

# Data Mining

## W4240 Section 001

Giovanni Motta

Columbia University, Department of Statistics

October 5, 2015

- ▶ Today:
  - ▶ Supervised Learning
  - ▶ Mean Squared Error
  - ▶ Bias-Variance Tradeoff
  - ▶ Errors in Classification
  - ▶ Image Processing Example

# Outline

Supervised Learning

Mean Squared Error

Bias and Variance

Errors in Classification

Image Processing Example

# Supervised Learning

Recall from before...

$$\begin{array}{ll} X = (X_1, X_2, \dots, X_p)^T & \text{inputs} \\ Y & \text{output} \\ Y = f(X) + \epsilon & \text{relationship} \end{array}$$

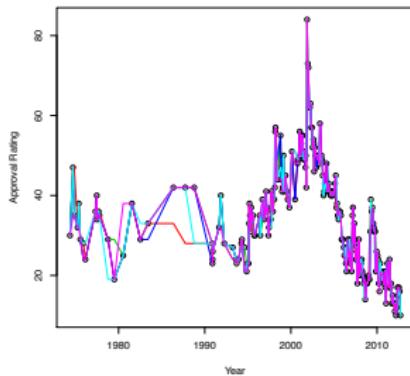
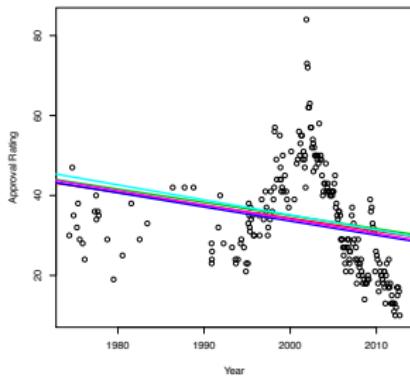
We are interested in studying  $f$  in two settings: regression and classification.

**Regression:**  $Y$  has continuous values, like \$81,200 or 72.

**Classification:**  $Y$  has categorical values, like low/medium/high or red/green/blue

# Supervised Learning

Recall from before...



We looked at least squares linear regression and  $k$ -nearest neighbors.

# Linear Regression

Linear regression:

- ▶ (we haven't covered this yet, but we know how it looks...)
- ▶ fit a global model, restricted to be a linear function of the covariates
- ▶ Our data is  $y = f(\mathbf{x}) + \epsilon$
- ▶ We fit

$$\hat{y} = \hat{f}(\mathbf{x}) = w_0 + \sum_{j=1}^p w_j x_j$$

- ▶ aka

$$\hat{y} = \mathbf{w}^\top \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}$$

- ▶ (The function is a linear combination of the covariates)
- ▶ (And how does this look in data matrix form?)

# $k$ -Nearest Neighbors Regression

Idea: fit a local model by averaging the values of the  $k$  closest observations

$$\hat{y} = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

where  $N_k(x)$  is the set of observations with the  $k$  smallest distances to the query point  $x$

(remember also we can do  $k$ -nearest neighbor *classification*)

Questions:

- ▶ Is linear regression or  $k$ NN better?
- ▶ Which  $k$  is best?
- ▶ Is there a best?

# Supervised Learning

Why is there no single best estimator?

# Supervised Learning

How do I measure an estimator? Prediction error is a logical choice:

Ideally: **Expected Mean Squared Error**:

$$\mathbb{E} \left[ (Y - \hat{Y})^2 \mid X = x_0 \right]$$

This measures the expected ability of an estimator (given random training data) to predict a new outcome at the point  $x_0$ . You can view this as the *generalization error*.

Why do I want this?

# Supervised Learning

Why squared error? First, assume we can parameterize our estimator  $f$  by some set of parameters  $\theta$ .

Assume  $\mathbb{P}(Y | X, \theta) = N(f_\theta(X), \sigma^2)$

$$\begin{aligned}\ell(\theta) &= -\frac{n}{2} \log(2\pi) - n \log \sigma - \frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - f_\theta(x_i))^2 \\ &\propto -\frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - f_\theta(x_i))^2\end{aligned}$$

*Gaussian errors mean squared loss function!*

# Supervised Learning

Really want **Expected Mean Squared Error**:

$$\mathbb{E} \left[ (Y - \hat{Y})^2 \mid X = x_0 \right]$$

This requires:

- ▶ ability to compute expectation over all possible outcomes for  $\hat{Y}$  (different training data)
- ▶ ability to compute expectation over  $Y$  (know distribution of response for a given  $x_0$ )

I have:  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$

# Outline

Supervised Learning

Mean Squared Error

Bias and Variance

Errors in Classification

Image Processing Example

# Mean Squared Error

Well, what if I just approximate

$$\mathbb{E} \left[ (Y - \hat{Y})^2 \mid X = x_0 \right]$$

with my data?

$$MSE_{training} = \frac{1}{n} \sum_{i=1}^n \left( y_i - \hat{f}(x_i) \right)^2$$

# Mean Squared Error

New metric:

$$MSE_{training} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{f}(x_i))^2$$

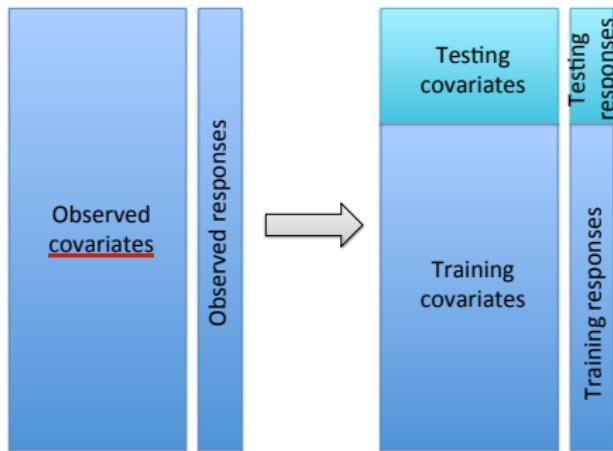
Relevant questions:

- ▶ What function will minimize this?
  
- ▶ Is this function a good predictor?

# Mean Squared Error

How do we estimate generalization error?

- 1) Break samples into training set and testing set (and possibly validation)

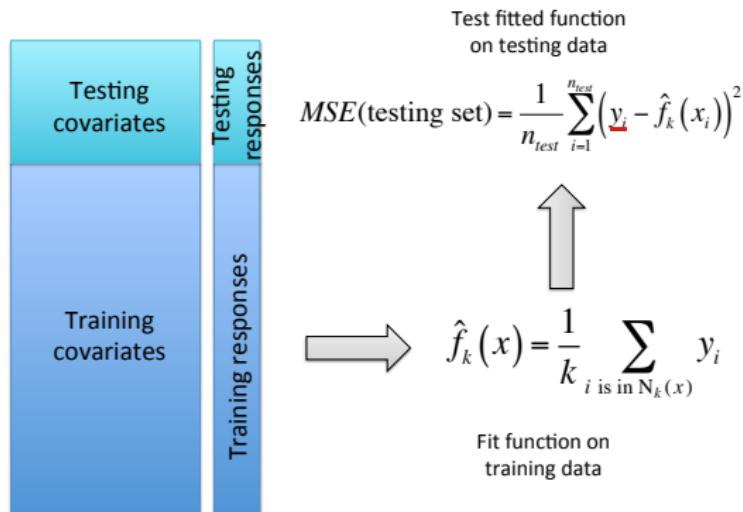


(it is usually a good idea to randomize the order of the data first)

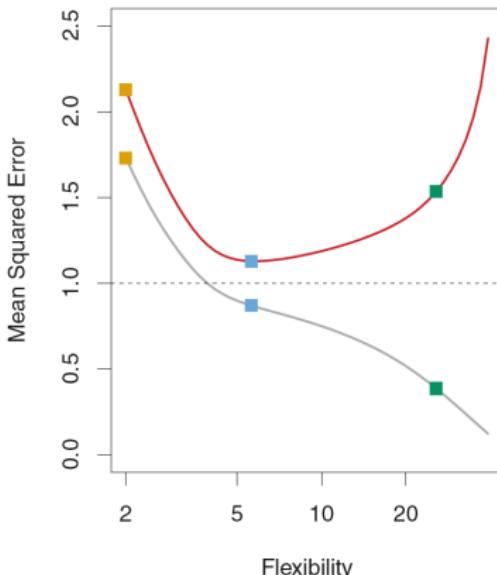
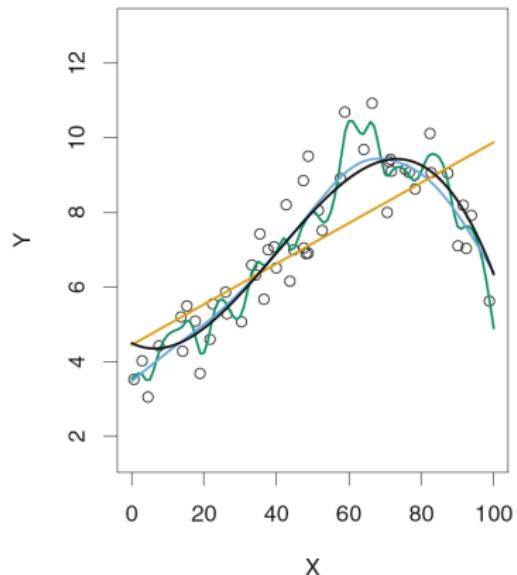
# Mean Squared Error

How do we estimate generalization error?

- 2) Fit function on training set
- 3) Apply it to test set and estimate error



# Mean Squared Error



Why do you think the testing MSE has a U shape? What about the training MSE?<sup>1</sup>

---

<sup>1</sup>Some images taken from *An Introduction to Statistical Learning* by James, Witten, Hastie and Tibshirani.

# Outline

Supervised Learning

Mean Squared Error

Bias and Variance

Errors in Classification

Image Processing Example

# Bias-Variance Decomposition

Training data  $\mathcal{T} = \{(x_i, y_i)\}_{i=1}^n$ .

- ▶ What is the mean squared error between the true  $f(x_0)$  and the approximated value for a fixed  $x_0$ ?
- ▶ Important:  $\hat{y} = \hat{f}(x_0)$  is a random variable here (determined by noisy  $\mathcal{T}$ )
- ▶ Hence the expected reducible error,  $\mathbb{E}$  over  $\mathcal{T}$ , is:

$$\begin{aligned} MSE(x_0) &= E[f(x_0) - \hat{f}(x_0)]^2 \\ &= E\left[\hat{f}(x_0) - E[\hat{f}(x_0)] + E[\hat{f}(x_0)] - f(x_0)\right]^2 \\ &\quad \dots \end{aligned}$$

## Bias-Variance Decomposition

$$\begin{aligned}MSE(x_0) &= E[f(x_0) - \hat{f}(x_0)]^2 \\&= E\left[\hat{f}(x_0) - E[\hat{f}(x_0)] + E[\hat{f}(x_0)] - f(x_0)\right]^2 \\&= E\left[\hat{f}(x_0) - E[\hat{f}(x_0)]\right]^2 + E\left[E[\hat{f}(x_0)] - f(x_0)\right]^2 \\&\quad + 2E\left[\hat{f}(x_0) - E[\hat{f}(x_0)]\right]E\left[E[\hat{f}(x_0)] - f(x_0)\right] \\&= E\left[\hat{f}(x_0) - E[\hat{f}(x_0)]\right]^2 + E\left[E[\hat{f}(x_0)] - f(x_0)\right]^2 \\&= E\left[\hat{f}(x_0) - E[\hat{f}(x_0)]\right]^2 + \left(E[\hat{f}(x_0)] - f(x_0)\right)^2 \\&= \text{Var}(\hat{f}(x_0)) + \text{Bias}^2(\hat{f}(x_0))\end{aligned}$$

*Generalization error is estimator variance plus bias squared!  
Note that all these terms are nonnegative!*

# Bias-Variance Decomposition

Bias:  $E[\hat{f}(x_0)] - f(x_0)$

- ▶ error caused by difference between expected estimator and true function
- ▶ we say an estimator is unbiased if  $E[\hat{f}(x_0)] - f(x_0) = 0$
- ▶ (estimate a mean with  $\frac{1}{n} \sum z_i$  and  $\frac{1}{n+100} \sum z_i$ )

Variance:  $E \left[ \hat{f}(x_0) - \mathbb{E}[\hat{f}(x_0)] \right]^2$

- ▶ this is a measure of *estimator spread*... that is, how much does the estimator vary with a new data set?

Reducible error is:

$$E[f(x_0) - \hat{f}(x_0)]^2 = \text{Var}(\hat{f}(x_0)) + \text{Bias}^2(\hat{f}(x_0))$$

# Bias-Variance Decomposition

Compare the predicted value with the actual value...  
(remember  $y = f(x) + \epsilon$ )

$$\begin{aligned} E[y_0 - \hat{f}(x_0)]^2 &= E[f(x_0) + \epsilon - \hat{f}(x_0)]^2 \\ &= E[f(x_0) - \hat{f}(x_0)]^2 + E[\epsilon]^2 + 2E\left[\epsilon(f(x_0) - \hat{f}(x_0))\right] \\ &= \text{Var}(\hat{f}(x_0)) + \text{Bias}^2(\hat{f}(x_0)) + \text{Var}(\epsilon) \end{aligned}$$

MSE = estimator variance + estimator bias<sup>2</sup> + noise variance

MSE = reducible + irreducible error

# Bias-Variance Decomposition

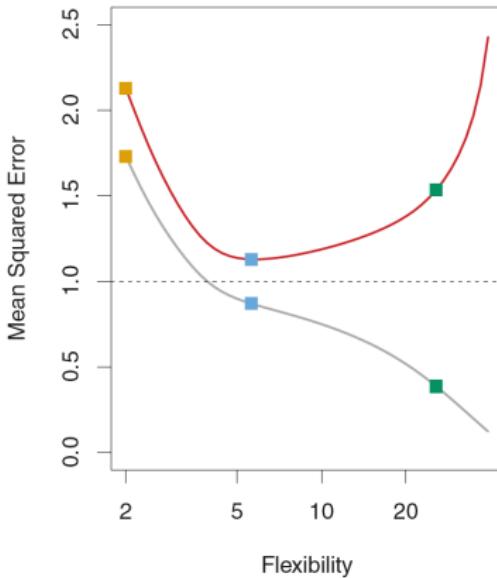
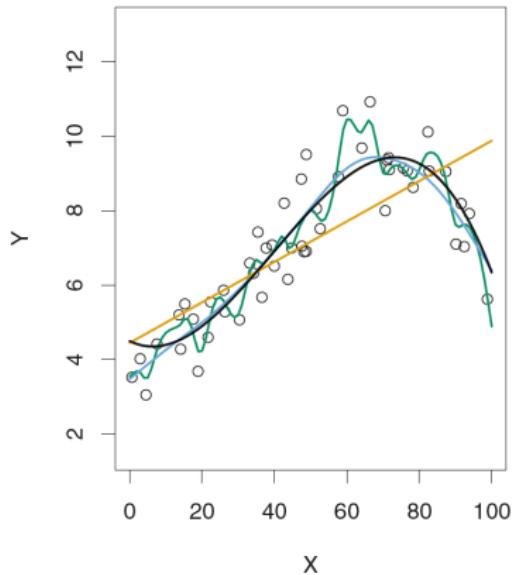
Let's think about estimators for a minute and answer some questions:

- ▶ Do **more flexible** estimators have more or less *bias* than **less flexible** estimators?
- ▶ Do **more flexible** estimators have more or less *variance* than **less flexible** estimators?
- ▶ Does **more data** increase or reduce the *bias* of an estimator?
- ▶ Does **more data** increase or reduce the *variance* of an estimator?

What rules of thumb can we get about data size? about model complexity?

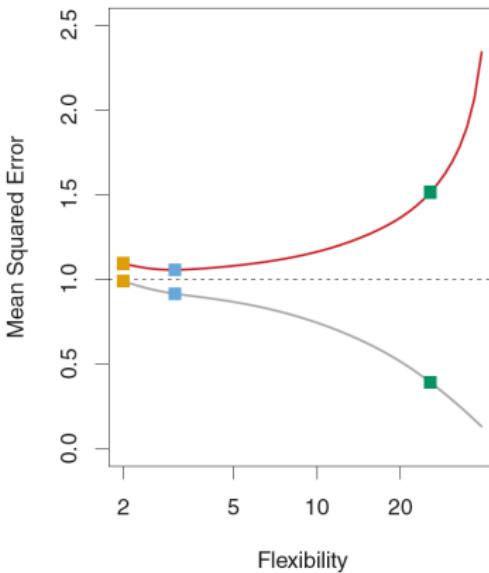
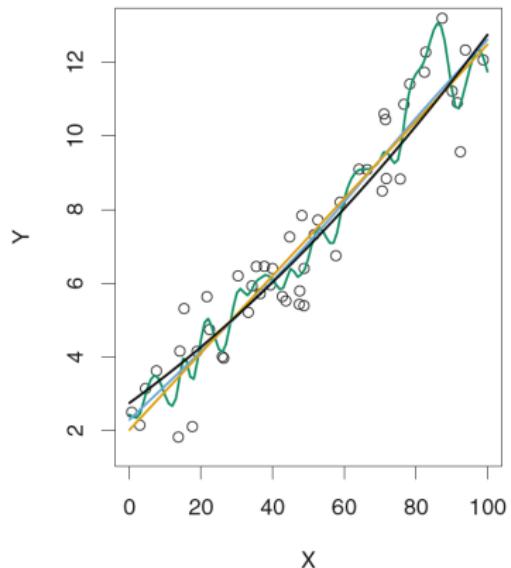
# Mean Squared Error

Let's look at this picture again. Is there any explanation in terms of estimator bias and variance?



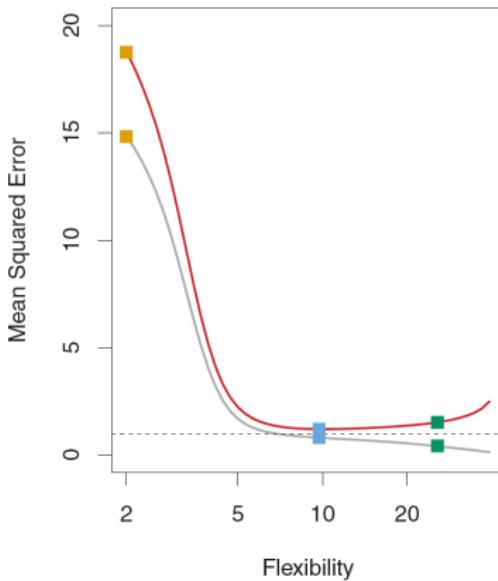
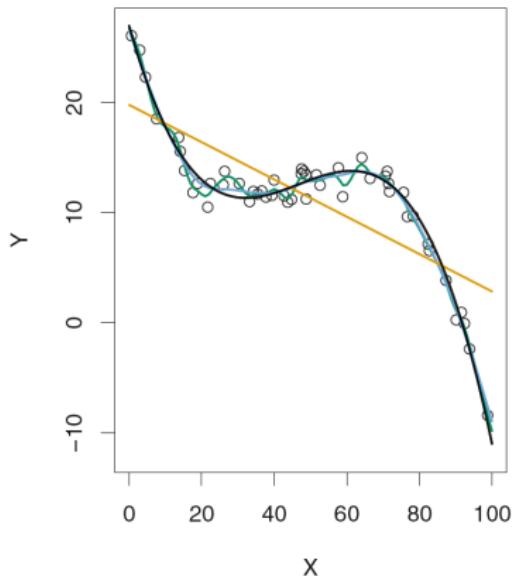
# Mean Squared Error

Let's look at another picture. Is there any explanation in terms of estimator bias and variance?



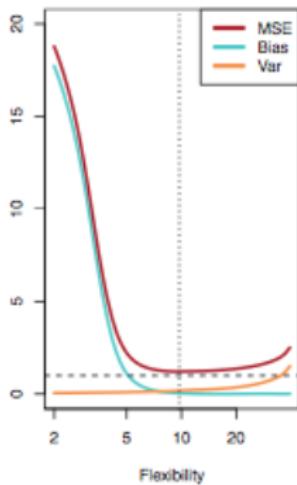
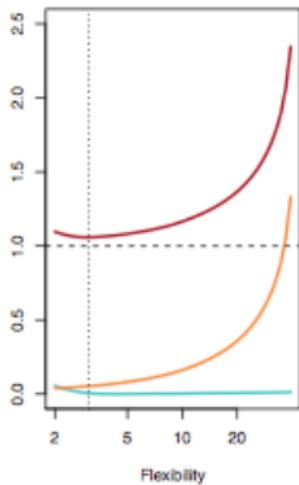
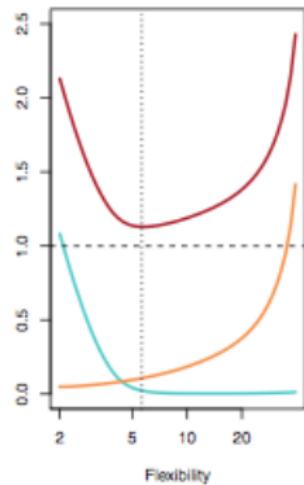
# Mean Squared Error

What about this third picture? Is there any explanation in terms of estimator bias and variance?



# Mean Squared Error

Here is a breakdown of MSE in terms of estimator bias, estimator variance and noise variance.



# Bias-Variance Decomposition: One more time

Compare the predicted value with the actual value...  
(remember  $y = f(x) + \epsilon$ )

$$\begin{aligned} & E[y_0 - \hat{f}(x_0)]^2 \\ = & E[f(x_0) + \epsilon - \hat{f}(x_0)]^2 \\ = & E[f(x_0) - \hat{f}(x_0)]^2 + E[\epsilon]^2 \\ = & E\left[\hat{f}(x_0) - E[\hat{f}(x_0)]\right]^2 + \left(E[\hat{f}(x_0)] - f(x_0)\right)^2 + E(\epsilon^2) \\ = & \text{Var}(\hat{f}(x_0)) + \text{Bias}^2(\hat{f}(x_0)) + \text{Var}(\epsilon) \end{aligned}$$

MSE = estimator variance + estimator bias<sup>2</sup> + noise variance  
MSE = reducible + irreducible error

# Outline

Supervised Learning

Mean Squared Error

Bias and Variance

Errors in Classification

Image Processing Example

# Classification Generalization Error

MSE is not a great metric for classification. (Errors not Gaussian!)  
Instead, we estimate the *error rate*, or the proportion misclassified:

$$Err = \mathbb{E} \left[ \mathbf{1}_{\{Y_0 \neq \hat{Y}_0\}} \right]$$

Here  $\mathbf{1}_{\{A\}}$  (denoted  $I(A)$  in book) is an *indicator function*:

$$\mathbf{1}_{\{A\}} = \begin{cases} 1 & \text{if } A \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

Training error rate:

$$Err_{training} = \frac{1}{n} \sum_{i=1}^n \mathbf{1}_{\{y_i \neq \hat{y}_i\}}$$

# The Bayes Classifier

It can be shown that the classifier that minimizes

$$Err = \mathbb{E} [\mathbf{1}_{\{Y_0 \neq \hat{Y}_0\}}]$$

assigns the predictive label to the class with the highest conditional probability:

$$\hat{Y}(x_0) = \arg \max \mathbb{P}(Y = j | X = x_0).$$

This is called the *Bayes Classifier*. The error rate is

$$1 - \mathbb{E} \left( \max_j \mathbb{P}(Y = j | X) \right).$$

Why can't we just use the Bayes classifier?

# The Bayes Classifier

Trying to get  $\mathbb{P}(Y = j | X = x_0)$ :

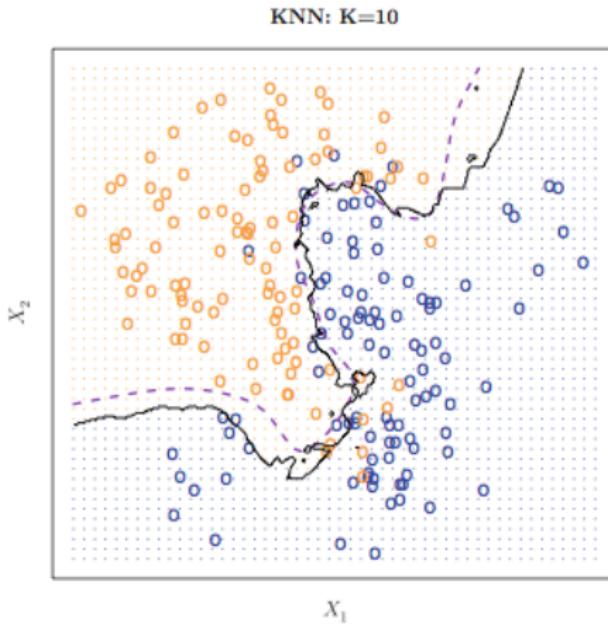
- ▶ can't really condition on event  $\{X = x_0\}$
- ▶ ...but we can condition on event  $\{X \text{ near } x_0\}$
- ▶ ...so let's choose the  $k$  closest neighbors to  $x_0$

$$\mathbb{P}(Y = j | X = x_0) \approx \frac{1}{k} \sum_{i \in N_k(x_0)} \mathbf{1}_{\{y_i=j\}}$$

For classification, kNN is an approximation to the Bayes classifier!

# The Bayes Classifier and kNN

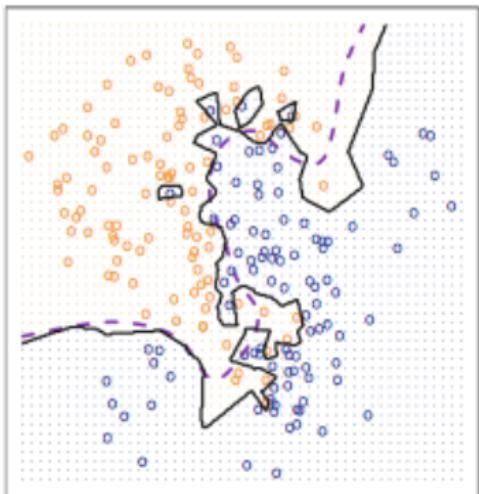
Let's compare the classification boundary produced by kNN (solid) and the Bayes classifier (dashed):



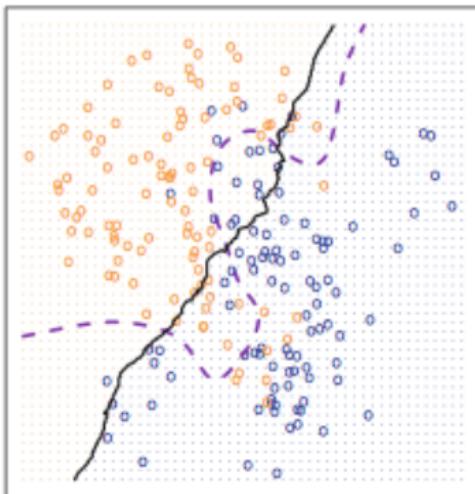
# The Bayes Classifier and kNN

What happens if we change  $k$ ?

KNN: K=1



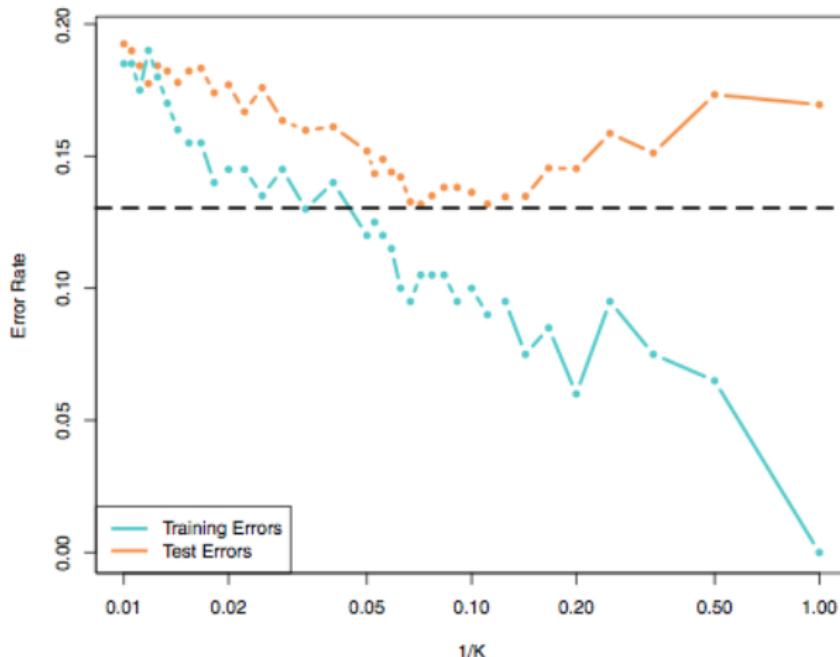
KNN: K=100



What happens to the bias? Variance?

# The Bayes Classifier and kNN

What happens if we change  $k$ ?



Can you explain this in terms of bias and variance?

# Outline

Supervised Learning

Mean Squared Error

Bias and Variance

Errors in Classification

Image Processing Example

# Fun with kNN: Image Processing



- ▶ This is a  $512 \times 512$  pixel portable networks graphics (.png) image
- ▶ Each pixel has an associated RGB (red, green, blue) value taking 1 of 256 values
- ▶ Stored as a  $512 \times 512 \times 3$  array

# Image Processing

How do I get that into R?

- ▶ various graphics packages, some focused on different image types
- ▶ I used the package png (how to download a package?)

```
> library(png)
> lena.im <- readPNG("Lenna.png")
> dim(lena.im)
[1] 512 512    3
```

# Image Processing

```
> lena.im[1:5,1:5,]  
, , 1  
  
 [,1]      [,2]      [,3]      [,4]      [,5]  
[1,] 0.8862745 0.8862745 0.8745098 0.8745098 0.8862745  
[2,] 0.8862745 0.8862745 0.8745098 0.8745098 0.8862745  
[3,] 0.8862745 0.8862745 0.8745098 0.8745098 0.8862745  
[4,] 0.8862745 0.8862745 0.8745098 0.8745098 0.8862745  
[5,] 0.8862745 0.8862745 0.8745098 0.8745098 0.8862745  
  
, , 2  
  
 [,1]      [,2]      [,3]      [,4]      [,5]  
[1,] 0.5372549 0.5372549 0.5372549 0.5333333 0.5411765  
[2,] 0.5372549 0.5372549 0.5372549 0.5333333 0.5411765  
[3,] 0.5372549 0.5372549 0.5372549 0.5333333 0.5411765  
[4,] 0.5372549 0.5372549 0.5372549 0.5333333 0.5411765  
[5,] 0.5372549 0.5372549 0.5372549 0.5333333 0.5411765  
  
, , 3  
  
 [,1]      [,2]      [,3]      [,4]      [,5]  
[1,] 0.4901961 0.4901961 0.5215686 0.5019608 0.4705882  
[2,] 0.4901961 0.4901961 0.5215686 0.5019608 0.4705882  
[3,] 0.4901961 0.4901961 0.5215686 0.5019608 0.4705882  
[4,] 0.4901961 0.4901961 0.5215686 0.5019608 0.4705882  
[5,] 0.4901961 0.4901961 0.5215686 0.5019608 0.4705882
```

# Image Processing



Image reconstruction:

- ▶ Pixels are randomly corrupted (here 25%)
- ▶ How can we reconstruct the original image?

# Image Processing



Lena: original, 1nn

# Image Processing



Lena: original, 1nn, 3nn, 5nn (clockwise from upper left)

# Image Processing



Lena: original, 1nn, 10nn, 100nn (clockwise from upper left)

# Image Processing

Thought experiment:

- ▶ what would happen if we used *all* of the pixels as a neighborhood?
- ▶ would this reconstruction look better or worse to you? why?
- ▶ and why does kNN work so well in this case?