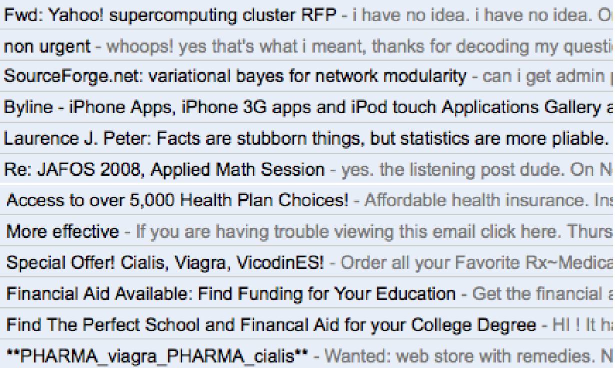
Learning to Generalize

Jake Hofman

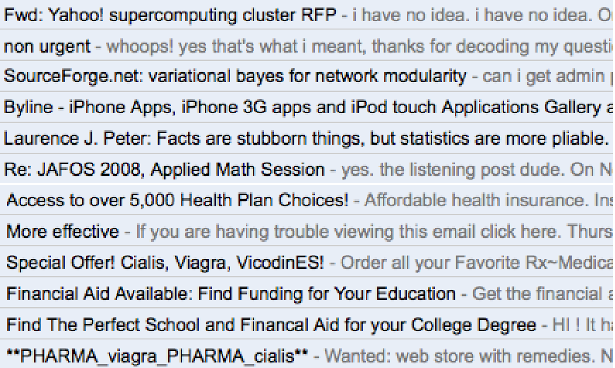
Microsoft Research

March 16, 2018

Learning by example



Learning by example



- How did you solve this problem?
- Can you make this process explicit (e.g. write code to do so)?

Machine Learning: A Definition

The study of algorithms that enable machines to learn from experience and improve over time

http://www.cs.cmu.edu/~tom/

Everything old is new again[1]

- Many fields ...
  - Statistics
  - Pattern recognition
  - Data mining
  - Machine learning
- ... similar goals
  - Extract and recognize patterns in data
  - Interpret or explain observations
  - Test validity of hypotheses
  - Efficiently search the space of hypotheses
  - Design efficient algorithms enabling machines to learn from data

[1] http://bit.ly/mloldnew

Statistics vs. machine learning[2]

Glossary

Machine learning	Statistics
network, graphs	model
weights	parameters
learning	fitting
generalization	test set performance
supervised learning	regression/classification
unsupervised learning	density estimation, clustering
large grant = \$1,000,000	large grant= \$50,000
nice place to have a meeting: Snowbird, Utah, French Alps	nice place to have a meeting: Las Vegas in August

[2] `http://bit.ly/statvsml`

Example results: machine learning[3]

Add an asymmetric frequency feature $\mathbf{y}^{(3)}_{j,f_{ut}}$: **SBRAMF-UTB-UTF-MTF-ATF-MFF-AFF**

$$\widehat{r_{uit}} = \mu_i + \mu_u + \mu_{u,t} + \mu_{i,\text{bin}(t)} + \left(\mathbf{p}_i^{(1)} + \mathbf{p}_{i,\text{bin}(t)}^{(2)} + \mathbf{p}_{i,f_{ut}}^{(3)}\right)^T \left(\mathbf{q}_u^{(1)} + \mathbf{q}_{u,t}^{(2)} + \frac{1}{\sqrt{|N(u)|}} \sum_{j \in N(u)} \left(\mathbf{y}_j^{(1)} + \mathbf{y}_{j,\text{bin}(t)}^{(2)} + \mathbf{y}_{j,f_{ut}}^{(3)}\right)\right) \tag{34}$$

Model extension (+)	epoch time	#epochs	probeRMSE, $k = 50$ features
SBRMF - SVD with biases	17[s]	69	0.9054
SBRAMF - asymmetric part	50[s]	30	0.8974
+**UTB** - user time bias	61[s]	50	0.8919
+**UTF** - user time feature	62[s]	38	0.8911
+**MTF** - movie time feature	74[s]	37	0.8908
+**ATF** - asymmetric time feature	74[s]	44	0.8905
+**MFF** - movie frequency feature	149[s]	46	0.8900
+**AFF** - asymmetric frequency feature	206[s]	45	0.8886 (0.8846 with $k = 1000$)

[3] Bell, Koren & Volinsky, 2008

Example results: social science[4]

Table 2: Relationship of clicks/added revisions and time dummies for direct neighbors of shocked articles in the 'featured articles' condition.

	clicks			Δ revisions		
	(1) compare control	(2) comp. placebo	(3) before after	(4) comp. control	(5) comp. placebo	(6) before after
Before: days - 7 to -4	0.821	0.486	-0.181	-0.000	0.002	0.001
	(1.342)	(1.222)	(1.066)	(0.004)	(0.004)	(0.003)
Before: days - 3 to -1	1.811	2.026	1.910	0.000	-0.001	-0.001
	(2.408)	(1.990)	(1.462)	(0.005)	(0.004)	(0.003)
t = 0	28.546***	34.577***	31.603***	0.030***	0.033***	0.030***
	(5.948)	(5.731)	(5.573)	(0.009)	(0.008)	(0.007)
t = 1	1.632	1.535	0.974	0.007	0.006	0.005
	(2.146)	(2.318)	(1.565)	(0.007)	(0.007)	(0.005)
t = 2	-0.569	-1.189	0.028	-0.013*	-0.011	-0.007*
	(2.768)	(2.395)	(1.910)	(0.007)	(0.007)	(0.004)
After: days 3 to 6	-2.170	-0.376	-0.531	0.002	-0.000	-0.001
	(2.052)	(2.296)	(1.359)	(0.004)	(0.004)	(0.003)
After: days 7 to 14	-0.639	0.207	0.207	0.001	-0.001	0.001
	(2.593)	(2.794)	(1.953)	(0.007)	(0.007)	(0.005)
Time Dummies	Yes	Yes	No	Yes	Yes	No
Mean dep. Variable	36.208	36.559	37.276	0.045	0.045	0.047
Observations	346104	371382	186384	346104	371382	186384
Number of Pages	15732	16881	8472	15732	16881	8472
Adj. R^2	0.002	0.003	0.004	0.000	0.000	0.000

[4] Krummer, 2015

Social science vs. machine learning?[5]

Machine Learning		Social science
$\hat{y}$	~~vs~~ **and**	$\hat{\beta}$
Predict		Explain

Important to view prediction and explanation as compliments, not substitutes

[5] Mullainathan & Spiess, JEP 2017

Why Machine Learning?[6]

- Prediction is often useful in its own right
  e.g., forecasting economic outcomes, quantifying risk
- Prediction can help fill in missing data
  e.g., inferring online demographics
- Prediction can aid in causal inference
  e.g., matching, instrumental variables, heterogenous effects
- Predictive models can provide benchmarks for causal theories
  e.g., gap in performance indicates work to be done

[6] Mullainathan & Spiess, JEP 2017

Outline for the day

What we'll cover:

Concepts:

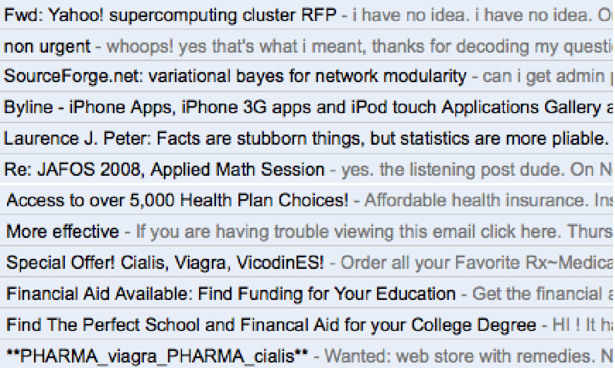
- Generalization error
- Overfitting
- Cross-validation
- Regularization

Methods:

- k-nearest neighbors
- Naive Bayes
- Ridge regression
- Gradient descent

What we won't cover:
tree-based methods, deep neural nets, ensemble methods, unsupervised learning, reinforcement learning, . . .

Learning by example



- How did you solve this problem?
- Can you make this process explicit (e.g. write code to do so)?

Roadmap?

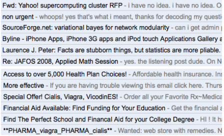
Step 1: Have data

Step 2: ???

Step 3: Profit

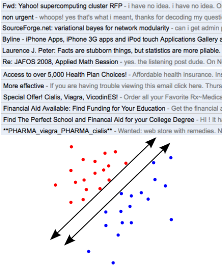
Roadmap, take two

1. Get data



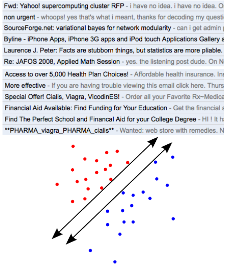
Roadmap, take two

1. Get data
2. Visualize/perform sanity checks
3. Clean/filter observations
4. Choose features to represent data



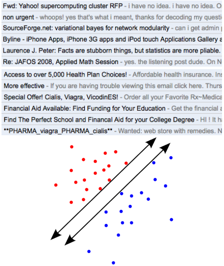
Roadmap, take two

1. Get data
2. Visualize/perform sanity checks
3. Clean/filter observations
4. Choose features to represent data
5. Specify model
6. Specify loss function



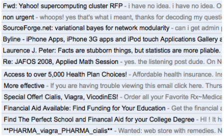
Roadmap, take two

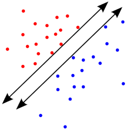
1. Get data
2. Visualize/perform sanity checks
3. Clean/filter observations
4. Choose features to represent data
5. Specify model
6. Specify loss function
7. Develop algorithm to minimize loss

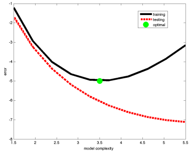


Roadmap, take two

1. Get data
2. Visualize/perform sanity checks
3. Clean/filter observations
4. Choose features to represent data
5. Specify model
6. Specify loss function
7. Develop algorithm to minimize loss
8. Choose performance measure
9. "Train" to minimize loss
10. "Test" to evaluate generalization







Themes

Fitting models is a relatively small piece of the pipeline

Themes

Cleaning and normalizing data is a substantial amount of the work (and likely impacts results)

Themes

The features you choose to represent the data often matter more than the algorithm that learns from it

Themes

There's a skill in matching tools (e.g., models and algorithms) to problems

Themes

Our models should be complex enough to explain the past, but simple enough to generalize to the future

Themes

Bigger models $\neq$ Better models

Themes

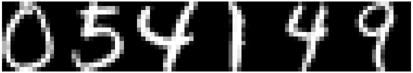
Simple methods (e.g., linear models) work surprisingly well, especially with lots of data

Themes

Even with simple methods, there are many decisions to be made in developing a successful machine learning solution

Digit recognition

Classification is an *supervised* learning task by which we aim to *predict the correct label* for an example given its features



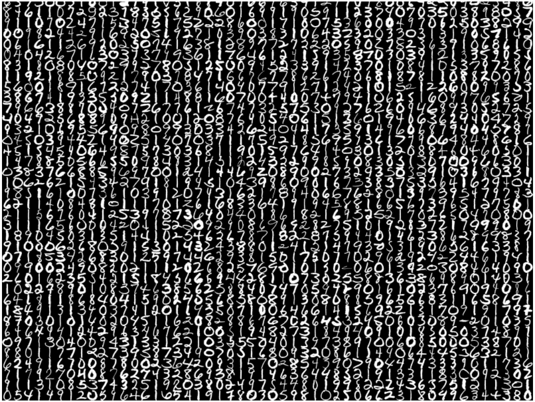
$$\downarrow$$

0 5 4 1 4 9

e.g. determine which digit $\{0, 1, \ldots, 9\}$ is in depicted in each image

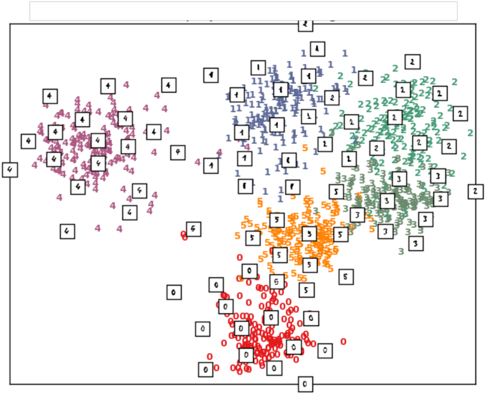
Digit recognition

Determine which digit $\{0, 1, \ldots, 9\}$ is in depicted in each image



k-Nearest Neighbors classification

Memorize training examples, predict labels using labels of the **k closest** training points



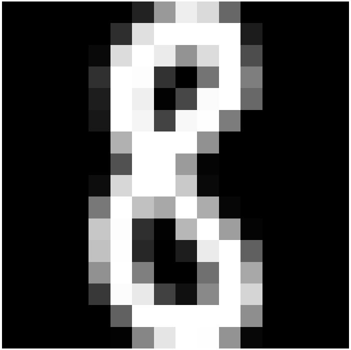
Intuition: **nearby** points have **similar** labels

k-Nearest Neighbors classification

```
Training:

  Load all training examples into memory

Prediction:

  Compute the distance between each training
  example and the query point

  Find the k closest training examples
  to the query point

  Return a majority vote over their labels
```

Choosing features: Images as arrays

Grayscale images $\leftrightarrow$ 2-d arrays of M-by-N pixel intensities



Choosing features: Images as arrays

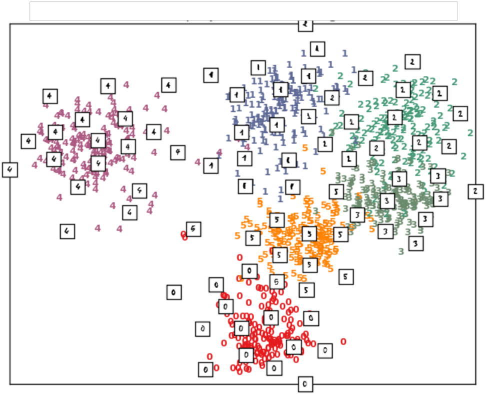
Grayscale images $\leftrightarrow$ 2-d arrays of M-by-N pixel intensities

```
array([[-1.   , -1.   , -1.   , -1.   , -1.   , -1.   , -1.   , -1.   ,
        -0.643,  0.625,  0.815, -0.533, -1.   , -1.   , -1.   , -1.   ],
       [-1.   , -1.   , -1.   , -1.   , -1.   , -1.   , -1.   , -0.441,
         0.951,  1.   ,  0.902, -0.111, -1.   , -1.   , -1.   , -1.   ],
       [-1.   , -1.   , -1.   , -1.   , -1.   , -1.   , -0.724,  0.776,
         1.   ,  0.622, -0.736, -1.   , -1.   , -1.   , -1.   , -1.   ],
       [-1.   , -1.   , -1.   , -1.   , -1.   , -1.   ,  0.389,  1.   ,
         0.739, -0.908, -1.   , -1.   , -1.   , -1.   , -1.   , -1.   ],
       [-1.   , -1.   , -1.   , -1.   , -1.   , -0.387,  0.985,  0.943,
        -0.598, -1.   , -1.   , -1.   , -1.   , -1.   , -1.   , -1.   ],
       [-1.   , -1.   , -1.   , -1.   , -0.824,  0.728,  1.   , -0.044,
        -1.   , -1.   , -1.   , -1.   , -1.   , -1.   , -1.   , -1.   ],
       [-1.   , -1.   , -1.   , -0.989,  0.364,  1.   ,  0.54 , -0.972,
        -1.   , -1.   , -1.   , -1.   , -1.   , -1.   , -1.   , -1.   ],
       [-1.   , -1.   , -1.   , -0.441,  0.999,  0.854, -0.675, -1.   ,
        -1.   , -1.   , -1.   , -1.   , -1.   , -1.   , -1.   , -1.   ],
       [-1.   , -1.   , -0.925,  0.8  ,  0.995, -0.121, -1.   , -1.   ,
        -1.   , -1.   , -1.   , -1.   , -1.   , -1.   , -1.   , -1.   ],
       [-1.   , -1.   , -0.495,  0.999,  0.782, -0.99 , -0.852, -0.172,
         0.208, -0.038, -0.516, -1.   , -1.   , -1.   , -1.   , -1.   ],
       [-1.   , -1.   , -0.179,  1.   ,  0.125, -0.242,  0.773,  1.   ,
         1.   ,  1.   ,  0.998,  0.534, -0.628, -1.   , -1.   , -1.   ],
       [-1.   , -1.   , -0.339,  1.   ,  0.756,  0.993,  0.99 ,  0.571,
        -0.05 , -0.317, -0.106,  0.666,  0.845, -0.699, -1.   , -1.   ],
       [-1.   , -1.   , -0.773,  0.945,  1.   ,  0.858, -0.304, -1.   ,
        -1.   , -1.   , -0.967,  0.327,  1.   , -0.079, -1.   , -1.   ],
       [-1.   , -1.   , -1.   ,  0.418,  1.   , -0.126, -0.995, -1.   ,
        -0.986, -0.57 ,  0.494,  1.   ,  0.957, -0.737, -1.   , -1.   ],
       [-1.   , -1.   , -1.   , -0.752,  0.297,  0.995,  0.698,  0.646,
         0.724,  1.   ,  1.   ,  0.912, -0.358, -1.   , -1.   , -1.   ],
       [-1.   , -1.   , -1.   , -1.   , -1.   , -0.607,  0.351,  0.595,
         0.841,  0.595,  0.312, -0.797, -1.   , -1.   , -1.   , -1.   ]])
```

Flatten each array into a vector, representing each image as a "vector of pixels"

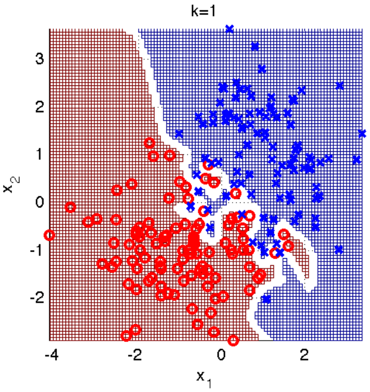
Measuring similarity: Euclidean distance

Grayscale images $\leftrightarrow$ 1-d vector of $M \cdot N$ pixel intensities



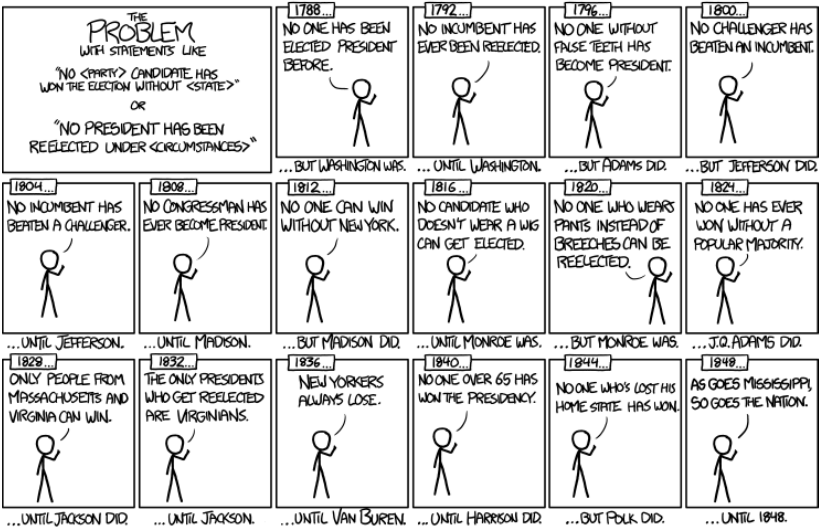
Measure similarity using the Euclidean distance between any two pixel vectors

Complexity control: how many neighbors k?



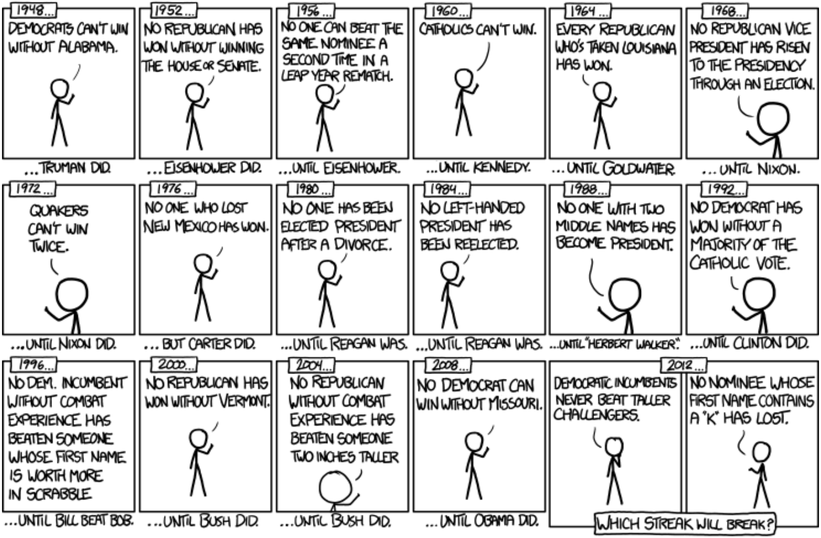
Small k gives a **complex** boundary, **large k** results in **coarse** averaging

Overfitting[7]



[7] http://xkcd.com/1122

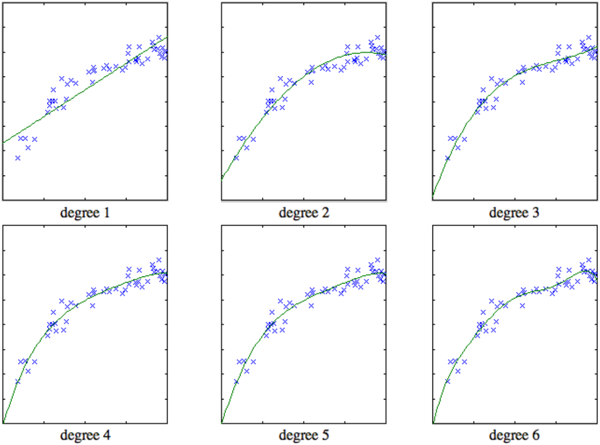
Overfitting[7]



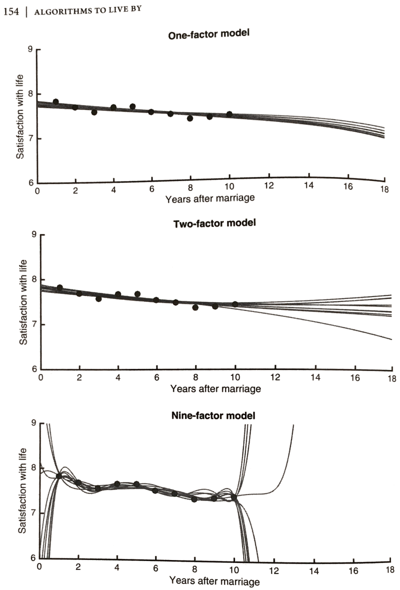
[7] http://xkcd.com/1122

Philosophy

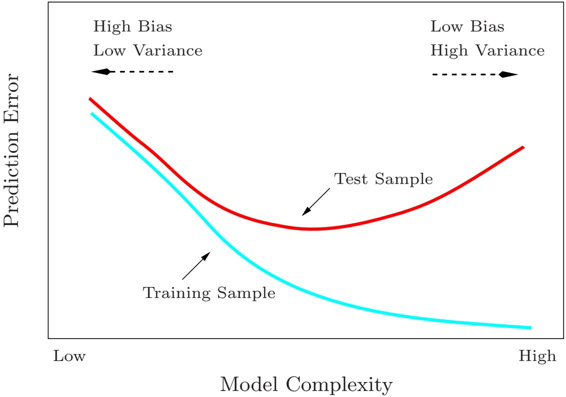
Our models should be complex enough to explain the past, but simple enough to generalize to the future



Bias-variance tradeoff

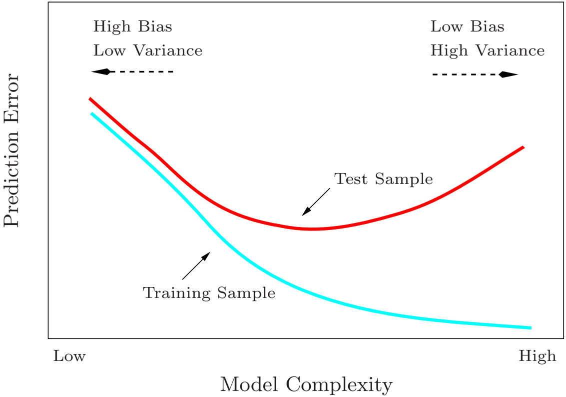


Bias-variance tradeoff



Simple models may be “wrong” (high bias), but fits don’t vary a lot with different samples of training data (low variance)

Bias-variance tradeoff

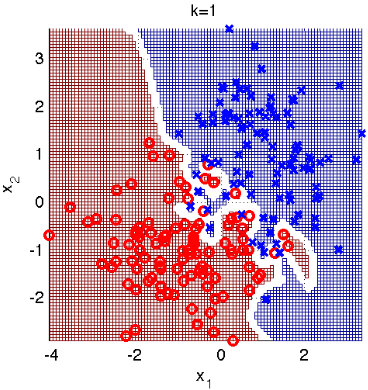


Flexible models can capture more complex relationships (low bias), but are also sensitive to noise in the training data (high variance)

Cross-validation

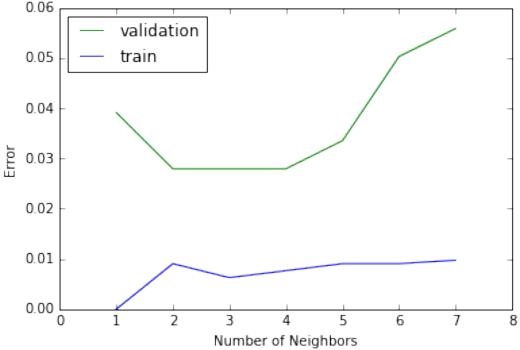
- Randomly split our data into three sets
- Fit models on the training set
- Use the validation set to find the best model
- Quote final performance of this model on the test set

Complexity control: how many neighbors k?



Evaluate performance on a **held-out test set** to assess **generalization error**

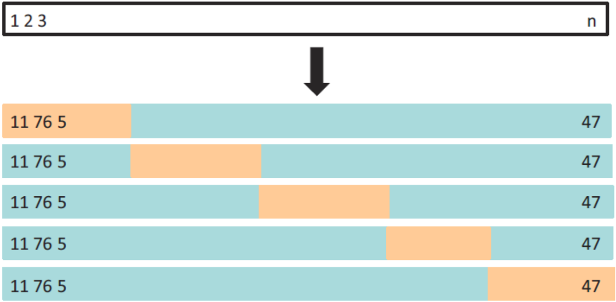
Cross-validation for digit recognition



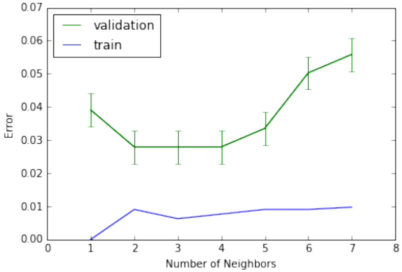
- Randomly split our data into three sets
- For each number of neighbors k:
    - Fit a model to the training set
    - Evaluate on the validation set
- Select the model with the lowest validation error
- Quote final performance of this model on the test set

K-fold cross-validation

Estimates of generalization error from one train / validation split can be noisy, so shuffle data and average over K distinct validation partitions instead



K-fold cross-validation for digit recognition



- Randomly divide the data into F folds
- For each number of neighbors k
  - For each fold:
    - Fit a model on everything but one fold
    - Evaluate the model on the held-out fold
  - Compute the average validation error across folds
- Select the model with the lowest average validation error
- Quote final performance of this model on the test set

Performance

Predicted Class

Actual Class	0	1	2	3	4	5	6	7	8	9
0	97.5%		0.2%	0.3%	0.7%	0.5%	0.2%		0.7%	
1		99.6%						0.2%	0.2%	
2	0.8%		94.5%	0.8%	0.8%	1.1%	0.3%	0.8%	0.8%	
3	0.3%		0.6%	92.1%	0.3%	3.2%		0.6%	2.2%	0.6%
4	0.3%	1.5%	1.8%		90.9%	0.3%	1.8%		1.2%	2.1%
5	1.1%		0.4%	2.2%	1.1%	90.3%	4.0%		1.1%	
6	2.6%		2.3%		1.8%	0.6%	92.4%		0.3%	
7					0.6%			95.5%	0.6%	3.3%
8	0.4%	0.7%	1.8%	2.2%	2.9%	0.4%	0.4%	1.1%	90.1%	
9		0.3%			1.6%	0.3%		2.3%	0.6%	94.8%

k-Nearest Neighbors

Simple approach:

Predict the future by memorizing the past

k-Nearest Neighbors

Still many choices:

Number of neighbors, feature transformations, distance measures, ...

k-Nearest Neighbors

Problems (practice):

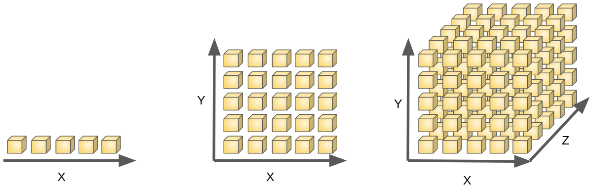
Becomes costly as we see more training examples

k-Nearest Neighbors

Problems (theory):

Performs poorly when large number of features relative to examples

Curse of dimensionality[8]



kNN requires a prohibitive number of training examples in high dimensions (e.g., for text data)

[8] http://bit.ly/cursedim

k-Nearest Neighbors

Even with simple methods, there are many decisions to be made in developing a successful machine learning solution

Bigger models $\neq$ Better models