

Learning From Data
Lecture 16
Similarity and Nearest Neighbor

Similarity
Nearest Neighbor

M. Magdon-Ismail
CSCI 4100/6100

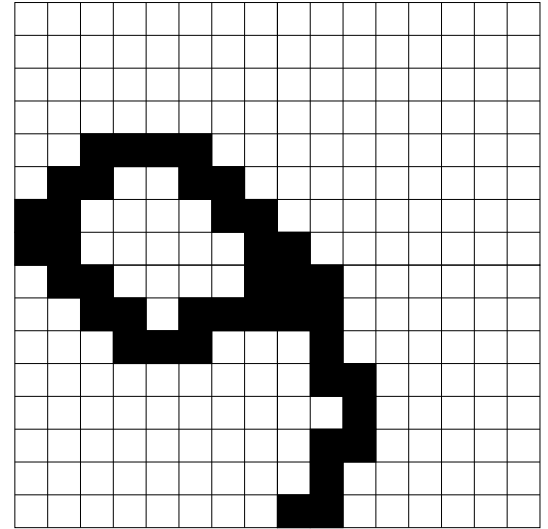
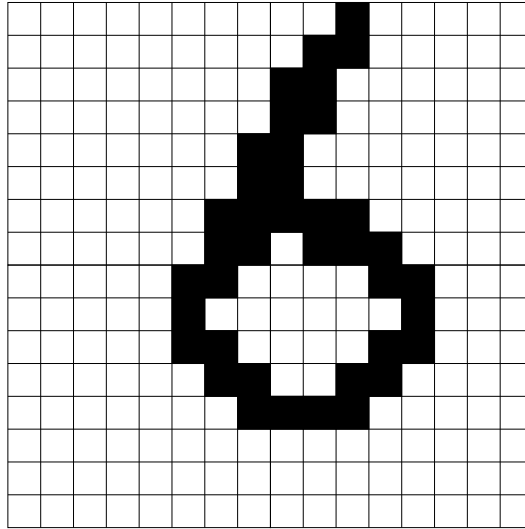
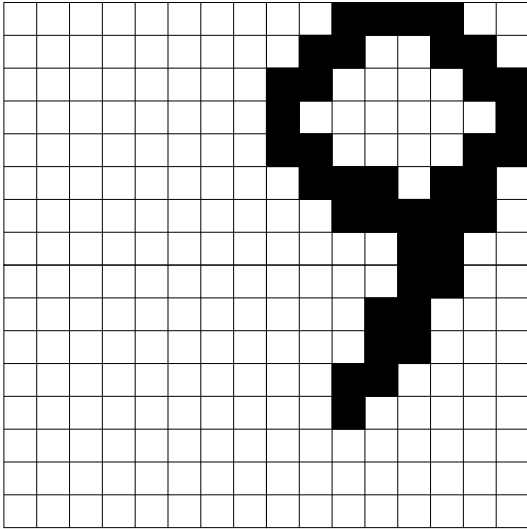
My 5-Year-Old Called It “A ManoHorse”

The simplest method of learning that we know.

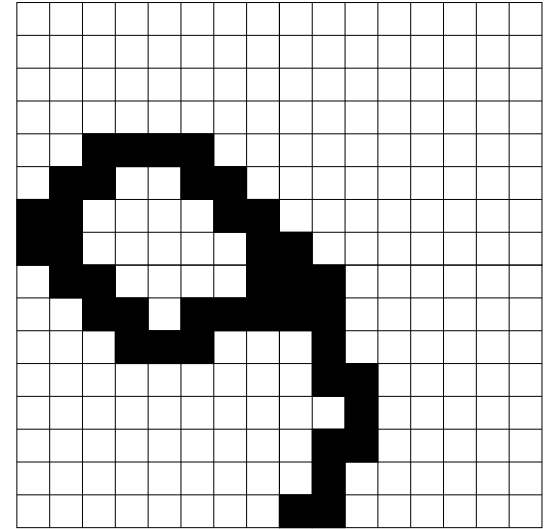
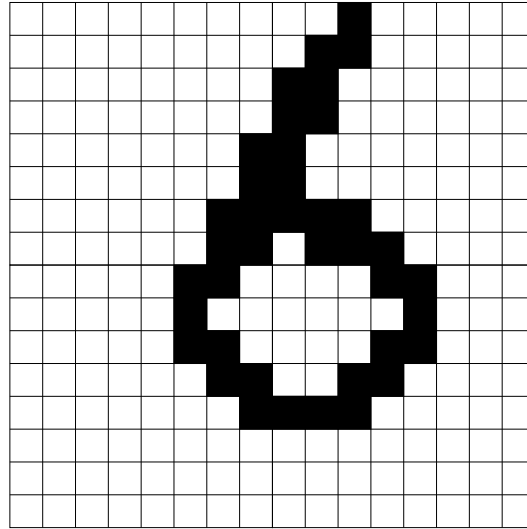
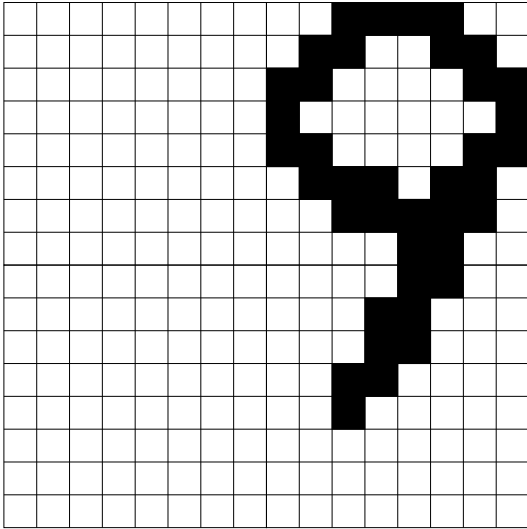
Classify according to [similar](#) objects you have seen.




Measuring Similarity



Measuring Similarity



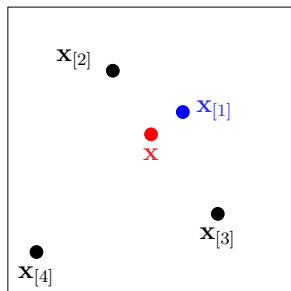
features, \mathbf{x}



$$d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|$$

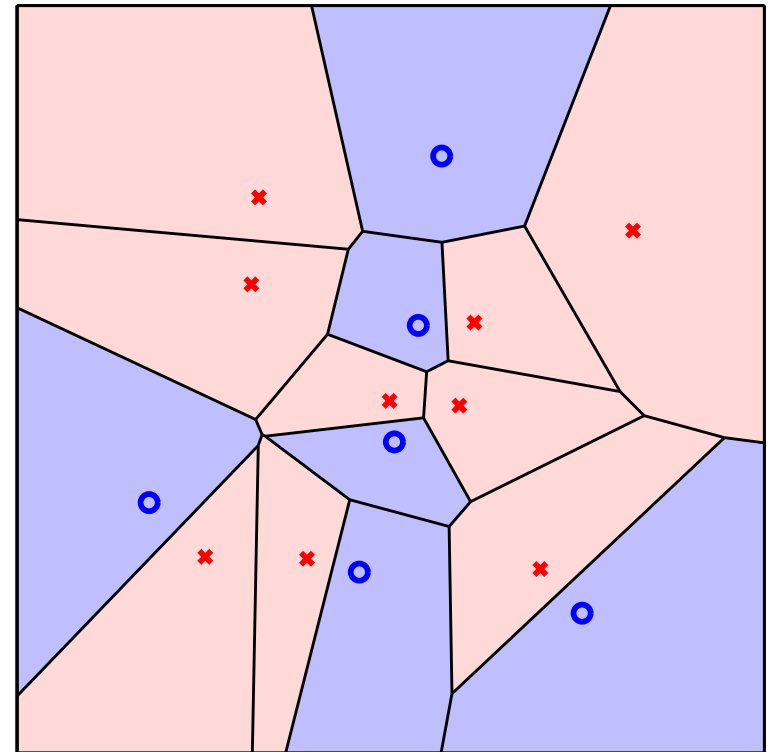
Nearest Neighbor

Test ' \mathbf{x} ' is classified using its nearest neighbor.



$$d(\mathbf{x}, \mathbf{x}_{[1]}) \leq d(\mathbf{x}, \mathbf{x}_{[2]}) \leq \dots \leq d(\mathbf{x}, \mathbf{x}_{[N]})$$

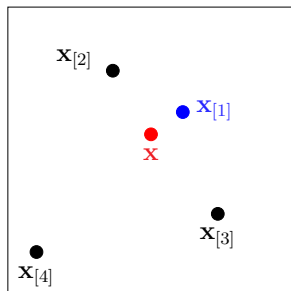
$$g(\mathbf{x}) = y_{[1]}(\mathbf{x})$$



Nearest neighbor Voronoi tessellation

Nearest Neighbor

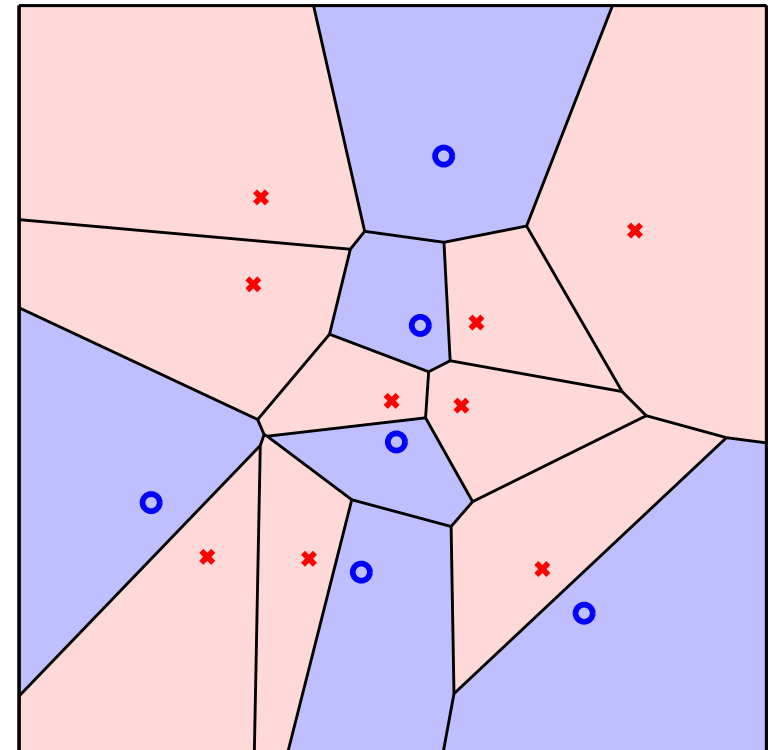
Test ' \mathbf{x} ' is classified using its nearest neighbor.



$$d(\mathbf{x}, \mathbf{x}_{[1]}) \leq d(\mathbf{x}, \mathbf{x}_{[2]}) \leq \dots \leq d(\mathbf{x}, \mathbf{x}_{[N]})$$

$$g(\mathbf{x}) = y_{[1]}(\mathbf{x})$$

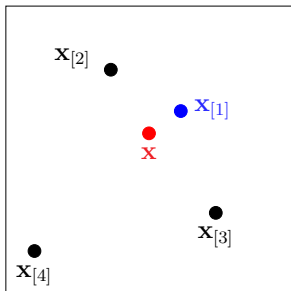
No training needed!



Nearest neighbor Voronoi tessellation

Nearest Neighbor

Test ' \mathbf{x} ' is classified using its nearest neighbor.

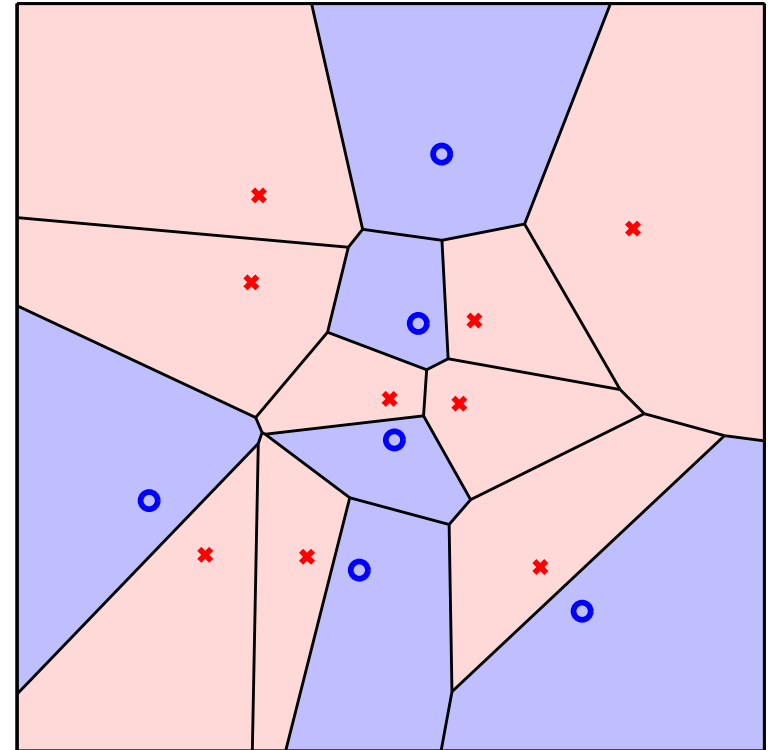


$$d(\mathbf{x}, \mathbf{x}_{[1]}) \leq d(\mathbf{x}, \mathbf{x}_{[2]}) \leq \dots \leq d(\mathbf{x}, \mathbf{x}_{[N]})$$

$$g(\mathbf{x}) = y_{[1]}(\mathbf{x})$$

No training needed!

$$E_{\text{in}} = 0$$



Nearest neighbor Voronoi tessellation

What about E_{out} ?

Theorem: $E_{\text{out}} \leq 2E_{\text{out}}^*$

(with high probability as $N \rightarrow \infty$)

VC analysis: E_{in} is an estimate for E_{out} .

Nearest neighbor analysis: $E_{\text{in}} = 0$, E_{out} is small.

So we will never know what E_{out} is, but it cannot be much worse than *the best anyone can do*.

Half the classification power of the data is in the nearest neighbor

Proving $E_{\text{out}} \leq 2E_{\text{out}}^*$

$$\pi(\mathbf{x}) = \mathbb{P}[y = +1 | \mathbf{x}].$$

← the target in logistic regression

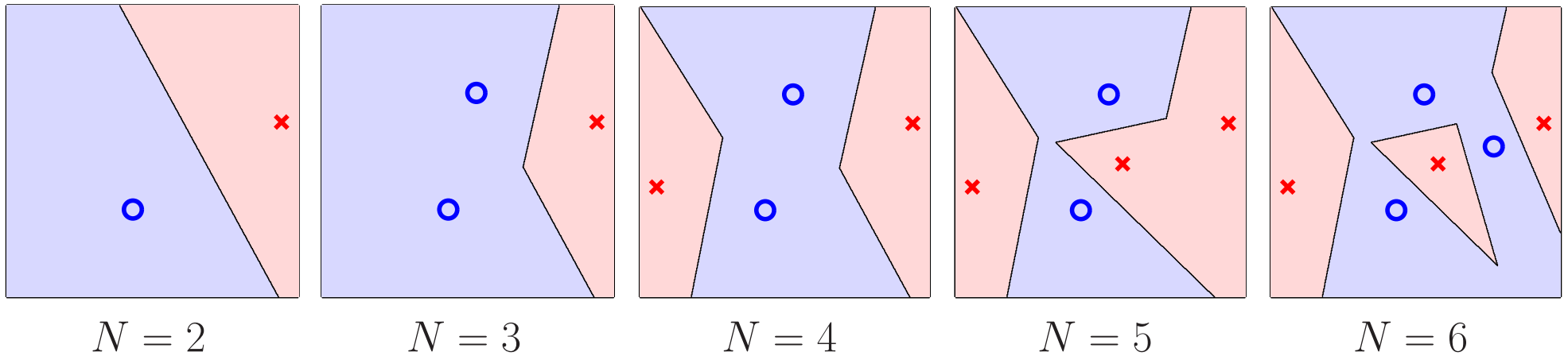
Assume $\pi(\mathbf{x})$ is continuous and $\mathbf{x}_{[1]} \xrightarrow{N \rightarrow \infty} \mathbf{x}$. Then $\pi(\mathbf{x}_{[1]}) \xrightarrow{N \rightarrow \infty} \pi(\mathbf{x})$.

$$\begin{aligned} \mathbb{P}[g_N(\mathbf{x}) \neq y] &= \mathbb{P}[y = +1, y_{[1]} = -1] + \mathbb{P}[y = -1, y_{[1]} = +1], \\ &= \pi(\mathbf{x}) \cdot (1 - \pi(\mathbf{x}_{[1]})) + (1 - \pi(\mathbf{x})) \cdot \pi(\mathbf{x}_{[1]}), \\ &\rightarrow \pi(\mathbf{x}) \cdot (1 - \pi(\mathbf{x})) + (1 - \pi(\mathbf{x})) \cdot \pi(\mathbf{x}), \\ &= 2\pi(\mathbf{x}) \cdot (1 - \pi(\mathbf{x})), \\ &\leq 2 \min\{\pi(\mathbf{x}), 1 - \pi(\mathbf{x})\}. \end{aligned}$$

The best you can do is

$$E_{\text{out}}^*(\mathbf{x}) = \min\{\pi(\mathbf{x}), 1 - \pi(\mathbf{x})\}.$$

Nearest Neighbor ‘Self-Regularizes’



A simple boundary is used with few data points.

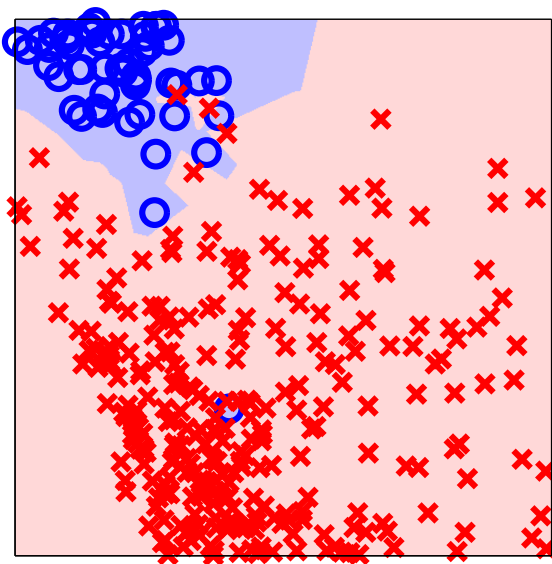
A more complicated boundary is possible *only* when you have more data points.

regularization guides you to simpler hypotheses when data quality/quantity is lower.

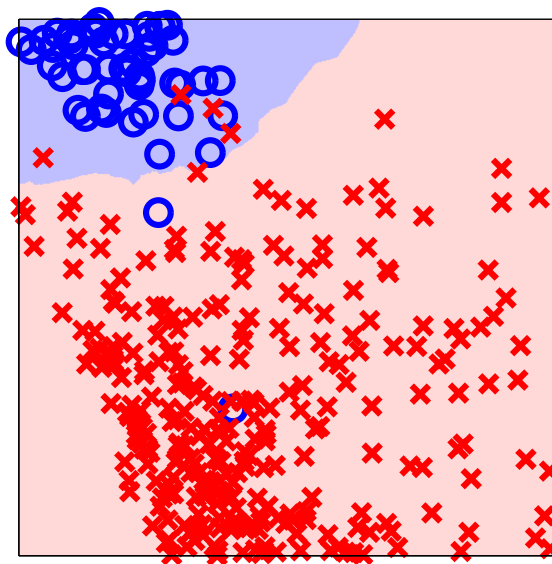
k -Nearest Neighbor

$$g(\mathbf{x}) = \text{sign} \left(\sum_{i=1}^k y_{[i]}(\mathbf{x}) \right).$$

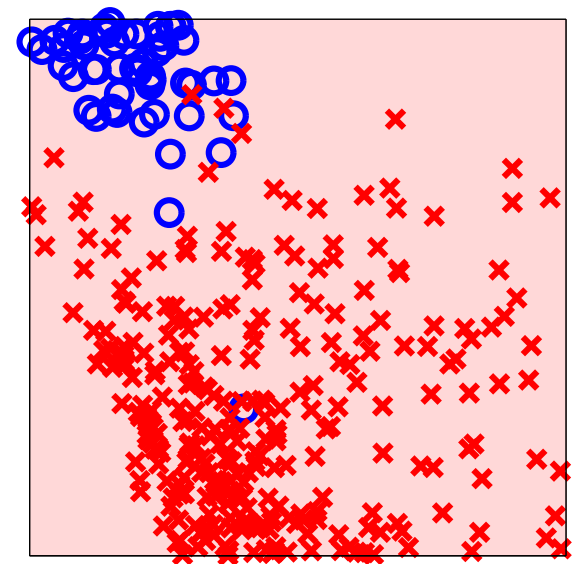
(k is odd and $y_n = \pm 1$).



1-NN rule



21-NN rule



127-NN rule

The Role of k

k determines the tradeoff between fitting the data and overfitting the data.

Theorem. For $N \rightarrow \infty$, if $k(N) \rightarrow \infty$ and $k(N)/N \rightarrow 0$ then,
$$E_{\text{in}}(g) \rightarrow E_{\text{out}}(g) \quad \text{and} \quad E_{\text{out}}(g) \rightarrow E_{\text{out}}^*.$$

For example $k = \lceil \sqrt{N} \rceil$.

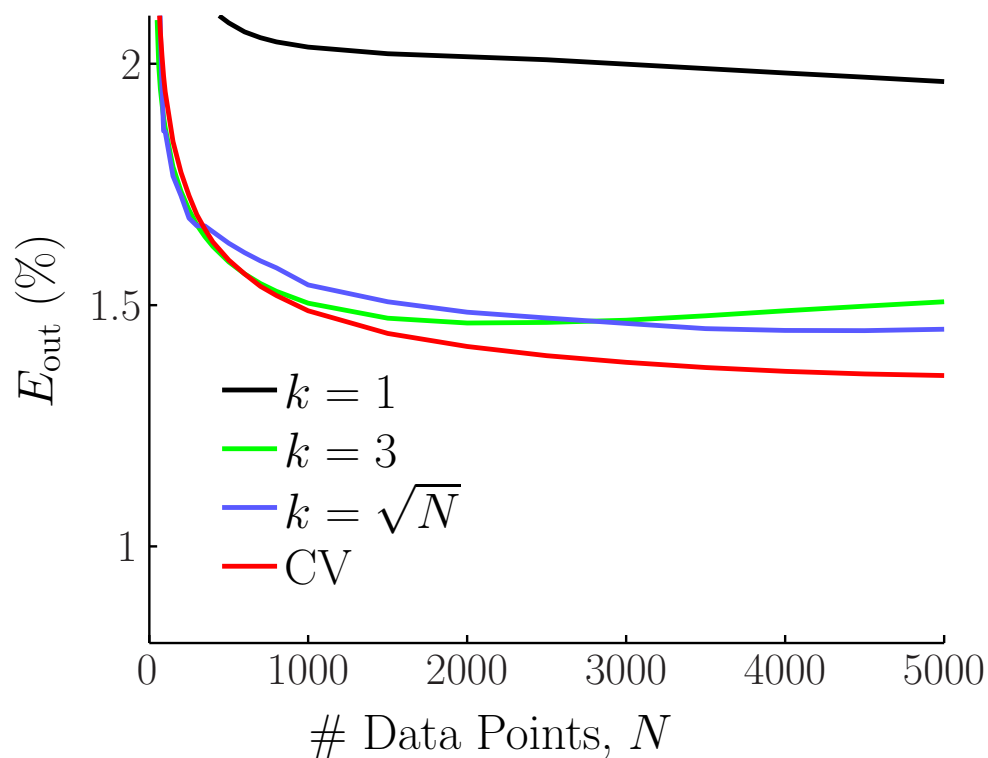
3 Ways To Choose k

1. $k = 3$.

2. $k = \lceil \sqrt{N} \rceil$.

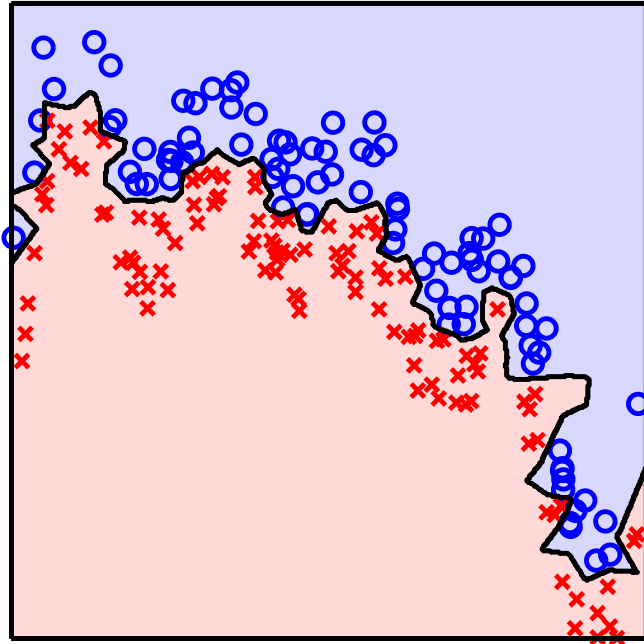
3. Validation or cross validation:

k -NN rule hypotheses g_k constructed on training set, tested on validation set, and best k is picked.



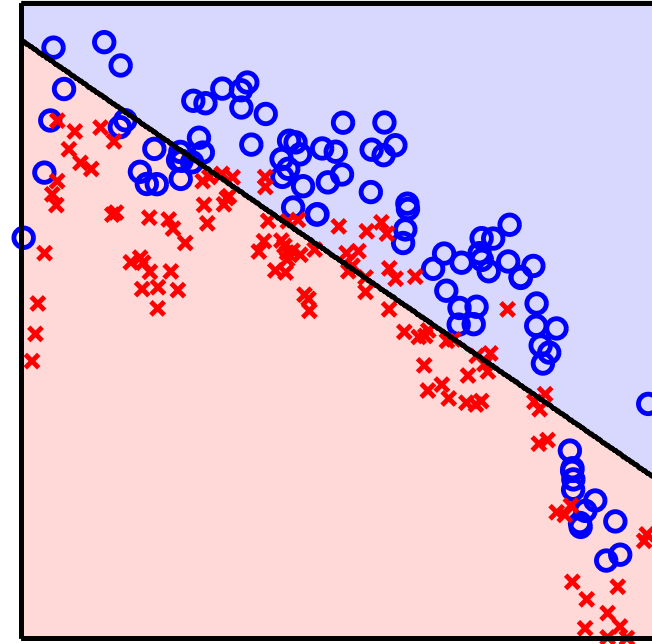
Nearest Neighbor is Nonparametric

NN-rule



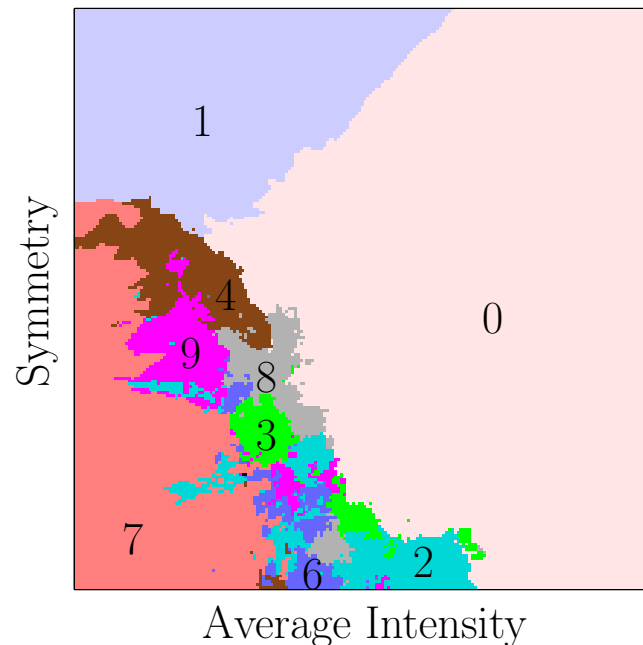
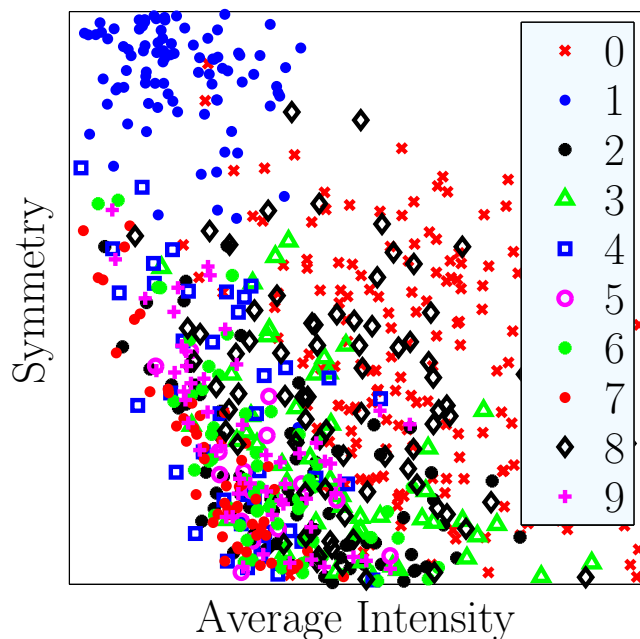
no parameters
expressive/flexible
 $g(\mathbf{x})$ needs data
generic, can model anything

Linear Model



$(d + 1)$ parameters
rigid, always linear
 $g(\mathbf{x})$ needs only weights
specialized

Nearest Neighbor Easily Extends to Multiclass



| True | Predicted | | | | | | | | | | |
|------|-------------|-------------|------------|----------|------------|------------|----------|----------|------------|----------|------|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
| 0 | 13.5 | 0.5 | 0.5 | 1 | 0 | 0.5 | 0 | 0 | 0.5 | 0 | 16.5 |
| 1 | 0.5 | 13.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 14 |
| 2 | 0.5 | 0 | 3.5 | 1 | 1 | 1.5 | 1 | 1 | 0 | 0.5 | 10 |
| 3 | 2.5 | 0 | 1.5 | 2 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 1 | 9.5 |
| 4 | 0.5 | 0 | 1 | 0.5 | 1.5 | 0.5 | 1 | 2 | 0 | 1.5 | 8.5 |
| 5 | 0.5 | 0 | 2.5 | 1 | 0.5 | 1.5 | 1 | 1 | 0 | 0.5 | 7.5 |
| 6 | 0.5 | 0 | 2 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 8.5 |
| 7 | 0 | 0 | 1.5 | 0.5 | 1.5 | 0.5 | 1 | 3 | 0 | 1 | 9 |
| 8 | 3.5 | 0 | 0.5 | 1 | 0.5 | 0.5 | 0.5 | 0 | 0.5 | 1 | 8 |
| 9 | 0.5 | 0 | 1 | 1 | 1 | 0.5 | 1 | 1 | 0.5 | 2 | 8.5 |
| | 22.5 | 14 | 14 | 9 | 7.5 | 7 | 7 | 9.5 | 2 | 8.5 | 100 |

41% accuracy!

Highlights of k -Nearest Neighbor

1. Simple.
2. No training.
3. Near optimal E_{out} .
4. Easy to justify classification to customer.
5. Can easily do multi-class.
6. Can easily adapt to regression or logistic regression

} A **good!** method

$$g(\mathbf{x}) = \frac{1}{k} \sum_{i=1}^k y_{[i]}(\mathbf{x})$$

$$g(\mathbf{x}) = \frac{1}{k} \sum_{i=1}^k \mathbb{I}[y_{[i]}(\mathbf{x}) = +1]$$

7. **Computationally demanding.** ← we will address this next

