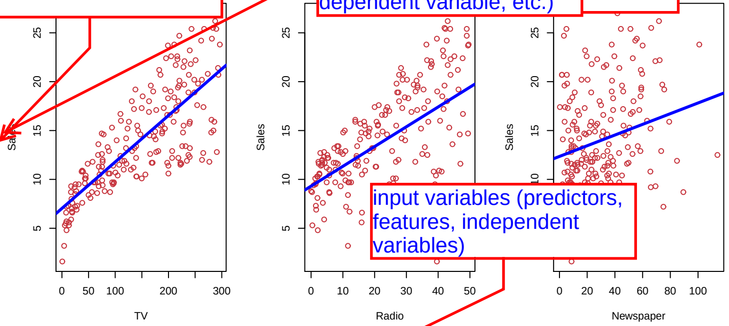
What is Sta

context: statistical consultant, hired to give advice on improving the

1000's of units

output variables (response, dependent variable, etc.)



input variables (predictors, features, independent variables)

Shown are `Sales` vs `TV`, `Radio` and `Newspaper`, with a blue linear-regression line fit separately to each.
Can we predict `Sales` using these three?
Perhaps we can do better using a model

$$\texttt{Sales} \approx f(\texttt{TV}, \texttt{Radio}, \texttt{Newspaper})$$

random variable

fixed variables (not random)

H *se* or *target* that we wish to predict. We generically refer to the response as Y.
TV is a *feature*, or *input*, or *predictor*; we name it X_1.
Likewise name Radio as X_2, and so on.
We can refer to the *input vector* collectively as

fixed (not random)

$$X = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix}$$

"data generating model": the model that we presume our data comes from

Y(x): the random response associated with fixed inputs x

f(x): a systematic function of inputs x (not random)

Now we write our model as

$$Y = f(X) + \epsilon$$

random variable modeling error (capture things we cannot control).
- assume E(\epsilon) (i.e., its expectation) = 0. zero-mean error term

where ϵ captures measurement errors and

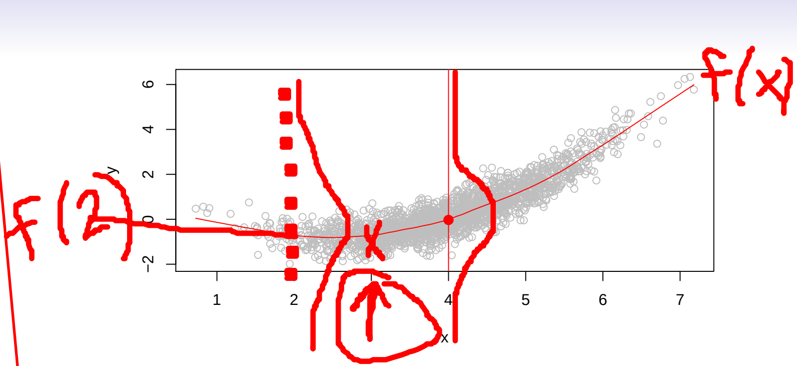
Two types of predictions:
- point prediction
- probabilistic predictions (predicting a distribution)

hat is $f(X)$

Two properties we ,may want to learn from a model:
(i) prediction
(ii) inference

- With a good f we can make predictions of Y at new points $X = x$.
- We can understand which components $X = (X_1, X_2, \ldots, X_p)$ are important i which are irrelevant. e.g. Seniority a Education have a big impact on Inco Status typically does not.
- Depending on the complexity of f, we understand how each component X_j c

inference: understanding the domain problem better from data
- which predictors are associated / influential for the response?
- what is the relationship between a predictor and a response?
- is this relationship linear? or is it more complex (nonlinear)?



Is there an ideal $f(X)$? In particular, what is a good value for $f(X)$ at any selected value of X, say $X = 4$? There can be many Y values at $X = 4$. A good value is

Let's start by assuming we know the generative model (with certainty):

Y(x) = f(x) + \epsilon

- we know what the function f is precisely
- we also know what the random variable \epsilon is (e.g., its pdf or pmf)

$E(Y$ $= 4.$

This ction.

The regression function $f(x)$

- Is also defined for vector X; e.g.
  $f(x) = f(x_1, x_2, x_3) = E(Y|X_1 = x_1, X_2 = x_2, X_3 = x_3)$

The regression

mean-squared prediction error (MSPE):
- one choice of loss function

- Is also defined for vector X; e.g. $f(x) = f(x_1, x_2, x_3) = E(Y|X_1 = x_1, X_2 = x_2, X_3 = x_3)$
- Is the *ideal* or *optimal* predictor of Y with regard to mean-squared prediction error: $f(x) = E(Y|X = x)$ is the function that minimizes $E[(Y - g(X))^2|X = x]$ over all functions g at all points $X = x$.

The regression function $f(x)$

- Is also defined for vector X; e.g. $f(x) = f(x_1, x_2, x_3) = E(Y|X_1 = x_1, X_2 = x_2, X_3 = x_3)$
- Is the *ideal* or *optimal* predictor of Y with regard to mean-squared prediction error: $f(x) = E(Y|X = x)$ is the function that minimizes $E[(Y - g(X))^2|X = x]$ over all functions g at all points $X = x$.
- $\epsilon = Y - f(x)$ is the *irreducible* error — i.e. even if we knew $f(x)$, we would still make errors in prediction, since at each $X = x$ there is typically a distribution of possible Y values.

ession function $f(x)$

what happens to reducible error as we collect more and more data?
- model misspecification: the assumed model for estimating f(x) is not the true regression model (e.g., last example, if we used a linear regression model to estimate f(x))
- when there is model misspecification, then reducible error will decrease as we collect more data, but NOT go to zero (systematic bias)
- assuming no model misspecification, reducible error -> 0 as we collect more data

ector X; e.g.

$= E(Y|X_1 = x_1, X_2 = x_2, X_3 = x_3)$

al predictor of Y with regard to

ction error: $f(x) = E(Y|X = x)$ is the

izes $E[(Y - g(X))^2|X = x]$ over all

ints $X = x$.

irreducible error —

cannot be reduced by our choice of predictor g

$f(x)$, we would still make errors in prediction, since at each $X = x$ there is typically a distribution of possible Y values.

- For any estimate $\hat{f}(x)$ of $f(x)$, we have

$$E[(Y - \hat{f}(X))^2|X = x] = \underbrace{[f(x) - \hat{f}(x)]^2}_{\textit{Reducible}} + \underbrace{\mathrm{Var}(\epsilon)}_{\textit{Irreducible}}$$

See board for derivation

How to estimate f

One strategy: K-nearest neighbor prediction at x=4
- take the closest K (e.g., 10) data points to x=4
- set our predictor of f(4) (denote as \hat{f}(4)) as simply the average of the K outputs from the data

e have few if any data points with $X = 4$

ot compute $E(Y|X = x)$!

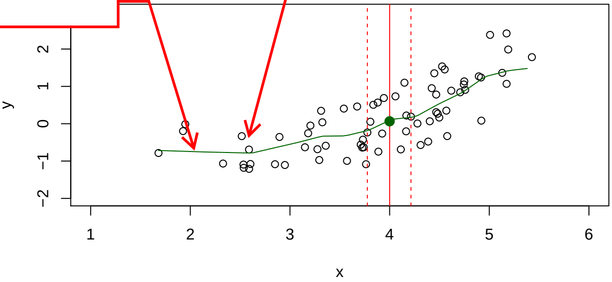
- Relax the definition and let

$$\hat{f}(x) = \text{Ave}(Y|X \in \mathcal{N}(x))$$

where $\mathcal{N}(x)$ is some *neighbor*

K-nearest neighbors (KNNs) are a popular class of non-parametric model for estimating f(x)

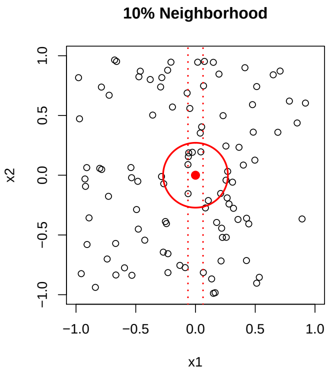
green line = prediction of f (x)

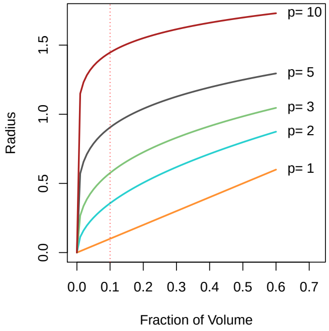


- Nearest neighbor averaging can be pretty good for small p — i.e. $p \leq 4$ and large-ish N.
- We will discuss smoother versions, such as kernel and spline smoothing later in the course.

- Nearest neighbor averaging can be pretty good for small p — i.e. $p \leq 4$ and large-ish N.
- We will discuss smoother versions, such as kernel and spline smoothing later in the course.
- Nearest neighbor methods can be *lousy* when p is large. Reason: the *curse of dimensionality*. Nearest neighbors tend to be far away in high dimensions.
    - We need to get a reasonable fraction of the N values of y_i to average to bring the variance down—e.g. 10%.
    - A 10% neighborhood in high dimensions need no longer be local, so we lose the spirit of estimating $E(Y|X = x)$ by local averaging.

The curse of dimensionality





Parametric model:
- assume a model on f that depends a finite number of parameters (\beta)
- estimate the parameters \beta to obtain a good predictor \hat{f} for the true regression function f(x)

etric and structured models

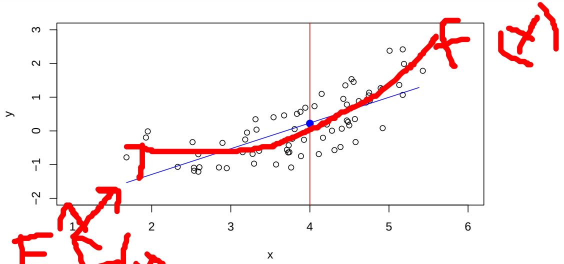
s an important example of a parametric model:

$$f_L(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2$$

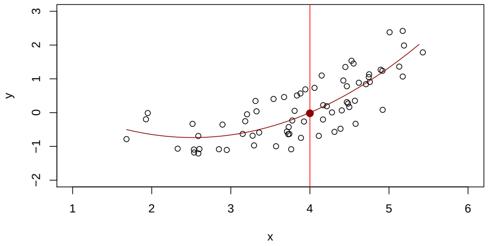
i.e., the true regression function f(x) is almost never a linear function of x

- A linear model is specified in terms of $p + 1$ parameters $\beta_0, \beta_1, \ldots, \beta_p$.
- We estimate the parameters by fitting the model to training data.
- Although it is *almost never correct*, a linear model often serves as a good and interpretable approximation to the unknown true function $f(X)$.

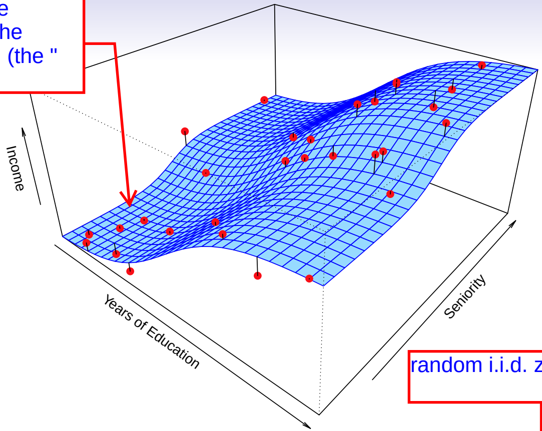
A linear model $\hat{f}_L(X) = \hat{\beta}_0 + \hat{\beta}_1 X$ gives a reasonable fit here



A quadratic model $\hat{f}_Q(X) = \hat{\beta}_0 + \hat{\beta}_1 X + \hat{\beta}_2 X^2$ fits slightly better.



we're simulating the training data from the regression function (the " truth")

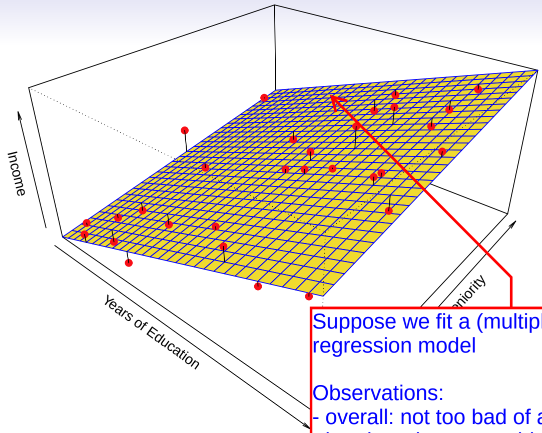


random i.i.d. zero-mean

Simulated example. Red points are simulated values for income from the model

$$\texttt{income} = f(\texttt{education}, \texttt{seniority}) + \epsilon$$

f is the blue surface.



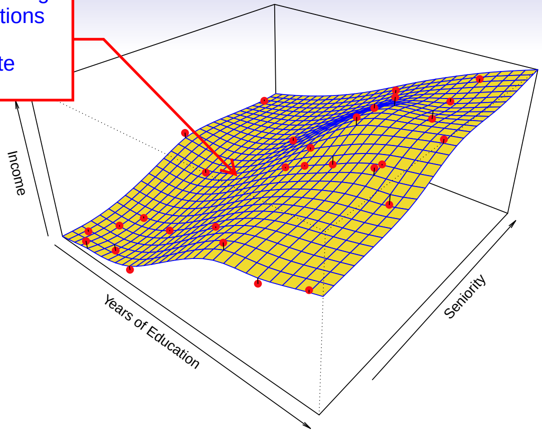
Suppose we fit a (multiple) linear regression model

Observations:
- overall: not too bad of a fit to the data
- but there is some evidence of systematic "bias" in the fitted model:
    - some of the fitted errors are mostly positive in certain regions, and others mostly negative
- assuming that there is model misspecification, reducible error will not go to 0 as sample size increases

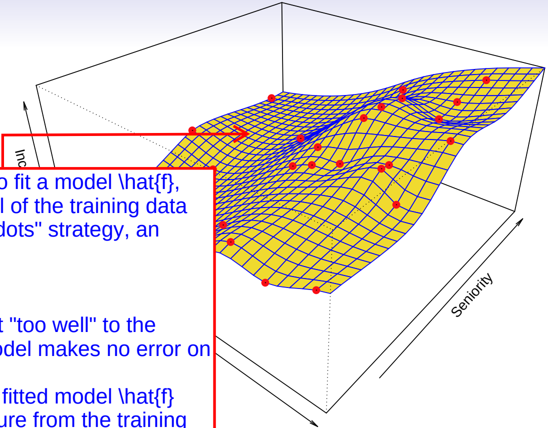
Linear regression model fit to the si

$\hat{f}_L(\texttt{education}, \texttt{seniority}) = \hat{\beta}_0 + \hat{\beta}$

Non-parametric modeling:
- much less assumptions on the form of f
- e.g., KNN, thin-plate splines



More flexible regression model $\hat{f}_S($**education**$,$ **seniority**$)$ fit to the simulated data. Here we use a technique called a *thin-plate spline* to fit a flexible surface. We control the roughness of the fit (chapter 7).



Suppose we choose to fit a model \hat{f}, which goes through all of the training data points - "connect-the-dots" strategy, an interpolator

Observations:
- predictor seems to fit "too well" to the training data, fitted model makes no error on training data
- overfitting: when the fitted model \hat{f} learns incorrect structure from the training data noise
- results in an overly complex / overly flexible model
- an overfit model has high VARIANCE:
    - if we draw a new training dataset from the generative model, and we fit a new predictor \hat{f}*, then \hat{f}* will likely be very different from \hat{f}
    - think connect-the-dots

gression model
fit to the simulated data. Here the
s on the training data! Also known

Some trade-offs

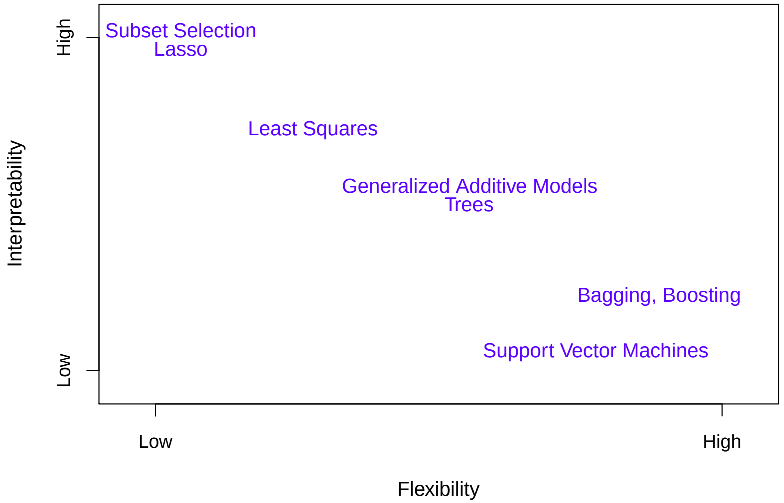
- Prediction accuracy versus interpretability.
  — Linear models are easy to interpret; thin-plate splines are not.

Some trade-offs

- Prediction accuracy versus interpretability.
  — Linear models are easy to interpret; thin-plate splines are not.
- Good fit versus over-fit or under-fit.
  — How do we know when the fit is just right?

Some trade-offs

- Prediction accuracy versus interpretability.
  — Linear models are easy to interpret; thin-plate splines are not.
- Good fit versus over-fit or under-fit.
  — How do we know when the fit is just right?
- Parsimony versus black-box.
  — We often prefer a simpler model involving fewer variables over a black-box predictor involving them all.


Subset Selection
Lasso
Least Squares
Generalized Additive Models
Trees
Bagging, Boosting
Support Vector Machines
Interpretability
High
Low
Flexibility
Low
High

Assessing Model Accuracy

Suppose we fit a model $\hat{f}(x)$ to some training data $\mathsf{Tr} = \{x_i, y_i\}_1^N$, and we wish to see how well it performs.

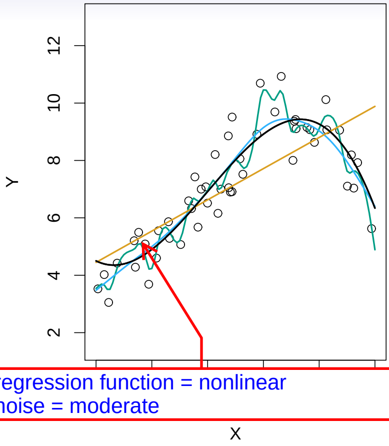
- We could compute the average squared prediction error over Tr:

$$\text{MSE}_{\mathsf{Tr}} = \text{Ave}_{i \in \mathsf{Tr}}[y_i - \hat{f}(x_i)]^2$$

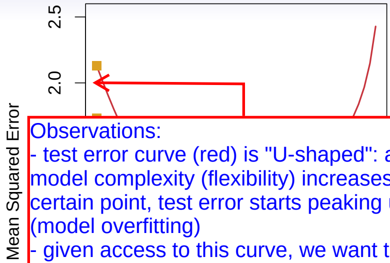
This may be biased toward more overfit models.

- Instead we should, if possible, compute it using fresh *test* data $\mathsf{Te} = \{x_i, y_i\}_1^M$:

$$\text{MSE}_{\mathsf{Te}} = \text{Ave}_{i \in \mathsf{Te}}[y_i - \hat{f}(x_i)]^2$$

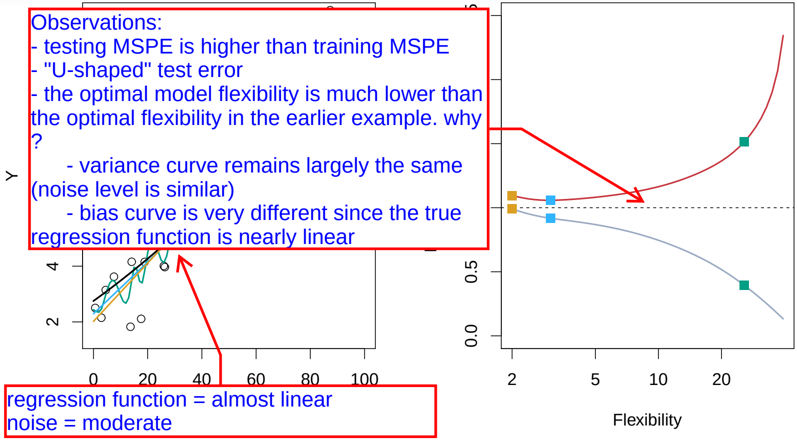


regression function = nonlinear
noise = moderate



Observations:
- test error curve (red) is "U-shaped": as model complexity (flexibility) increases past a certain point, test error starts peaking up (model overfitting)
- given access to this curve, we want to be at the "optimal" model complexity to mimimize MSPE (blue model)
- training MSPE curve (gray): always decreases as model complexity increases, hence it's not a good criterion for prediction
- testing MSPE generally higher than training MSPE. why? model is fitted to minimize training error, and not testing

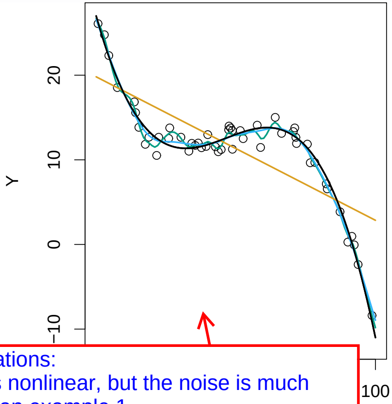
Black curve is truth. Red curve on right is $\mathrm{MSE}_{\mathsf{Te}}$, grey curve is $\mathrm{MSE}_{\mathsf{Tr}}$. Orange, blue and green curves/squares correspond to fits of different flexibility.

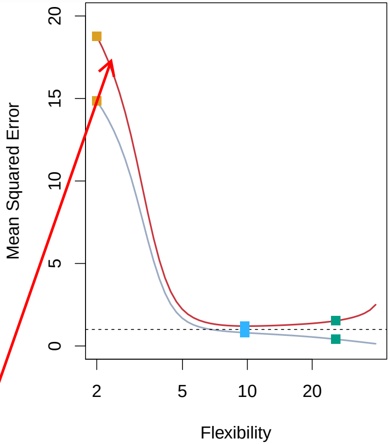


Observations:
- testing MSPE is higher than training MSPE
- "U-shaped" test error
- the optimal model flexibility is much lower than the optimal flexibility in the earlier example. why ?
    - variance curve remains largely the same (noise level is similar)
    - bias curve is very different since the true regression function is nearly linear

regression function = almost linear
noise = moderate

Here the truth is smoother, so the smoother fit and linear model do really well.





Observations:
- truth is nonlinear, but the noise is much lower than example 1
- there is much less danger in overfitting the model
    - bias curve remains largely the same as example 1 (similar degrees of nonlinearity)
    - variance curve increases much more slowly, since with less noise there is less risk of overfitting
- the optimal model for prediction is much more complex

se is low, so the more flexible fits

Bias-Va

Suppose we have fit a mod
let (x_0, y_0) be a test observ
the true model is $Y = f(X$
then

$$E\left(y_0 - \hat{f}(x_0)\right)^2 = \mathrm{Var}$$

The expectation averages o
the variability in Tr. Note

Typically as the *flexibility* of $\hat{f}$ increases, its variance increases, and its bias decreases. So choosing the flexibility based on average test error amounts to a *bias-variance trade-off.*

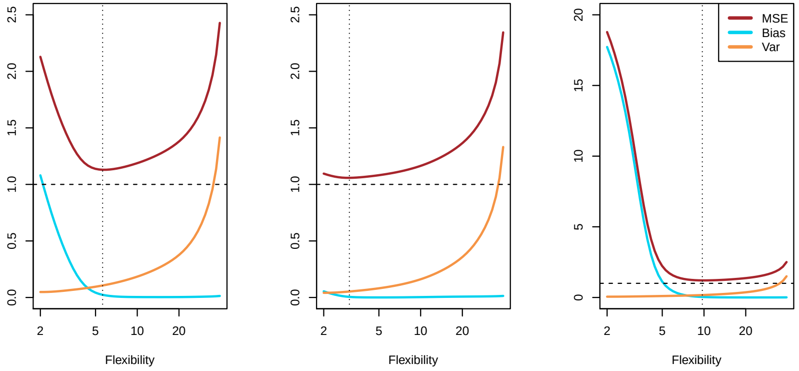
- LHS: testing MSPE which we wish to minimize
- RHS: reducible + irreducible error (from earlier)

The reducible error has two components: bias & variance
BIAS:
- "error from approximating a complex regression function with a simpler model", i.e., model misspecification
- linear models are often biased, unless the true regression function is linear
VARIANCE:
- "how much the predictor \hat{f} changes when we estimate this with a new training dataset"
- linear models have little variance: smaller # of parameters to fit with the data
- flexible models have high variance (connect-the-dots)

TRADE-OFF: find an optimal compromise between bias and variance for fitting \hat{f}

derivation on board

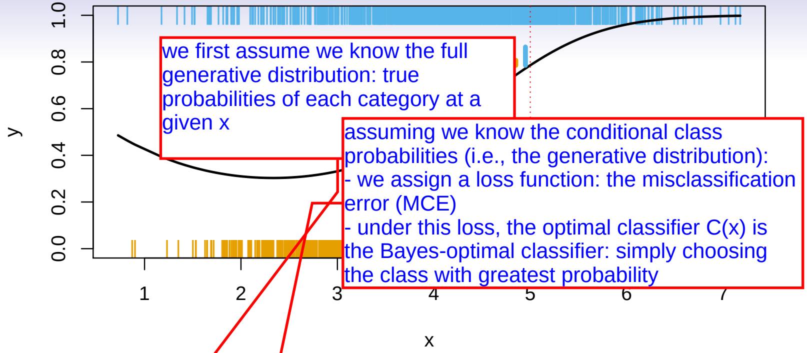
Bias-variance trade-off for the three examples



Classification Problems

Here the response variable Y is *qualitative* — e.g. email is one of $\mathcal{C} = (\texttt{spam}, \texttt{ham})$ (`ham`=good email), digit class is one of $\mathcal{C} = \{\texttt{0}, \texttt{1}, \ldots, \texttt{9}\}$. Our goals are to:

- Build a classifier $C(X)$ that assigns a class label from $\mathcal{C}$ to a future unlabeled observation X.
- Assess the uncertainty in each classification
- Understand the roles of the different predictors among $X = (X_1, X_2, \ldots, X_p)$.



we first assume we know the full generative distribution: true probabilities of each category at a given x

assuming we know the conditional class probabilities (i.e., the generative distribution):
- we assign a loss function: the misclassification error (MCE)
- under this loss, the optimal classifier C(x) is the Bayes-optimal classifier: simply choosing the class with greatest probability

Is there an ideal $C(X)$? Suppose the K elements in $\mathcal{C}$ are numbered $1, 2, \ldots, K$. Let

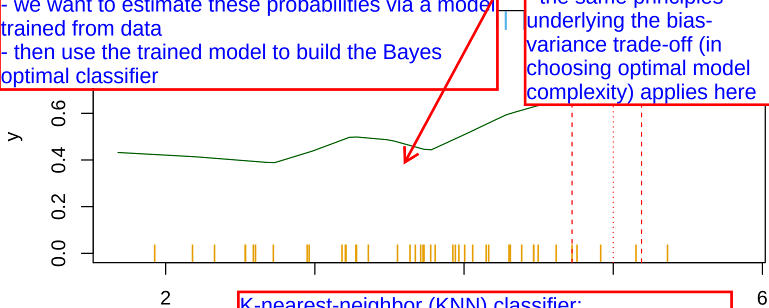
$$p_k(x) = \Pr(Y = k|X = x),\ k = 1, 2, \ldots, K.$$

These are the *conditional class probabilities* at x; e.g. see little barplot at $x = 5$. Then the *Bayes optimal* classifier at x is

$$C(x) = j \text{ if } p_j(x) = \max\{p_1(x), p_2(x), \ldots, p_K(x)\}$$

Similar to the regression problem: we don't know what the conditional class probabilities are in practice!
- we want to estimate these probabilities via a model trained from data
- then use the trained model to build the Bayes optimal classifier

- an alternate (parametric) model for the conditional class probabilities would be logistic regression.
- the same principles underlying the bias-variance trade-off (in choosing optimal model complexity) applies here



K-nearest-neighbor (KNN) classifier:
- e.g., let's build the trained classifier at x=5 with K= 10 neighbors
- we see that there are 4 0's and 6 1's of the K=10 nearest neighbors
- using this, our estimate of p1(x) = 60%, of p0(x) = 40%
- using these estimated probabilities, since our estimate of p1(x) > p0(x), our estimated Bayes optimal classifier is class 1

Nearest-neighbor a

Also breaks down a ct on

$\hat{C}(x)$ is less than or

Classification: some details

- Typically we measure the performance of $\hat{C}(x)$ using the misclassification error rate:

$$\text{Err}_{\textsf{Te}} = \text{Ave}_{i \in \textsf{Te}} I[y_i \neq \hat{C}(x_i)]$$

- The Bayes classifier (using the true $p_k(x)$) has smallest error (in the population).

Classification: some details

- Typically we measure the performance of $\hat{C}(x)$ using the misclassification error rate:

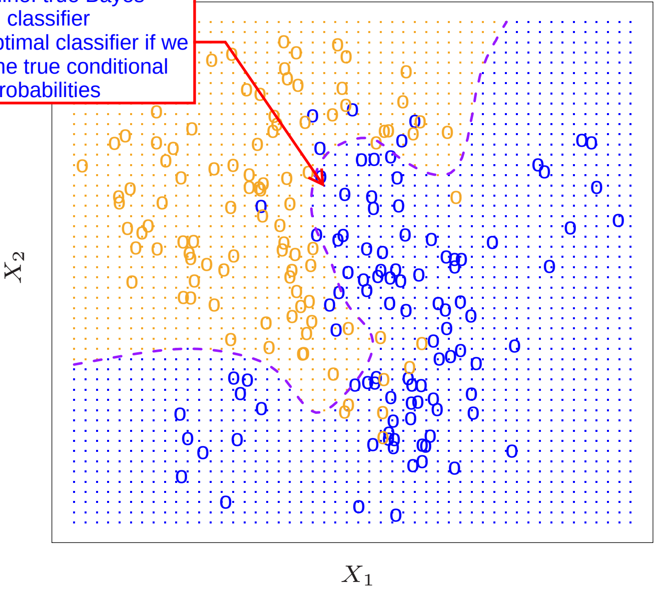
$$\mathrm{Err}_{\mathsf{Te}} = \mathrm{Ave}_{i \in \mathsf{Te}} I[y_i \neq \hat{C}(x_i)]$$

- The Bayes classifier (using the true $p_k(x)$) has smallest error (in the population).

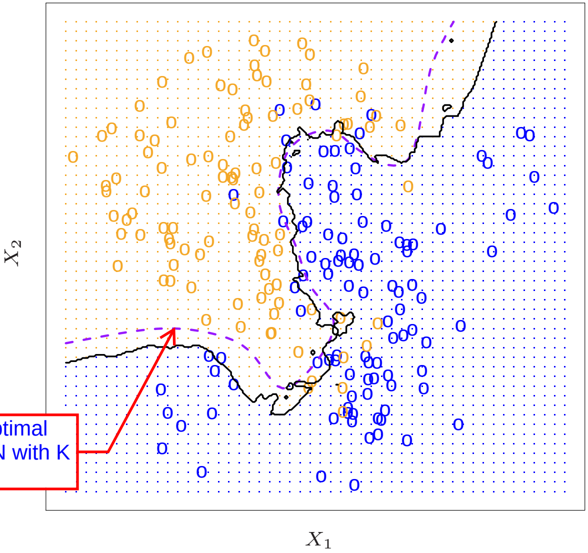
- Support-vector machines build structured models for $C(x)$.
- We will also build structured models for representing the $p_k(x)$. e.g. Logistic regression, generalized additive models.

Example: K-nearest neighbors in two dimensions

purple line: true Bayes optimal classifier
- the optimal classifier if we know the true conditional class probabilities



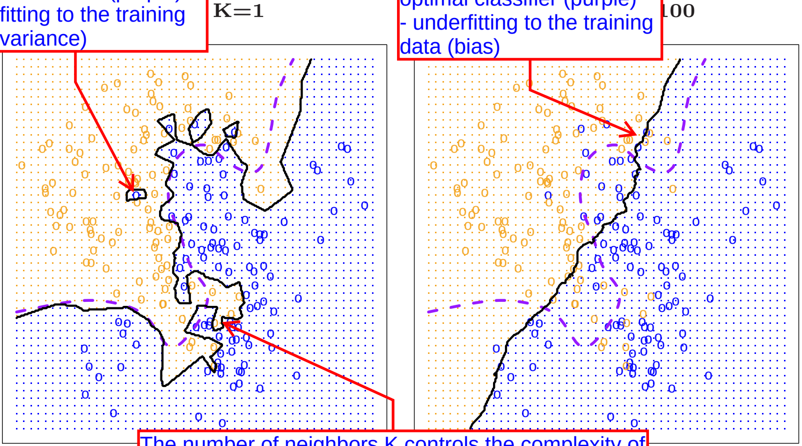
KNN: K=10



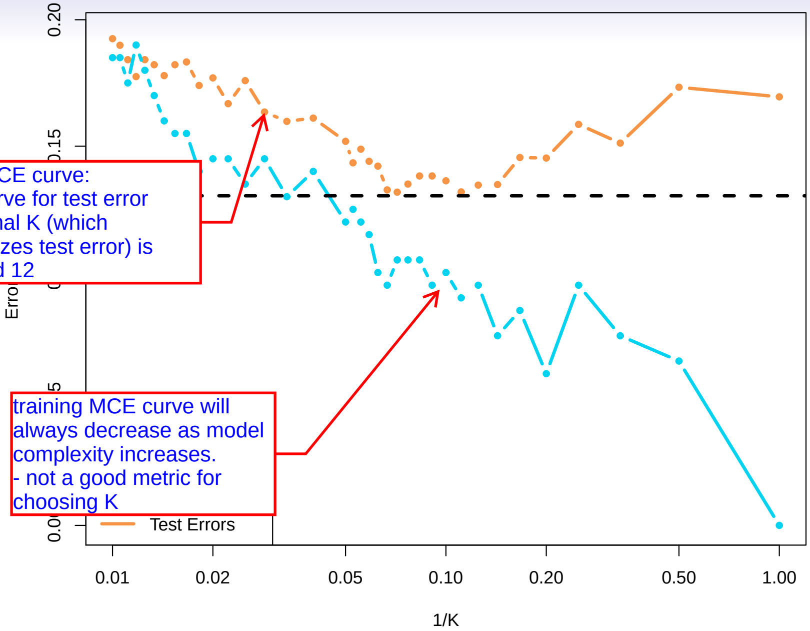
estimated Bayes optimal classifier using KNN with K = 10 neighbors

black line: trained classifier using KNN with K = 1
- not too close to the true optimal classifier (purple)
- overfitting to the training data (variance)

black line: trained classifier using KNN with K=100
- not close at all to the true optimal classifier (purple)
- underfitting to the training data (bias)



The number of neighbors K controls the complexity of our classifier: bias-variance trade-off



test MCE curve:
- U-curve for test error
- optimal K (which minimizes test error) is around 12

training MCE curve will always decrease as model complexity increases.
- not a good metric for choosing K