K-Nearest Neighbor Approach KNN

Dr. Lourdes Peña-Castillo
Departments of Computer Science and Biology
Memorial University of Newfoundland

A simple (yet powerful) learning algorithm: K-nearest neighbor (KNN)

- Nearest-neighbor methods use those instances in the training set *T* closest in input space to an unseen instance *x* to predict *y.*
- “Closest” is determined by using a distance metric; e.g., Euclidean distance, Edit distance, etc.
- KNN is an instance-based approach and is considered lazy learning
  - No model is learned

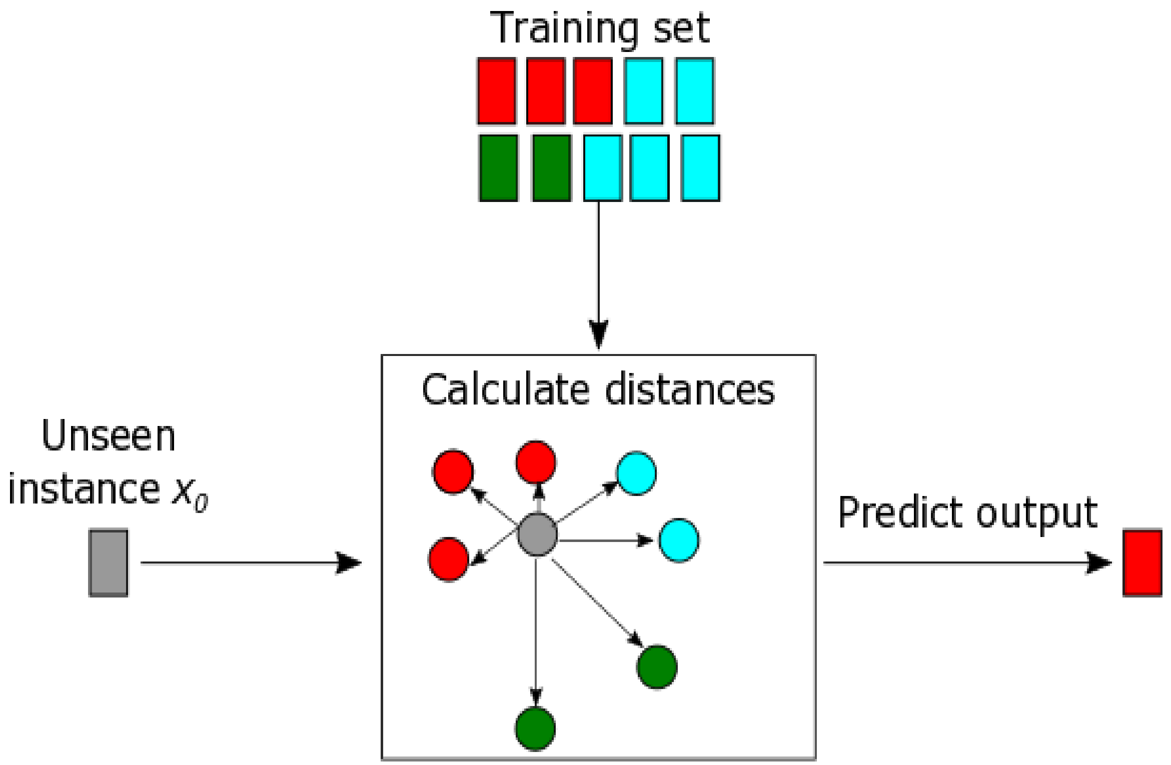
KNN Approach

1) Receive new instance x_0

2) Calculate x_0 distance to all instances in T

3) Select the *k* closest instances to x_0

4) Use these *k* instances to predict the output of x_0



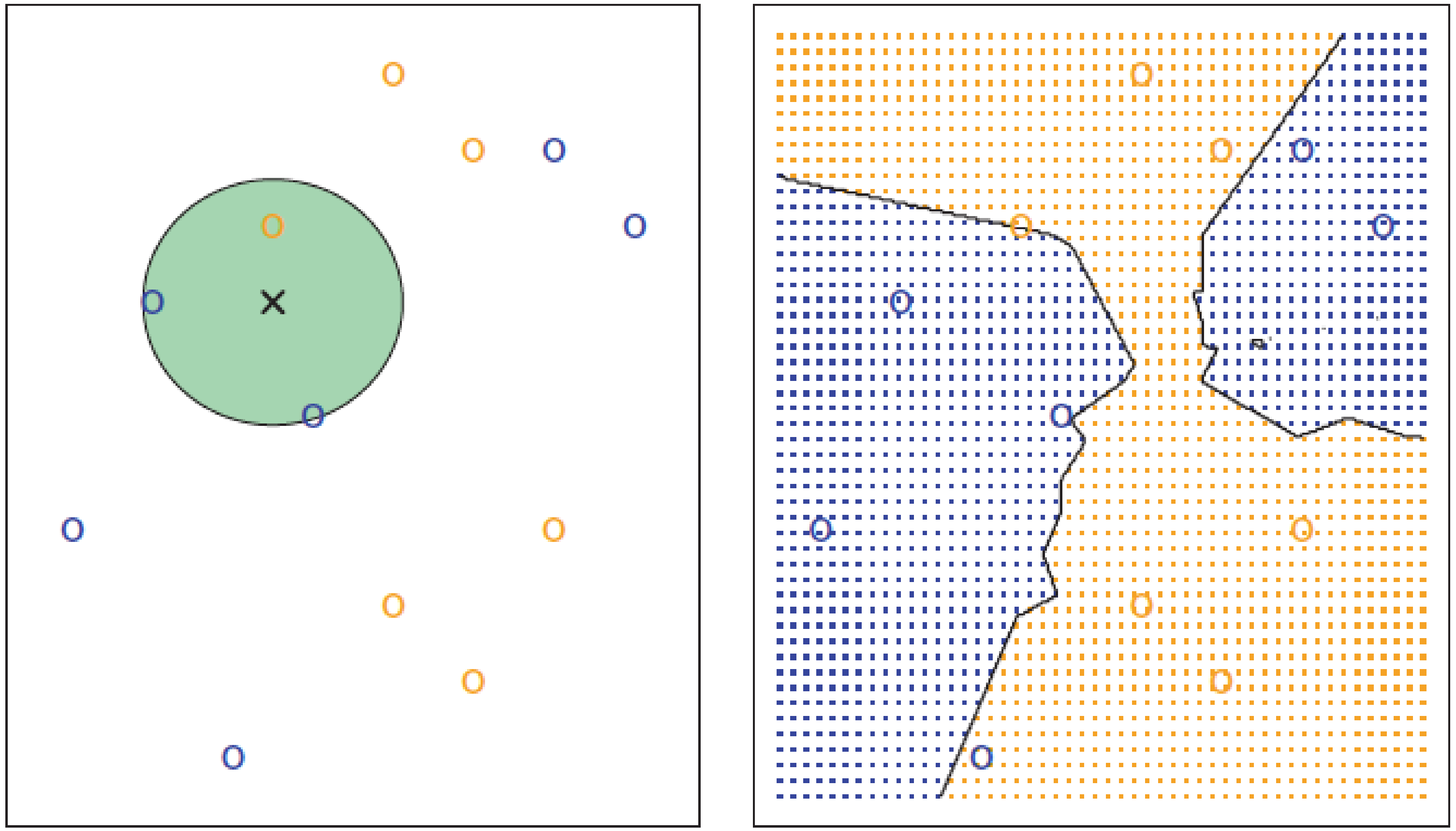


FIGURE 2.14. *The KNN approach, using $K = 3$, is illustrated in a simple situation with six blue observations and six orange observations.* Left: *a test observation at which a predicted class label is desired is shown as a black cross. The three closest points to the test observation are identified, and it is predicted that the test observation belongs to the most commonly-occurring class, in this case blue.* Right: *The KNN decision boundary for this example is shown in black. The blue grid indicates the region in which a test observation will be assigned to the blue class, and the orange grid indicates the region in which it will be assigned to the orange class.*

Fig. from *An Introduction to Statistical Learning* by James et al, 2013.

KNN Prediction

- For regression:
  - Find the *k* observations in T closest to x_0 and average their responses
- For classification:
  - Find the *k* observations in T closest to x_0 and take their most common class label (majority vote)

KNN Prediction

Regression

$\hat{Y}(x_0) = \frac{1}{k} \sum_{x_i \in N_k(x_0)} y_i$, where $N_k(x_0)$ is the neighborhood of x_0 defined by the k closest examples x_i in T.

Classification

$\hat{Y}(x_0) = max_j Pr(Y = j|X = x_0) = \frac{1}{k} \sum_{i \in N_k(x_0)} I(y_i = j))$, where I is an indicator function that equals 1 if $y_i = j$ and zero if $y_i \neq j$.

An Example

Training set

Day	Temperature	Outlook	Humidity	Windy	Play Golf?
07-05	hot	sunny	high	false	no
07-06	hot	sunny	high	true	no
07-07	hot	overcast	high	false	yes
07-09	cool	rain	normal	false	yes
07-10	cool	overcast	normal	true	yes
07-12	mild	sunny	high	false	no
07-14	cool	sunny	normal	false	yes
07-15	mild	rain	normal	false	yes
07-20	mild	sunny	normal	true	yes
07-21	mild	overcast	high	true	yes
07-22	hot	overcast	normal	false	yes
07-23	mild	rain	high	true	no
07-26	cool	rain	normal	true	no
07-30	mild	rain	high	false	yes

x_0

tomorrow	mild	sunny	normal	false

An Example

Training set

Day	Temperature	Outlook	Humidity	Windy	Play Golf?	Distance
07-05	hot	sunny	high	false	no	2
07-06	hot	sunny	high	true	no	3
07-07	hot	overcast	high	false	yes	3
07-09	cool	rain	normal	false	yes	2
07-10	cool	overcast	normal	true	yes	3
07-12	mild	sunny	high	false	no	1
07-14	cool	sunny	normal	false	yes	1
07-15	mild	rain	normal	false	yes	...
07-20	mild	sunny	normal	true	yes	
07-21	mild	overcast	high	true	yes	
07-22	hot	overcast	normal	false	yes	
07-23	mild	rain	high	true	no	
07-26	cool	rain	normal	true	no	
07-30	mild	rain	high	false	yes	

x_0

tomorrow	mild	sunny	normal	false

An Example

Day	Temperature	Outlook	Humidity	Windy	Play Golf?
07-05	hot	sunny	high	false	no
07-06	hot	sunny	high	true	no
07-07	hot	overcast	high	false	yes
07-09	cool	rain	normal	false	yes
07-10	cool	overcast	normal	true	yes
07-12	mild	sunny	high	false	no
07-14	cool	sunny	normal	false	yes
07-15	mild	rain	normal	false	yes
07-20	mild	sunny	normal	true	yes
07-21	mild	overcast	high	true	yes
07-22	hot	overcast	normal	false	yes
07-23	mild	rain	high	true	no
07-26	cool	rain	normal	true	no
12-30	mild	rain	high	false	yes

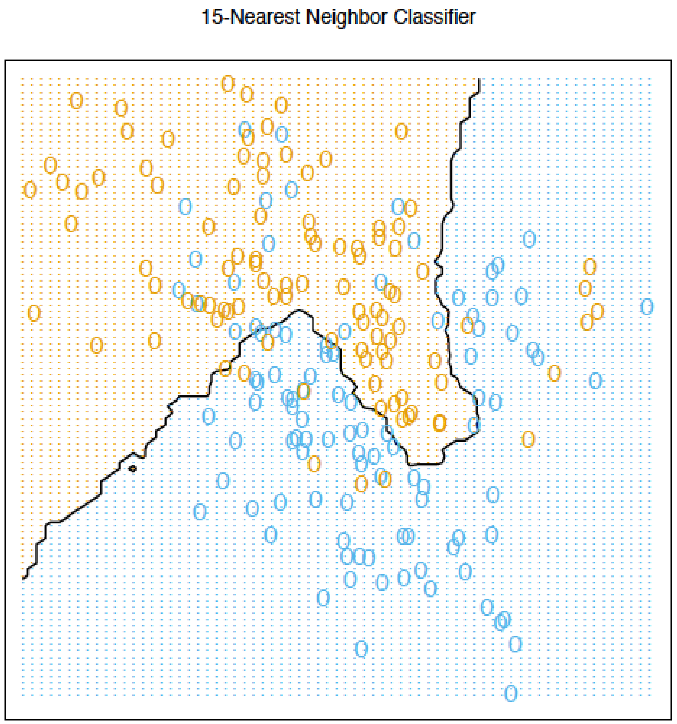
tomorrow	mild	sunny	normal	false	yes

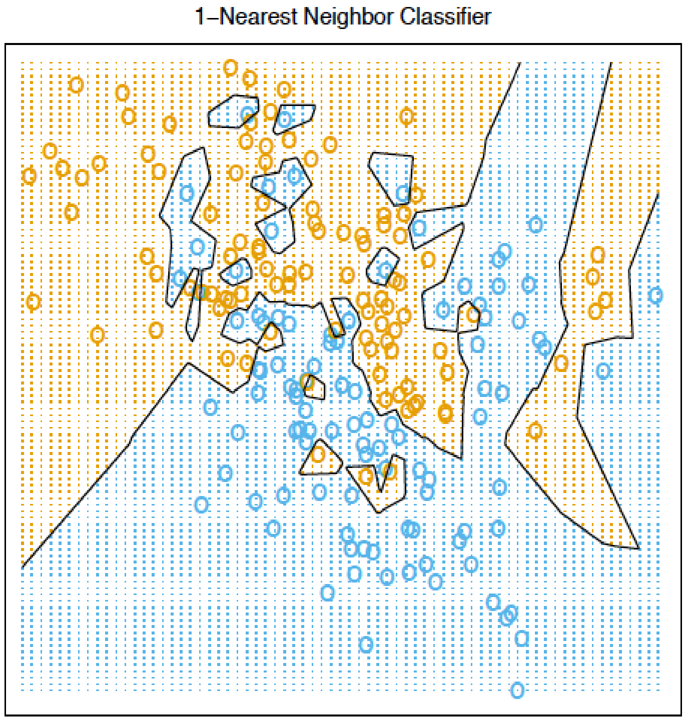
Let's assume k=4, $\Pr(Y=\text{yes}|X=x_0) = 0.75$

KNN

- KNN assumes f(x) is well approximated by a locally constant function
- The choice of *k* has a drastic effect on the KNN classifier obtained
- As *k* grows KNN becomes less flexible (variance decreases and bias increases)

A simple learning algorithm: k-nearest-neighbor (KNN)





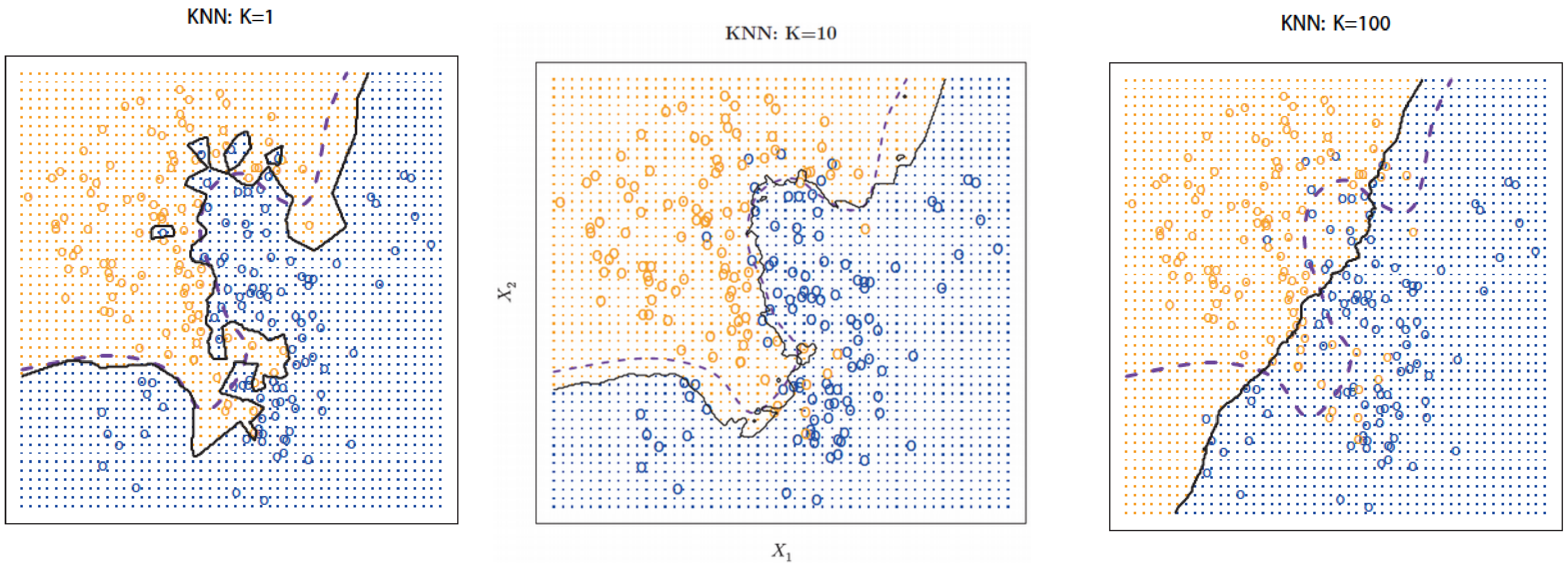
- Suppose we have simulated data where the outputs have the values BLUE or ORANGE and there are 100 points in each class.
- Figures show the results for 15-KNN and 1-KNN classification
- Note that decision boundaries are irregular and corresponds to local clusters where one class dominates.
- Which classifier fits the data more? Which one is more generalizable?

Figures from *The Elements of Statistical Learning* by Hastie, Tibshirani and Friedman, 2009.

Bayes Classifier

- The Bayes classifier assigns each observation to the most likely class, given its predictor values.
  - That is, assign x_0 to the class j for which the conditional probability $Pr(Y = j \mid X = x_0)$ is largest
- The Bayes classifier produces the lowest possible test error rate
- Computing the Bayes classifier is impossible if we do not know the conditional distribution of Y given X
- KNN attempts to estimate the conditional distribution of Y given X, and then classify a given observation to the class with the highest estimated probability

Choice of K



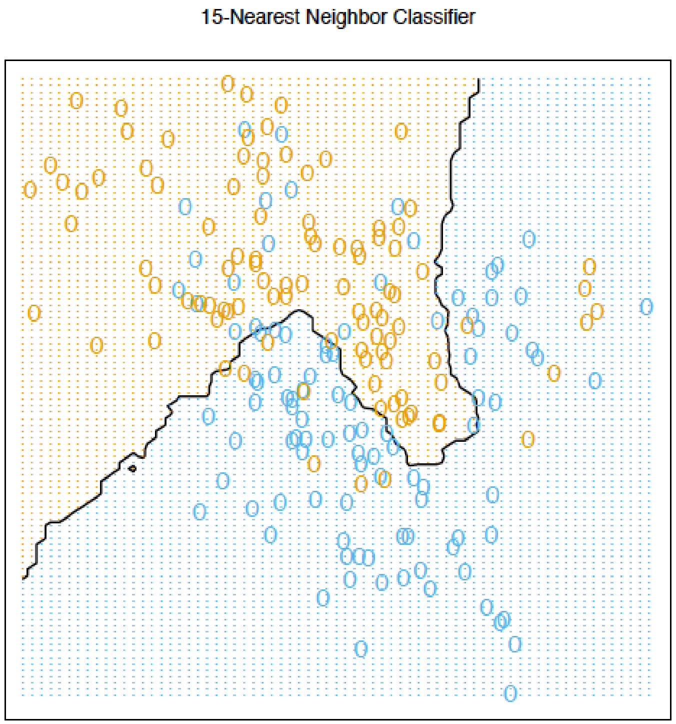
A comparison of the KNN decision boundaries obtained using various values for K.
With k=1, the decision boundary is too flexible (overfits the data).
With k=10, KNN decision boundary is very similar to the Bayes decision boundary (adequate capacity).
With k=100 is almost linear.

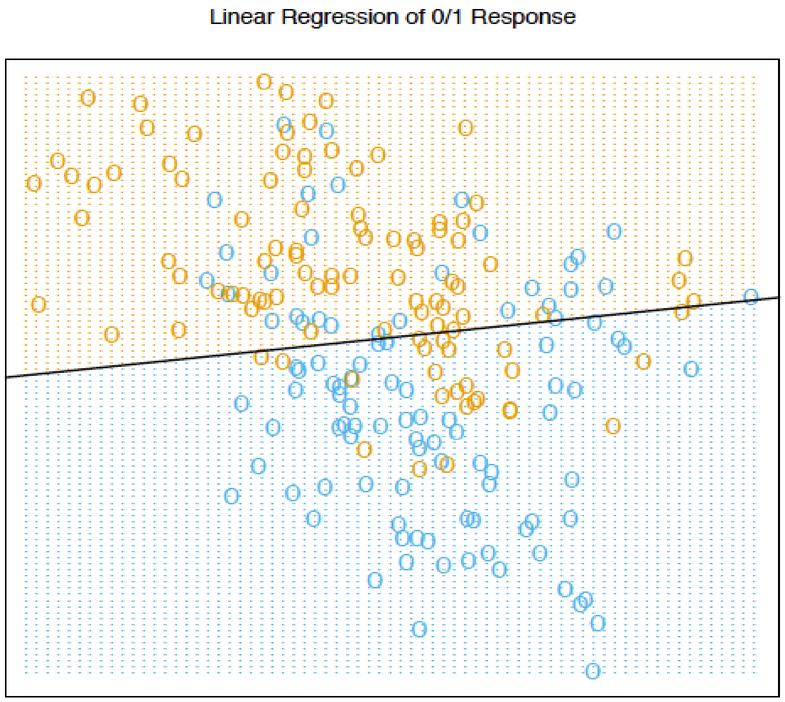
Fig. from *An Introduction to Statistical Learning* by James et al, 2013.

How to choose K?

- A good value can be found by evaluating various values with cross-validation on the training data
  - We will see cross-validation next

Comparing classifiers

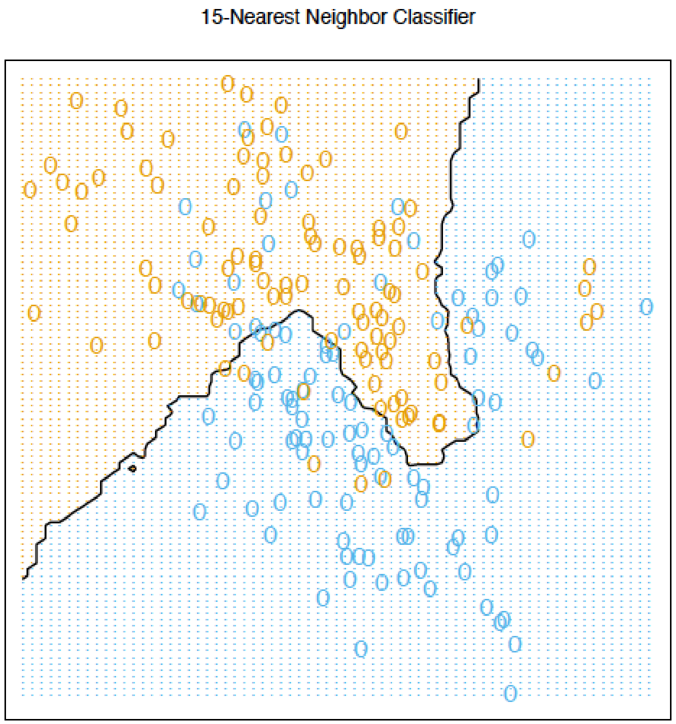


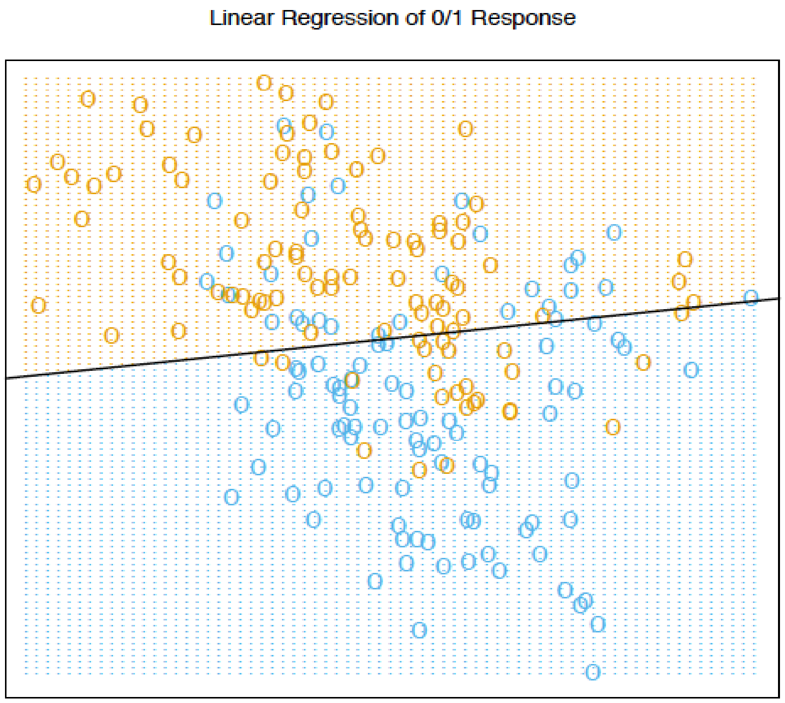


- Compare the classification of KNN with that of a linear classifier.
- The linear classifier has low variance and high inductive bias (it relies on a strong assumption about the data)
- The KNN classifier has high variance and low inductive bias

Figures from *The Elements of Statistical Learning* by Hastie, Tibshirani and Friedman, 2009.

Comparing classifiers





- Which classifier would be better for data generated from two Gaussian distributions with different means?
- Which classifier would be better for data generated from a mixture of 10 low-variance Gaussian distributions with different means?

Figures from *The Elements of Statistical Learning* by Hastie, Tibshirani and Friedman, 2009.

Improving KNN

- Use a distance-based voting scheme
  - Closer neighbors have more influence

$$\hat{y} = \frac{\sum_{i=1}^{k} w_i \cdot y_i}{\sum_{i=1}^{k} w_i}$$

$$w_i = \frac{1}{d(x_i, x)^2}$$

Improving KNN

- Normalize attributes
  - Different attributes are measured on different scales
  - If attribute values are approx. uniformly distributed, one can normalized values in [0,1]:

    $x'_{ip} = (x_{ip} – \min x_{jp}) / (\max x_{jp} – \min x_{jp})$

  - For other distributions, other normalizations might be more suitable. For example, logarithmic transformation

Improving KNN

- Choose the right distance function.
- A popular choice:

Euclidean Distance:

$$d(x_1, x_2) = \sqrt{\sum_{z=1}^{p} d(x_{1,z}, x_{2,z})^2}$$

If $x_{1,z}$ and $x_{2,z}$ are numerical then

$$d(x_{1,z}, x_{2,z}) = |x_{1,z} - x_{2,z}|$$

If $x_{1,z}$ and $x_{2,z}$ are categorical then

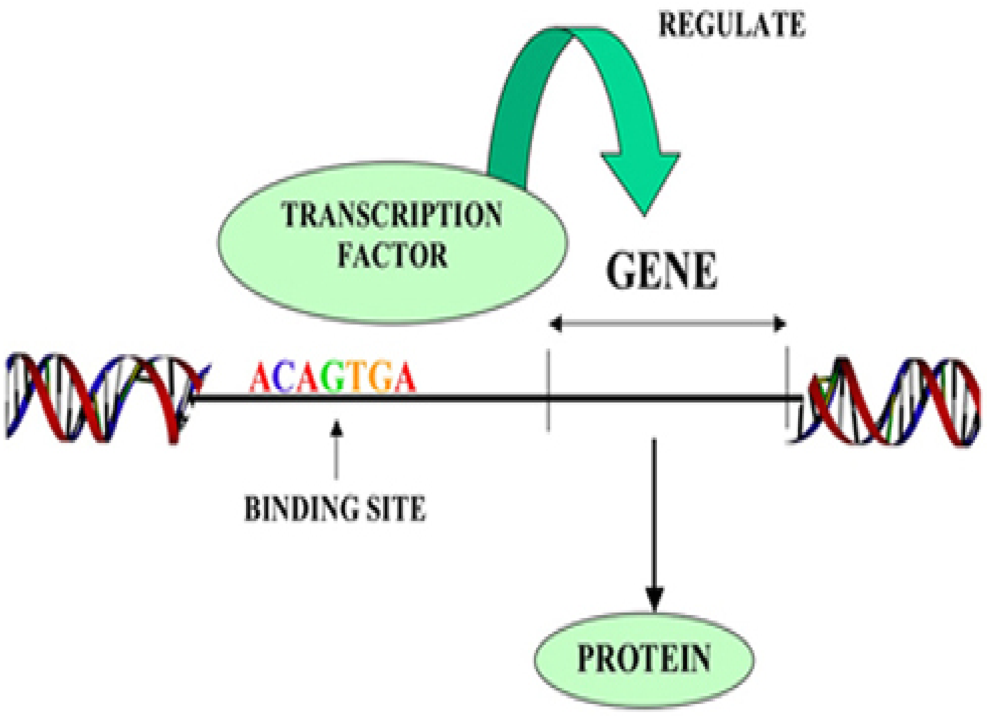
$$d(x_{1,z}, x_{2,z}) = \begin{cases} 0 \text{ if } x_{1,z} = x_{2,z} \\ 1 \text{ if } x_{1,z} \neq x_{2,z} \end{cases}$$

Improving KNN

- Other possible distances:
  - Manhattan or City-block distance
  - 1- correlation coefficient (Pearson or Spearman)
  - Edit distance (for strings)
- Keep in mind that:
  - distances are domain-dependent
  - need to be chosen apropriately

A sample application of KNN: Predicting sequence preferences of transcription factors

- Transcription factors are proteins that bind DNA and regulate transcription of other genes
- Each transcription factor binds to a certain sequence (string pattern)
- To understand transcriptional regulation we would like to predict binding preferences of transcription factors



A sample application of KNN

- Each training instance consists of

    (Amino acid sequence,

    sequence preference profile)

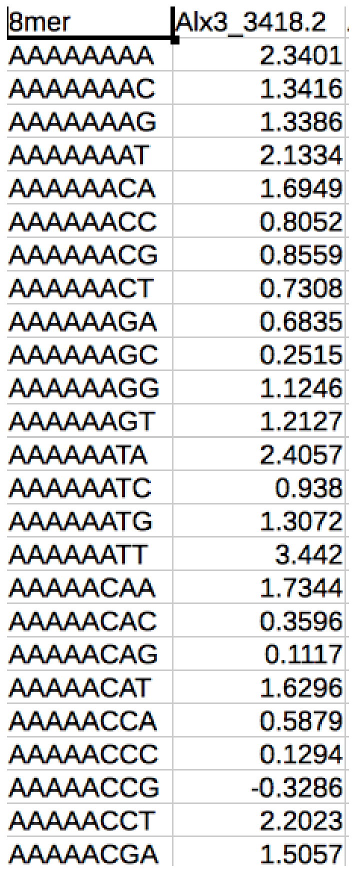
  of one transcription factor
- Goal:
  - Predict sequence preference profile of an unseen transcription factor based on its amino acid sequence

What kind of learning task is this?

x

KPKRQMKTPFQLETLEKVYSEEKYPSEATRAELSEKLDLSDRQLQMWFCHRRLKDKK

y



8mer	Alx3_3418.2
AAAAAAAA	2.3401
AAAAAAAC	1.3416
AAAAAAAG	1.3386
AAAAAAAT	2.1334
AAAAAACA	1.6949
AAAAAACC	0.8052
AAAAAACG	0.8559
AAAAAACT	0.7308
AAAAAAGA	0.6835
AAAAAAGC	0.2515
AAAAAAGG	1.1246
AAAAAAGT	1.2127
AAAAAATA	2.4057
AAAAAATC	0.938
AAAAAATG	1.3072
AAAAAATT	3.442
AAAAACAA	1.7344
AAAAACAC	0.3596
AAAAACAG	0.1117
AAAAACAT	1.6296
AAAAACCA	0.5879
AAAAACCC	0.1294
AAAAACCG	-0.3286
AAAAACCT	2.2023
AAAAACGA	1.5057

Calculating Distances

- Number of non-identical characters over a set of 15 DNA-contacting amino acids. These 15 residues are 3, 5, 6, 25, 31, 44, 46, 47, 48, 50, 51, 53, 54, 55 and 57
- For example,

        012345678901234567890123456789012345678901234567890123456

Alx3 RRNRTTFSTFQLEELEKVFQKTHYPDVYAREQLALRTDLTEARVQVWFQNRRAKWRK

Alx4 RRNRTTF**TSY**QLEELEKVFQKTHYPDVYAREQLA**M**RTDLTEARVQVWFQNRRAKWRK

- Distance = 0

A

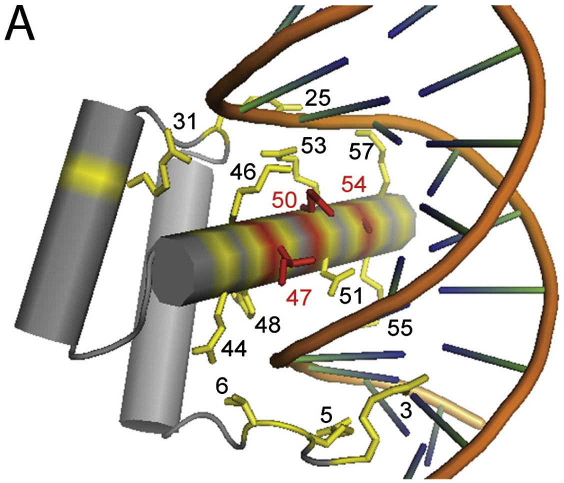


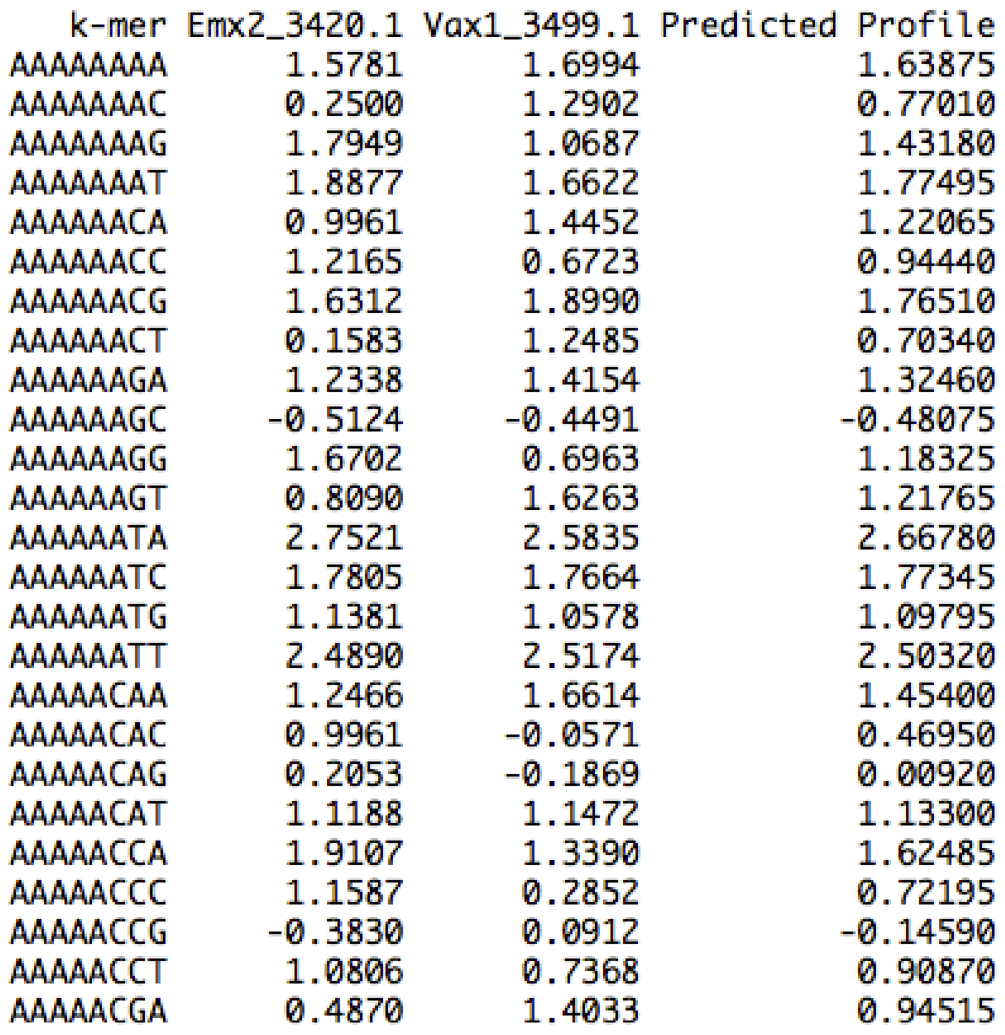
Figure from M. Berger *et al*., Cell 133, 1266-1276, June 27, 2008.

Modifications Done

- Take all the nearest neighbors without randomly selecting some of them

Predicting Profile

Suppose an unseen TF has as nearest neighbors: Emx2 and Vax1, then its predicted profile is the average of their profiles.

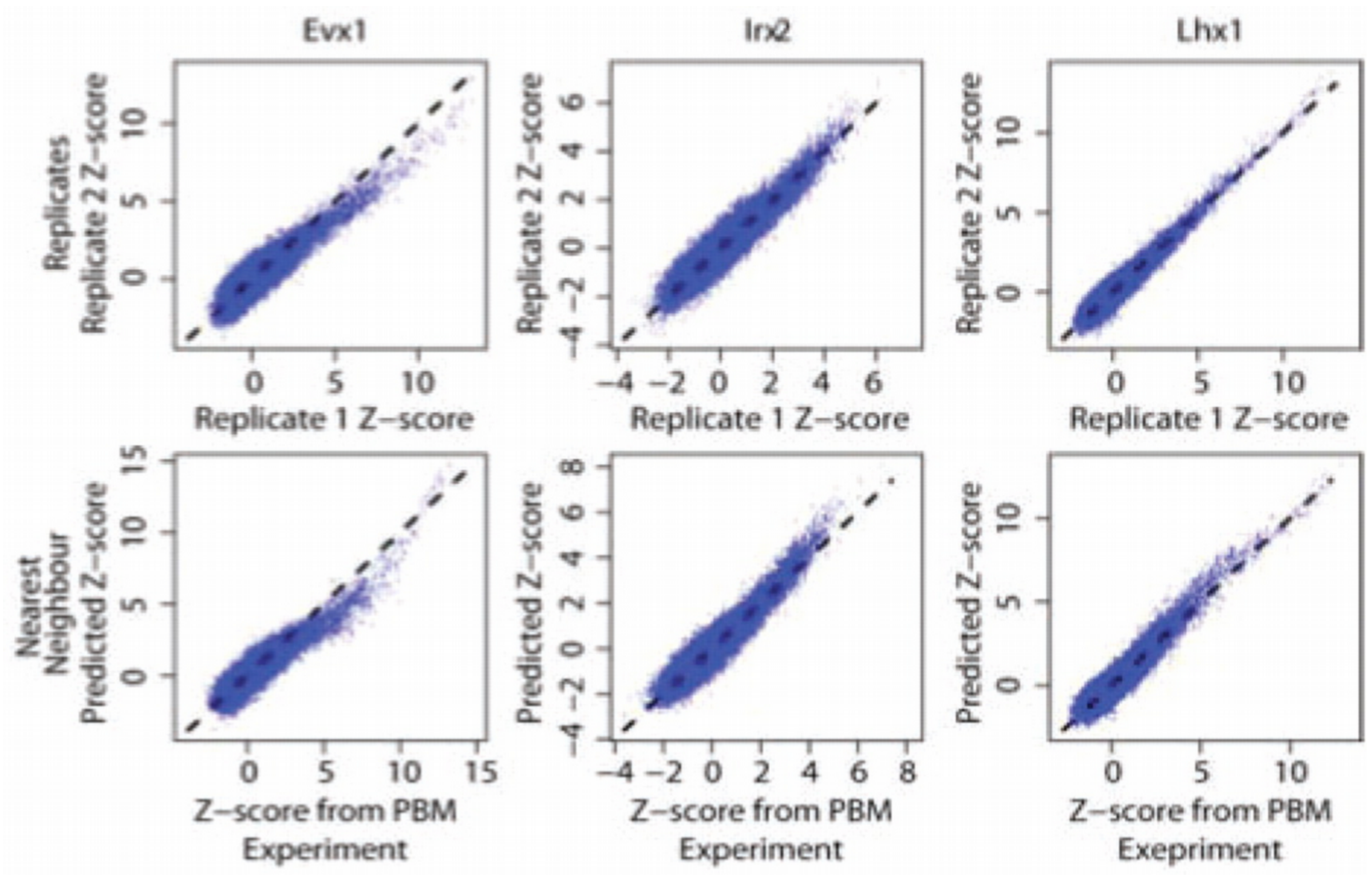


```
   k-mer Emx2_3420.1 Vax1_3499.1 Predicted Profile
AAAAAAAA      1.5781      1.6994           1.63875
AAAAAAAC      0.2500      1.2902           0.77010
AAAAAAAG      1.7949      1.0687           1.43180
AAAAAAAT      1.8877      1.6622           1.77495
AAAAAACA      0.9961      1.4452           1.22065
AAAAAACC      1.2165      0.6723           0.94440
AAAAAACG      1.6312      1.8990           1.76510
AAAAAACT      0.1583      1.2485           0.70340
AAAAAAGA      1.2338      1.4154           1.32460
AAAAAAGC     -0.5124     -0.4491          -0.48075
AAAAAAGG      1.6702      0.6963           1.18325
AAAAAAGT      0.8090      1.6263           1.21765
AAAAAATA      2.7521      2.5835           2.66780
AAAAAATC      1.7805      1.7664           1.77345
AAAAAATG      1.1381      1.0578           1.09795
AAAAAATT      2.4890      2.5174           2.50320
AAAAACAA      1.2466      1.6614           1.45400
AAAAACAC      0.9961     -0.0571           0.46950
AAAAACAG      0.2053     -0.1869           0.00920
AAAAACAT      1.1188      1.1472           1.13300
AAAAACCA      1.9107      1.3390           1.62485
AAAAACCC      1.1587      0.2852           0.72195
AAAAACCG     -0.3830      0.0912          -0.14590
AAAAACCT      1.0806      0.7368           0.90870
AAAAACGA      0.4870      1.4033           0.94515
```

….

How to evaluate performance?

- Three performance metrics were used:
  - RMSE (root mean square error)
  - The number of top-100 8-mers in common
  - Spearman correlation value over the complete sequence preference profile



From: Predicting the binding preference of transcription factors to individual DNA k-mers

Bioinformatics. 2008;25(8):1012-1018. doi:10.1093/bioinformatics/btn645

How KNN compare with other ML approaches?

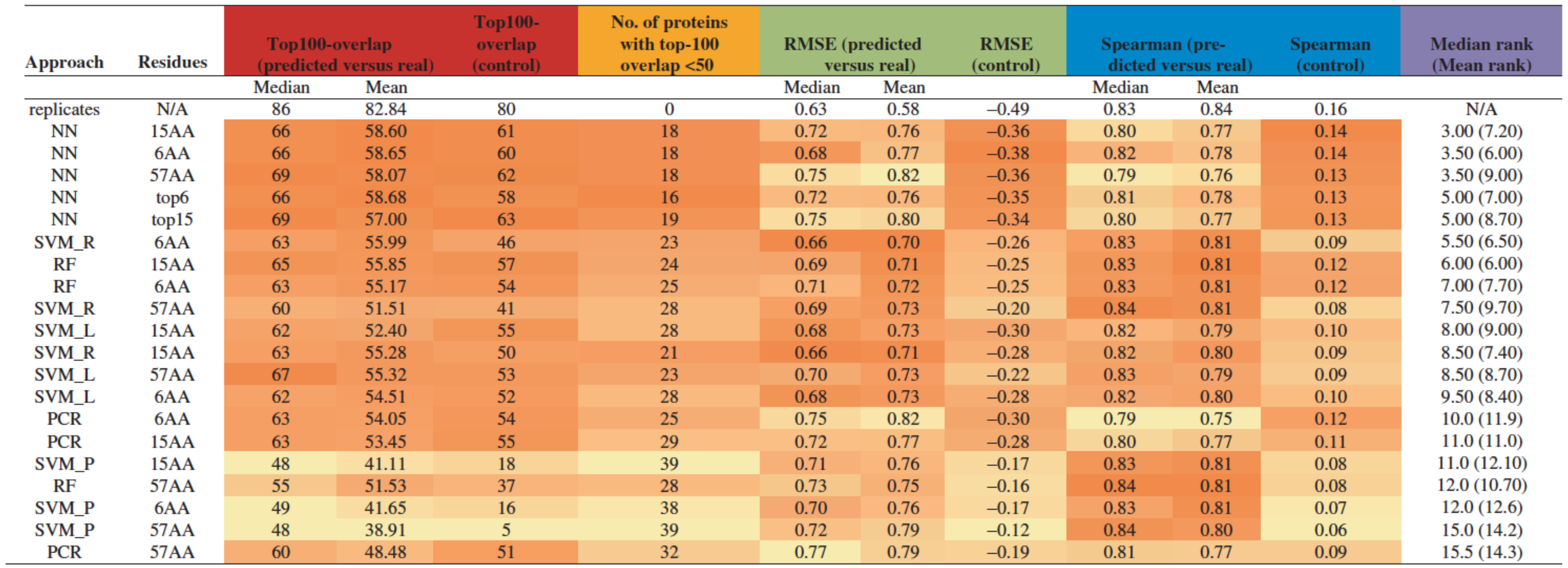


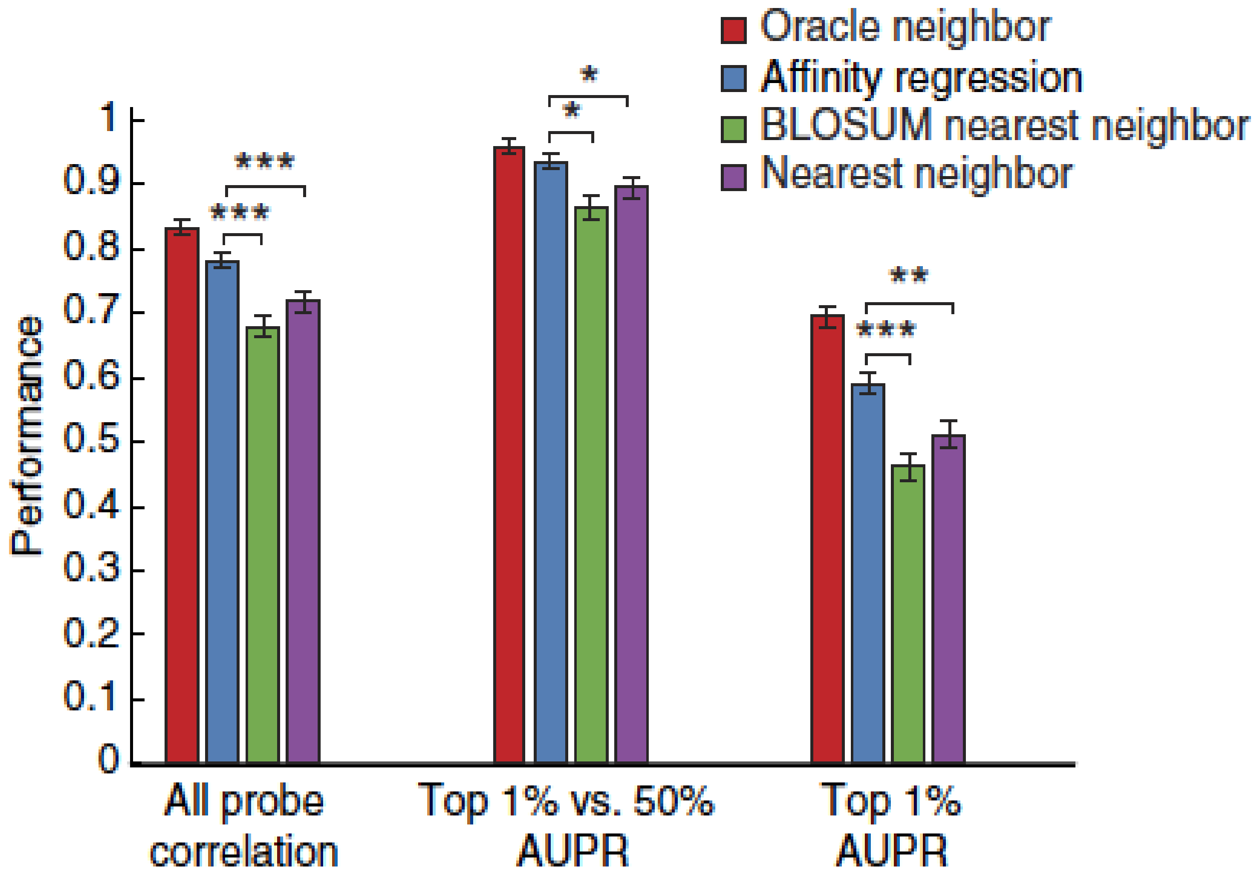
Table 2. Leave-one-out cross-validation measures for 8mer *Z*-score profile prediction algorithms on 32 896 8mers for 75 homeodomains

Approach	Residues	Top100-overlap (predicted versus real)		Top100-overlap (control)	No. of proteins with top-100 overlap <50	RMSE (predicted versus real)		RMSE (control)	Spearman (predicted versus real)		Spearman (control)	Median rank (Mean rank)
		Median	Mean			Median	Mean		Median	Mean		
replicates	N/A	86	82.84	80	0	0.63	0.58	–0.49	0.83	0.84	0.16	N/A
NN	15AA	66	58.60	61	18	0.72	0.76	–0.36	0.80	0.77	0.14	3.00 (7.20)
NN	6AA	66	58.65	60	18	0.68	0.77	–0.38	0.82	0.78	0.14	3.50 (6.00)
NN	57AA	69	58.07	62	18	0.75	0.82	–0.36	0.79	0.76	0.13	3.50 (9.00)
NN	top6	66	58.68	58	16	0.72	0.76	–0.35	0.81	0.78	0.13	5.00 (7.00)
NN	top15	69	57.00	63	19	0.75	0.80	–0.34	0.80	0.77	0.13	5.00 (8.70)
SVM_R	6AA	63	55.99	46	23	0.66	0.70	–0.26	0.83	0.81	0.09	5.50 (6.50)
RF	15AA	65	55.85	57	24	0.69	0.71	–0.25	0.83	0.81	0.12	6.00 (6.00)
RF	6AA	63	55.17	54	25	0.71	0.72	–0.25	0.83	0.81	0.12	7.00 (7.70)
SVM_R	57AA	60	51.51	41	28	0.69	0.73	–0.20	0.84	0.81	0.08	7.50 (9.70)
SVM_L	15AA	62	52.40	55	28	0.68	0.73	–0.30	0.82	0.79	0.10	8.00 (9.00)
SVM_R	15AA	63	55.28	50	21	0.66	0.71	–0.28	0.82	0.80	0.09	8.50 (7.40)
SVM_L	57AA	67	55.32	53	23	0.70	0.73	–0.22	0.83	0.79	0.09	8.50 (8.70)
SVM_L	6AA	62	54.51	52	28	0.68	0.73	–0.28	0.82	0.80	0.10	9.50 (8.40)
PCR	6AA	63	54.05	54	25	0.75	0.82	–0.30	0.79	0.75	0.12	10.0 (11.9)
PCR	15AA	63	53.45	55	29	0.72	0.77	–0.28	0.80	0.77	0.11	11.0 (11.0)
SVM_P	15AA	48	41.11	18	39	0.71	0.76	–0.17	0.83	0.81	0.08	11.0 (12.10)
RF	57AA	55	51.53	37	28	0.73	0.75	–0.16	0.84	0.81	0.08	12.0 (10.70)
SVM_P	6AA	49	41.65	16	38	0.70	0.76	–0.17	0.83	0.81	0.07	12.0 (12.6)
SVM_P	57AA	48	38.91	5	39	0.72	0.79	–0.12	0.84	0.80	0.06	15.0 (14.2)
PCR	57AA	60	48.48	51	32	0.77	0.79	–0.19	0.81	0.77	0.09	15.5 (14.3)

Algorithms are sorted in descending order of median rank across all columns, where ties are resolved using mean rank. The first row shows the agreement between 19 experimental replicates and their corresponding true *Z*-score profiles as measured using PBM. Columns labelled 'predicted versus real' show the mean or median performance between each predicted profile and its true, measured *Z*-score profile. Columns labelled 'control' show the difference between the median predicted versus real performance and the median of the performance between all pairs of predicted and actual profiles. Cells in a given column are coloured according to their position in the range of that column. Rows labelled top6 and top15 represent the result obtained if we use the 6 and 15 most important amino acid positions according to the RF importance score on the 57AA set.

From: Alleyne et al, Bioinformatics. 2008;25(8):1012-1018. doi:10.1093/bioinformatics/btn645

How KNN compare with other ML approaches?



From: Pelossof et al, Nature Biotechnology, 33(12). 2015

By now, you should be able to

- explain how KNN works
- implement KNN for regression and classification
- understand the effect of the value of k in the performance of KNN
- define the Bayes classifier and decision boundary
- have an insight into the kind of function which is well approximated by KNN
- know strategies to improve KNN
- describe a bioinformatic application of KNN