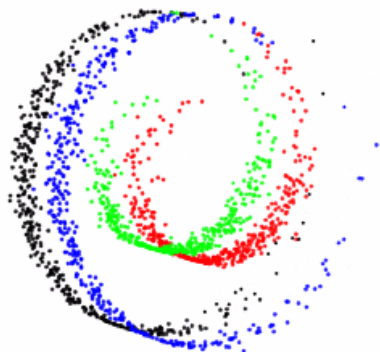
real-world data is often far from uniformly random

**manifold hypothesis:** real data lies close to the surface of a *manifold*

example

data dimension:  $D = 3$

manifold dimension:  $\hat{D} = 2$



example

data dimension:  $D = \text{number of pixels (64x64)}$

manifold dimension:  $\hat{D} = 2$

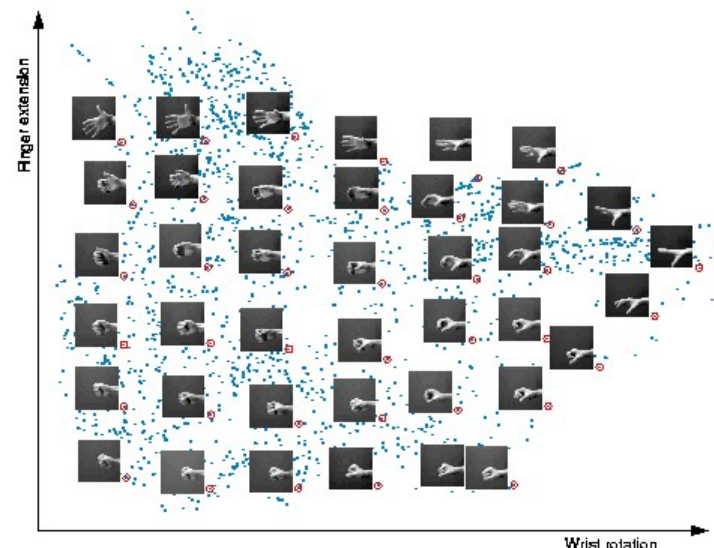


image from [here](#)

# No free lunch

consider the binary classification task:

suppose this is our dataset →

x1	x2	x3	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



read more [here](#)

there are  $2^4 = 16$  binary functions that perfectly fit our dataset

our **learning algorithm** can produce one of these as our classifier  $\hat{f} : \{0, 1\}^3 \rightarrow \{0, 1\}$

the same algorithm cannot perform well for all possible class of problems (f)

no free lunch

each ML algorithm is biased to perform well on some class of problems

there is no single algorithm that performs well on all class of problems

# Summary

- complex models can have very different training and test error (*generalization gap*)
- regularization bounds this gap by penalizing model complexity
- bias-variance trade off:
  - formalizes the relation between
    - training error (bias)
    - complexity (variance) and
    - and the test error (bias + variance)
  - not so elegant beyond L2 loss
- what we care about is the **generalization** of ML algorithms
  - **overfitting**: good performance on the training set doesn't mean the same for the test set
  - **underfitting**: we don't even have a good performance on the training set
- estimated using a **validation set** or better, we could use **cross-validation**