COMP 551 – Applied Machine Learning
Lecture 8: Instance-based learning

Associate Instructor: Herke van Hoof

(herke.vanhoof@mcgill.ca)

Slides mostly by: Joelle Pineau (*jpineau@cs.mcgill.ca)*

Class web page: *www.cs.mcgill.ca/~jpineau/comp551*

Today's quiz

Q1. Consider the following dataset.

Day	Weather	GoHiking?
Mon	Sunny	No
Tues	Cloudy	No
Wed	Rain	No
Thurs	Rain	No
Fri	Sunny	No
Sat	Sunny	No
Sun	Sunny	Yes

Let "Day" and "Weather" be the input features and "GoHiking?" be the output.

a) What is the entropy of this set of examples?

H(D) = ??

b) What is the information gain of feature "Weather"?

IG = H(D) – H(D | Weather) = ??

c) What is the information gain of feature "Day"?

IG = H(D) – H(D | Day) = ??

Q2. Give a decision tree that correctly represents the following Boolean function: Y = [X1 AND X2] OR [X2 AND X3]

(Many possible correct answers.)

Today's quiz

Q1. Consider the following dataset.

Day	Weather	GoHiking?
Mon	Sunny	No
Tues	Cloudy	No
Wed	Rain	No
Thurs	Rain	No
Fri	Sunny	No
Sat	Sunny	No
Sun	Sunny	Yes

Let "Day" and "Weather" be the input features and "GoHiking?" be the output.

a) What is the entropy of this set of examples?

H(D) = -(1/7)*log(1/7)/log(2)-(6/7)*log(6/7)/log(2)

b) What is the information gain of feature "Weather"?

IG = H(D) – H(D | Weather)

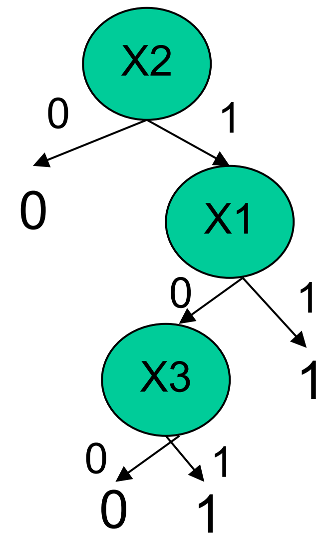
IG = H(D) – ((4/7)*(-(3/4)*log(3/4)/log(2)-(1/4)*log(1/4)/log(2)))
+ (2/7)*(0) + (1/7)*(0)

c) What is the information gain of feature "Day"?

IG = H(D) – H(D | Day) = H(D) – 0 = H(D)

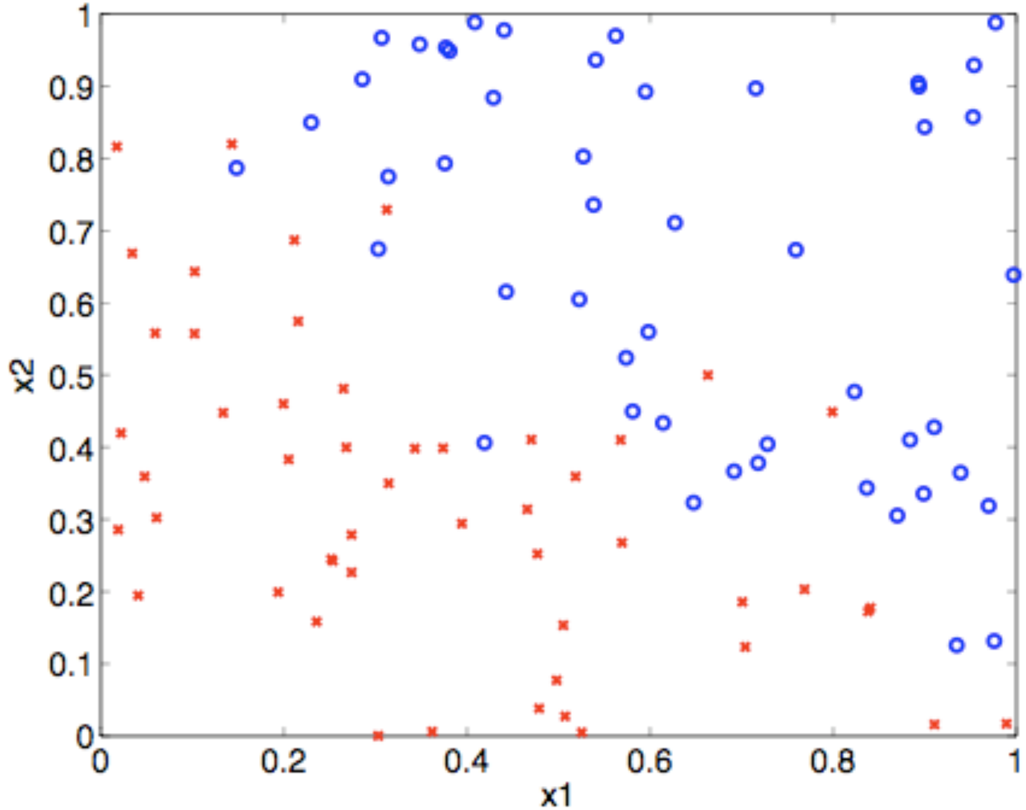
Q2. Give a decision tree that correctly represents the following Boolean function: Y = [X1 AND X2] OR [X2 AND X3]

(Many possible correct answers.)



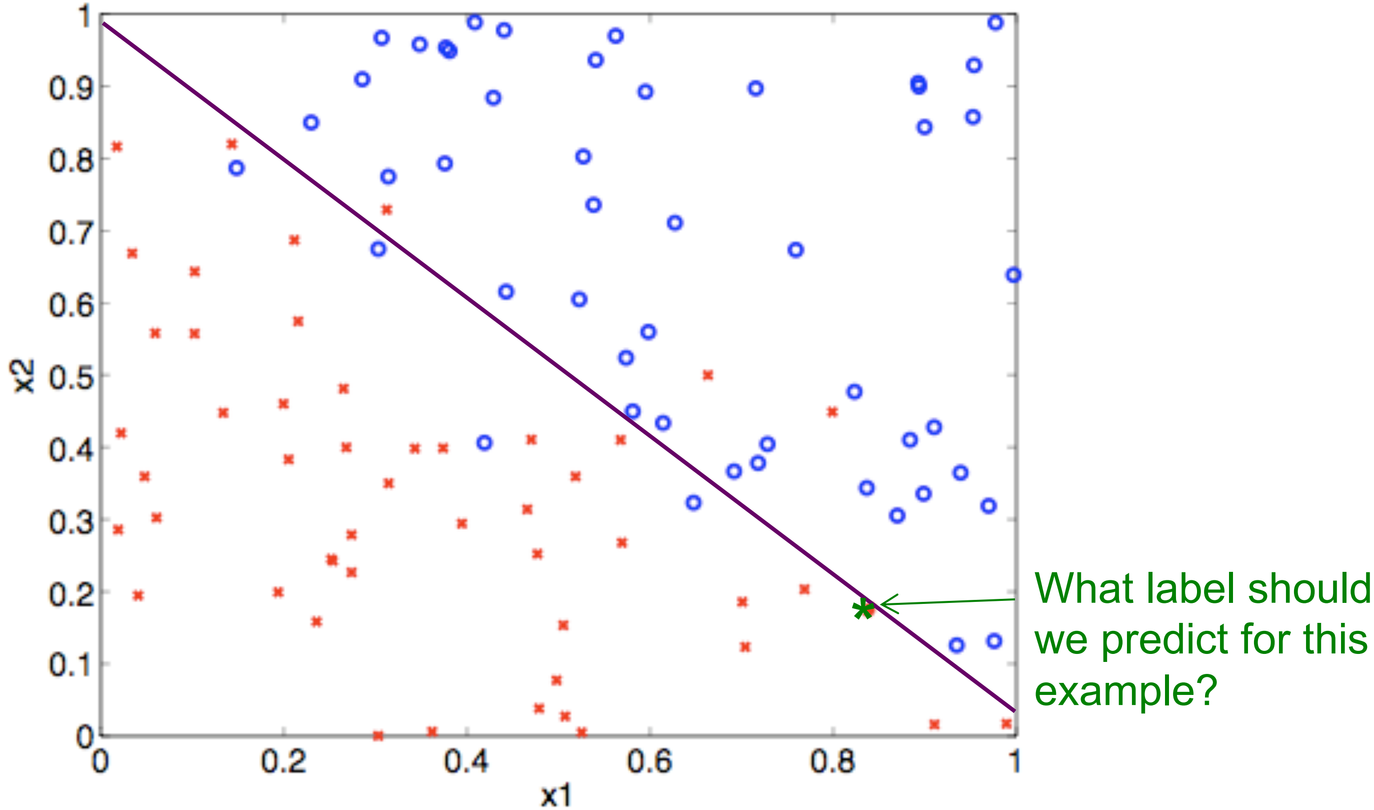
A complete (artificial) example

- An artificial binary classification problem with two real-valued input features:



A complete (artificial) example

- An artificial binary classification problem with two real-valued input features:



Parametric supervised learning

- Example: logistic regression. Input: dataset of labeled examples.
- From this, learn a parameter vector of a fixed size such that some error measure based on the training data is minimized.
- These methods are called parametric, and main goal is to summarize the data using the parameters.
  - Parametric methods are typically **global** = one set of parameters for the entire data space.

Instance based learning methods

- Key idea: just store all training examples $< \mathbf{x}_i, y_i >$.
- When a query is made, **locally** compute the value y of new instance based on the values of the most similar points.

Instance based learning methods

- Key idea: just store all training examples < $\boldsymbol{x}_i$, y_i >.
- When a query is made, **locally** compute the value y of new instance based on the values of the most similar points.
- The regressor / classifier can now **not** be represented by a fixed-sized vector: representation depends on dataset

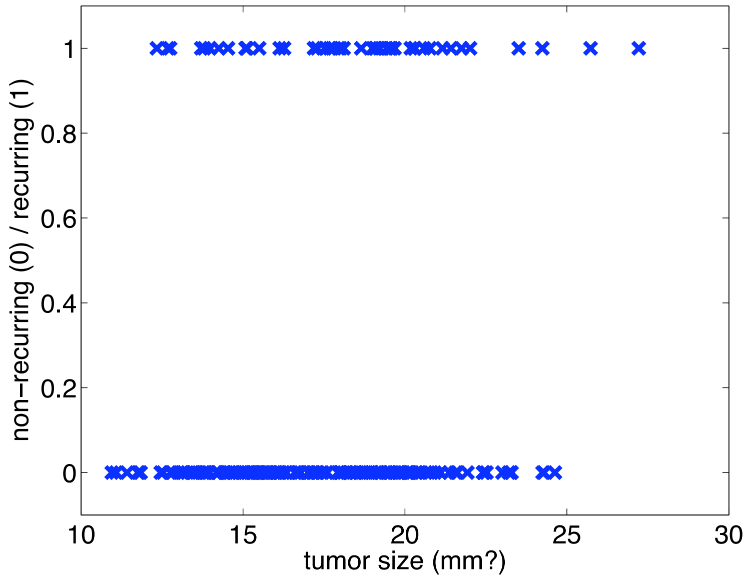
Instance based learning methods

- Key idea: just store all training examples < $\boldsymbol{x}_i$, y_i >.
- When a query is made, **locally** compute the value y of new instance based on the values of the most similar points.
- The regressor / classifier can now **not** be represented by a fixed-sized vector: representation depends on dataset
- Different algorithms for computing the value of the new point based on the existing values

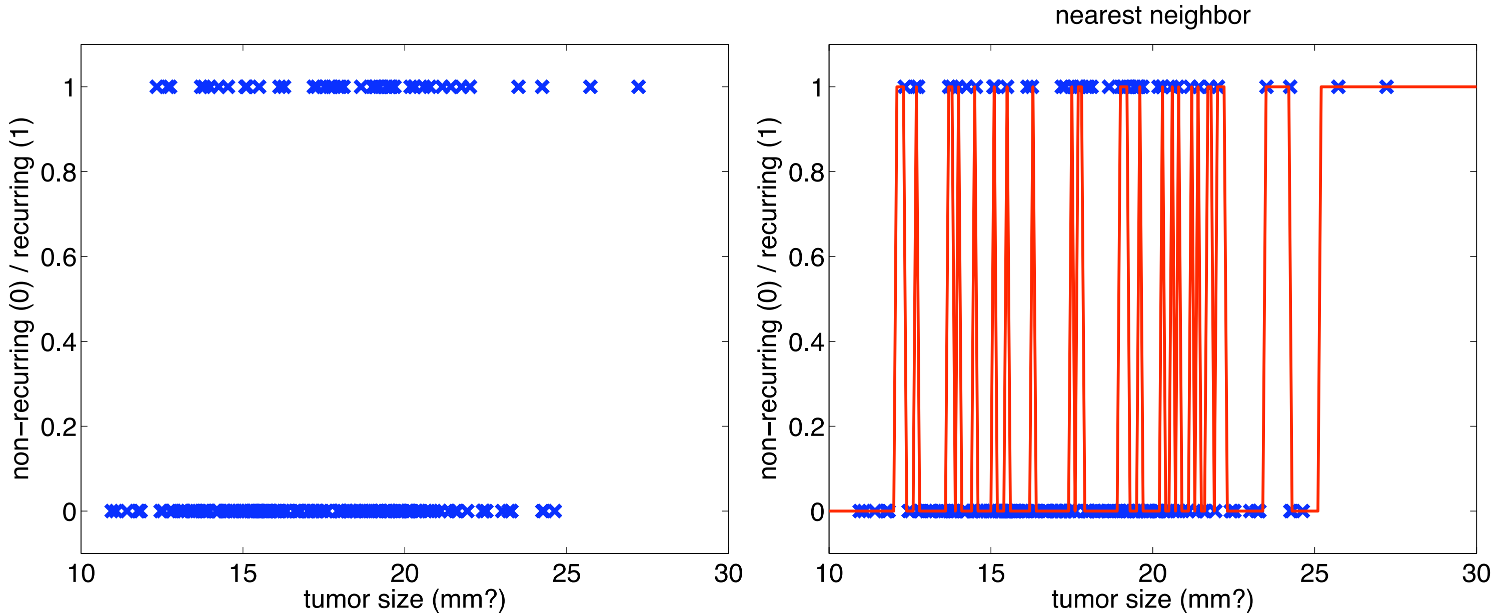
Non-parametric learning methods

- Key idea: just store all training examples $< \boldsymbol{x}_i, y_i >$.
- When a query is made, computer the value of the new instance based on the values of the closest (most similar) points.
- Requirements:
  - **A distance function.**
  - How many closest points (neighbors) to look at?
  - How do we computer the value of the new point based on the existing values?

Simple idea: Connect the dots!

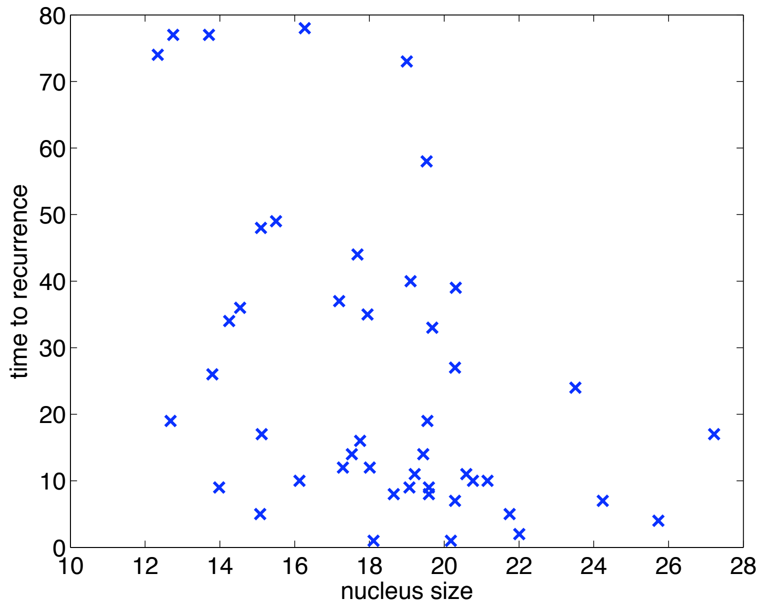


Simple idea: Connect the dots!

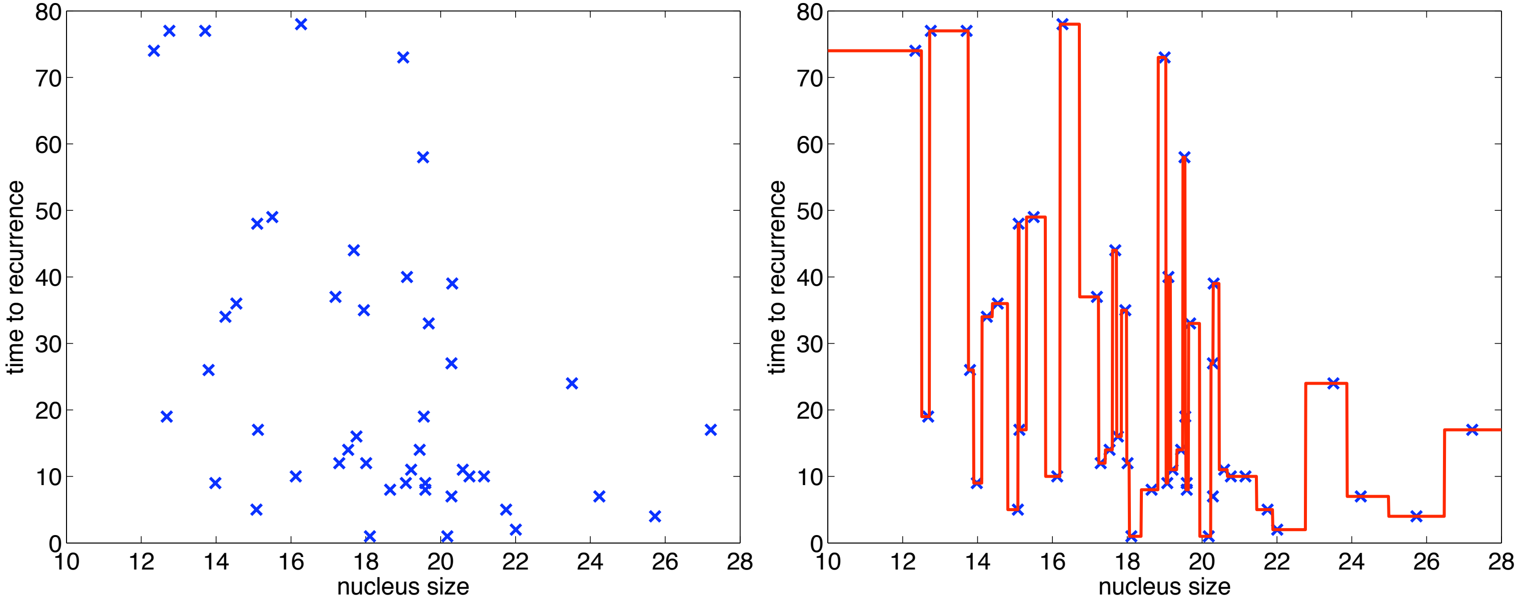


Wisconsin data set, classification

Simple idea: Connect the dots!



Simple idea: Connect the dots!



Wisconsin data set, regression

One-nearest neighbor

- **Given**: Training data *X*, distance metric *d* on *X*.
- **Learning**: Nothing to do! (Just store the data).

One-nearest neighbor

- **Given**: Training data X, distance metric d on X.
- **Learning**: Nothing to do! (Just store the data).
- **Prediction**: For $\mathbf{x} \in X$

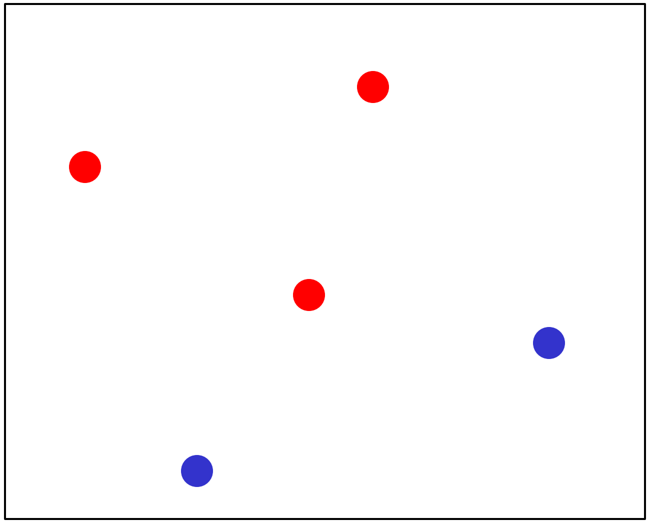
  Find nearest training sample $\mathbf{x}_i$.

  $$i^* = \text{argmin}_i\, d(\mathbf{x}_i, \mathbf{x})$$

  Predict $y = y_{i^*}$

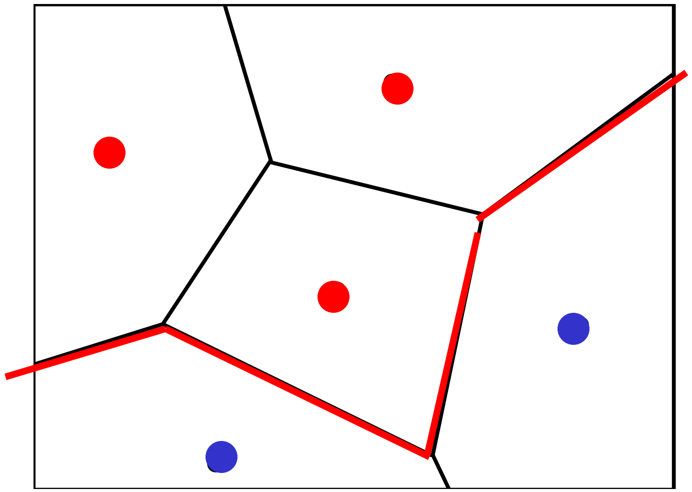
What does the approximator look like?

- What do you think the decision boundary looks like?



What does the approximator look like?

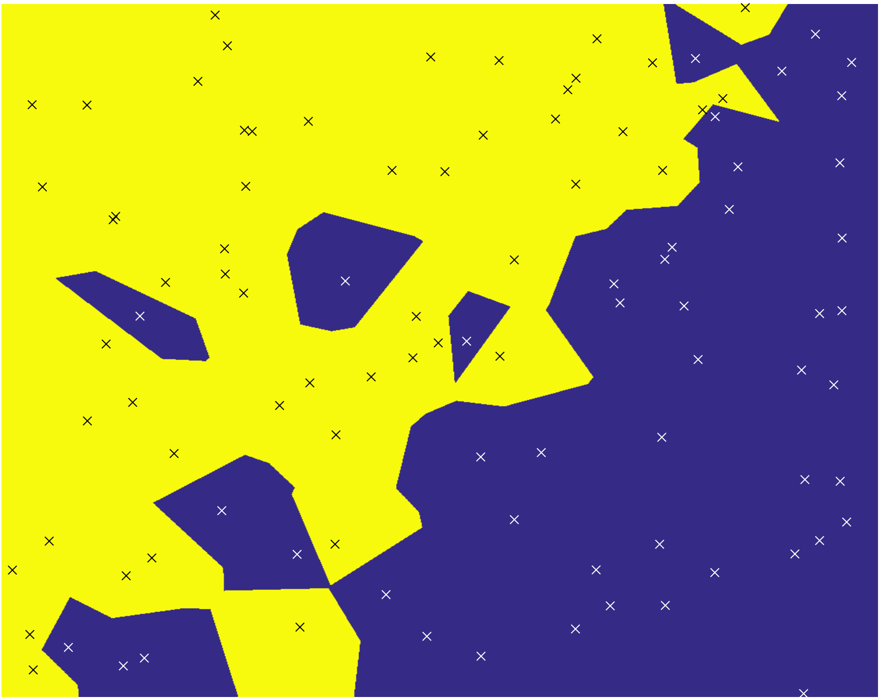
- Nearest-neighbor does not explicitly compute decision boundaries.
- But the effective decision boundaries are a subset of the Voronoi diagram for the training data.



- Each decision boundary is a line segment that is equidistant between two points of opposite classes.

What does the approximator look like?

- Example



One-nearest neighbor

- **Given**: Training data X, distance metric d on X.

- **Learning**: Nothing to do! (Just store the data).

- **Prediction**: For $\mathbf{x} \in X$

  Find nearest training sample $\mathbf{x}_i$.

  $$i^* = \text{argmin}_i \, d(\mathbf{x}_i, \mathbf{x})$$

  Predict $y = y_{i^*}$

What kind of distance metric?

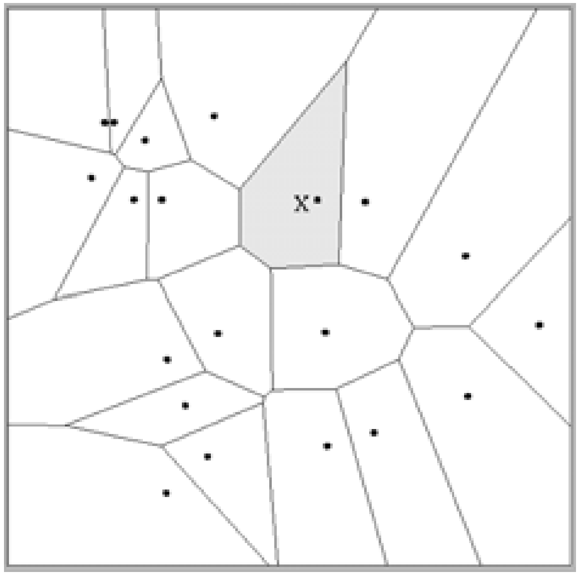
What kind of distance metric?

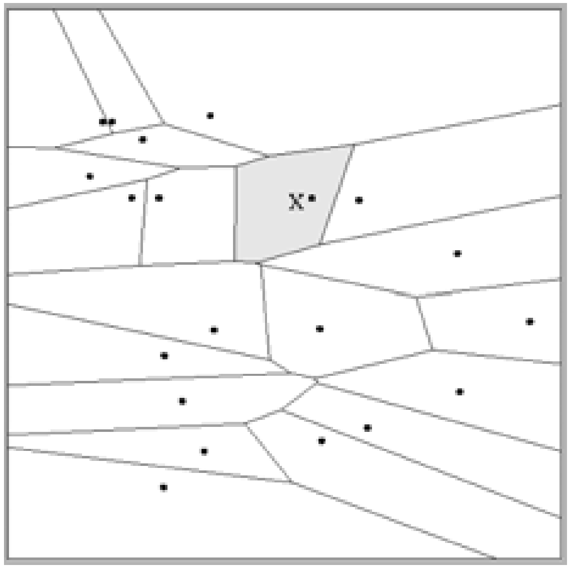
- Euclidean distance.
- Weighted Euclidean distance (with weights based on domain knowledge): $d(\mathbf{x}, \mathbf{x}') = \sum_{j=1:m} w_j (x_j - x_j')^2$

What kind of distance metric?

- Euclidean distance.
- Weighted Euclidean distance (with weights based on domain knowledge): $d(\mathbf{x}, \mathbf{x}') = \sum_{j=1:m} w_j (x_j - x_j')^2$
- Maximum / minimum difference along any axis.
- An arbitrary distance or similarity function d, specific for the application at hand (works best, if you have one.)

Choice of distance metric is important!





Left: both attributes weighted equally; Right: second attributes weighted more

Distance metric tricks

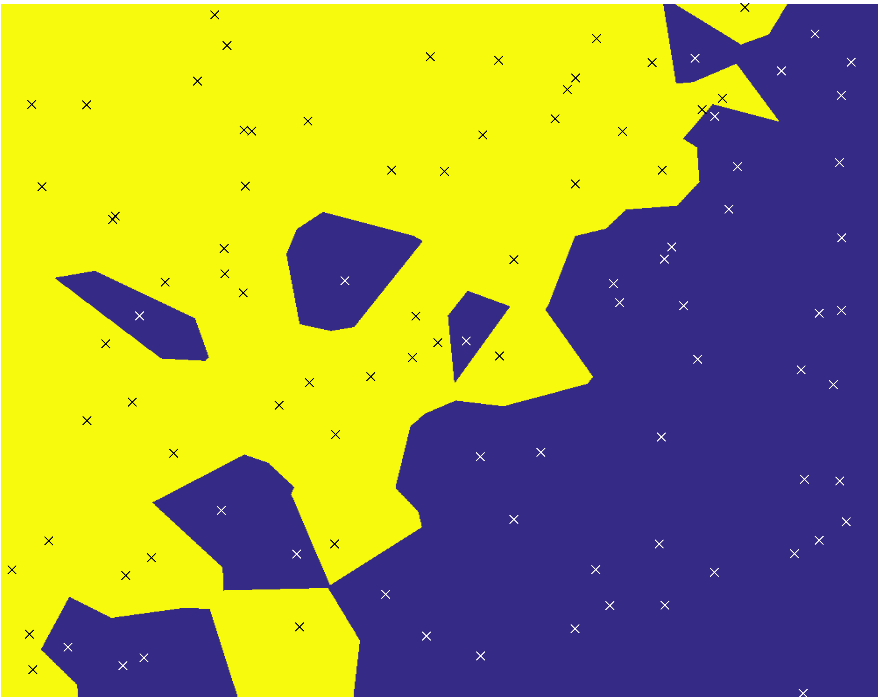
- You may need to do feature preprocessing:
  - Scale the input dimensions (or normalize them).
  - Remove noisy and irrelevant inputs.
  - Determine weights for attributes based on cross-validation (or information-theoretic methods).

Distance metric tricks

- You may need to do feature preprocessing:
  - Scale the input dimensions (or normalize them).
  - Remove noisy and irrelevant inputs.
  - Determine weights for attributes based on cross-validation (or information-theoretic methods).
- Distance metric is often **domain-specific**.
  - E.g. string edit distance in bioinformatics.
  - E.g. trajectory distance in time series models for walking data.
- Distance can be learned sometimes.

k-nearest neighbor (kNN)

- In case of noise, a **single** bad label can cause a patch to be misclassified
- Safer to look at more than one close point?



k-nearest neighbor (kNN)

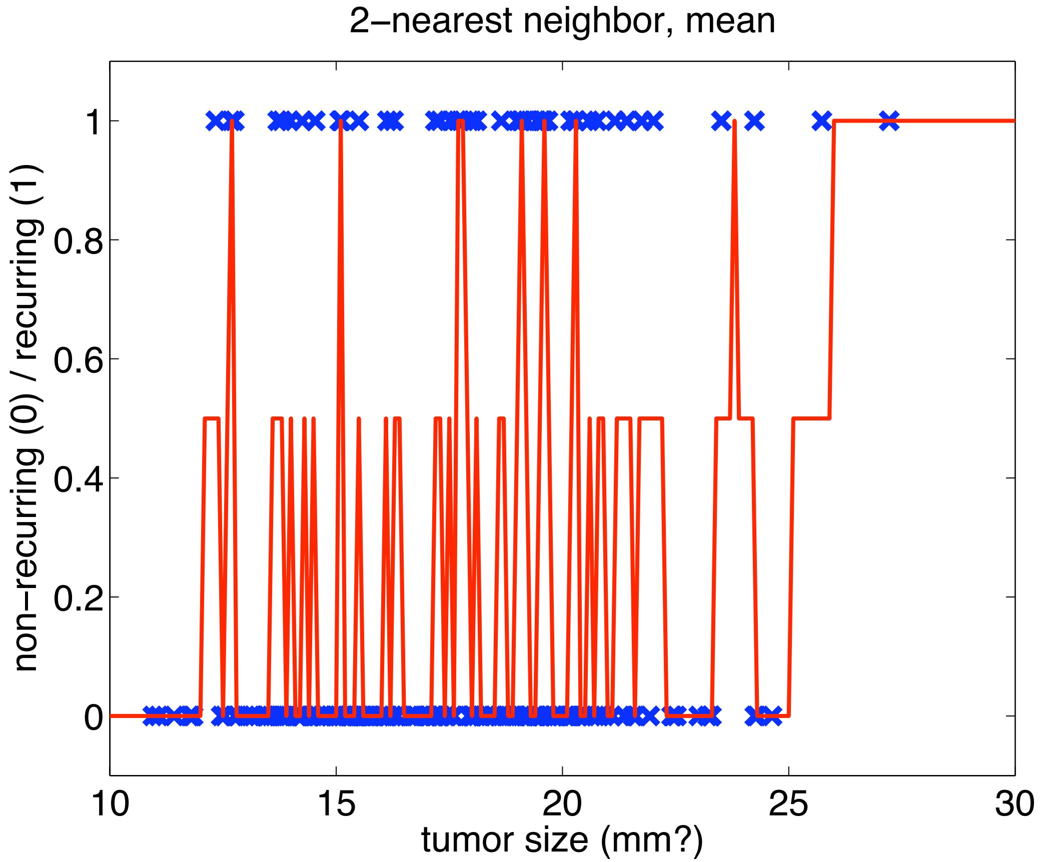
- **Given**: Training data X, distance metric d on X.

- **Learning**: Nothing to do! (Just store the data).

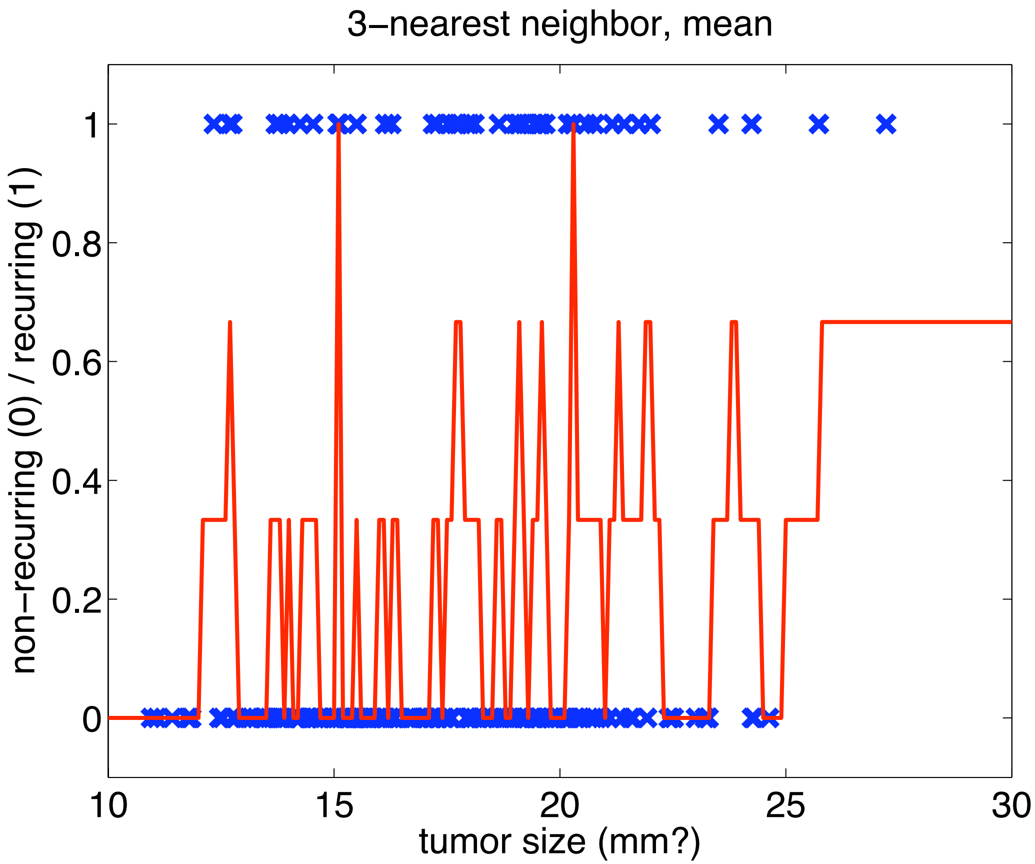
- **Prediction**:
  - For $\mathbf{x} \in X$, find the k nearest training samples to $\mathbf{x}$.
  - Let their indices be $i_1, i_2, \ldots, i_k$.
  - Predict: y = mean/median of $\{y_{i1}, y_{i2}, \ldots, y_{ik}\}$ for regression

    y = majority of $\{y_{i1}, y_{i2}, \ldots, y_{ik}\}$ for classification, or empirical probability of each class.

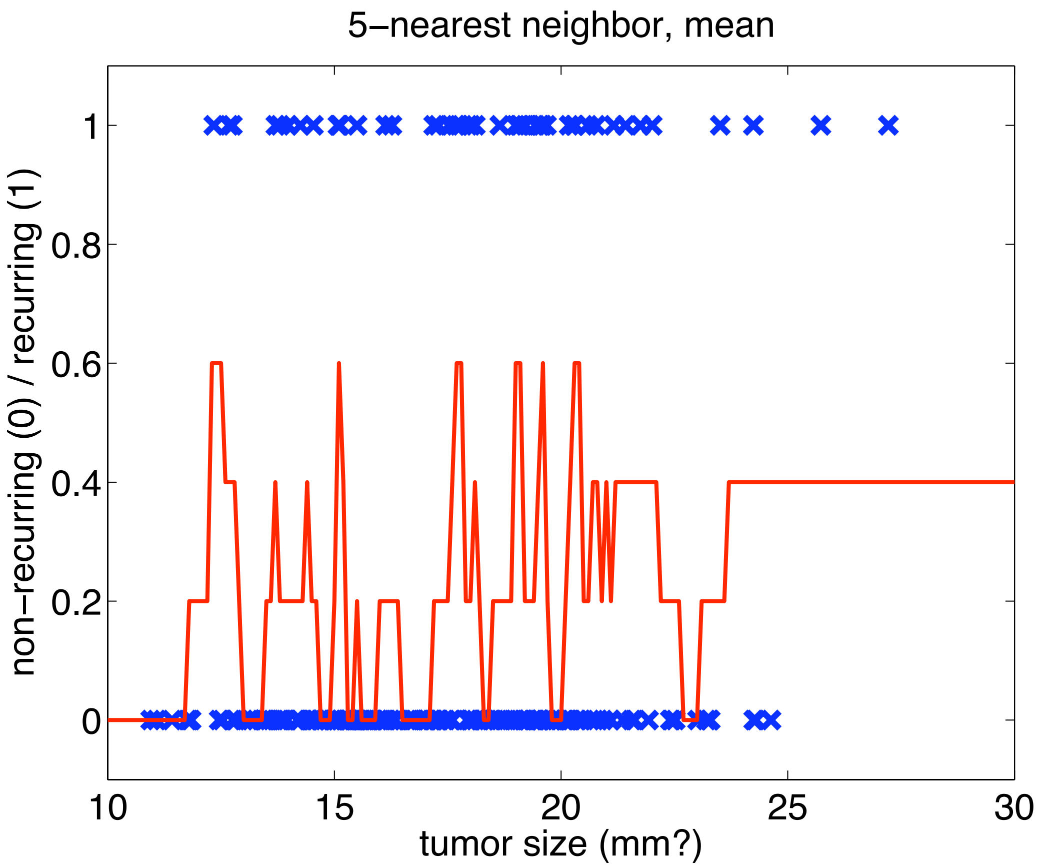
Classification, 2-nearest neighbor



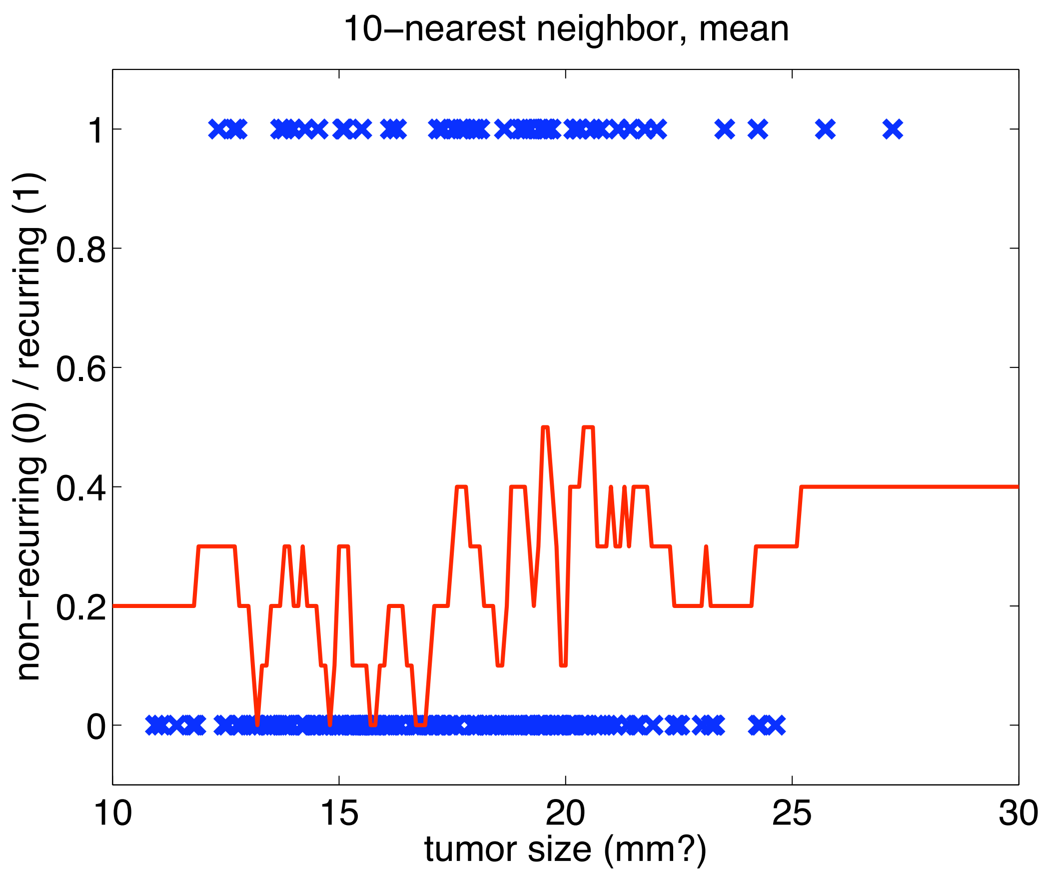
Classification, 3-nearest neighbor



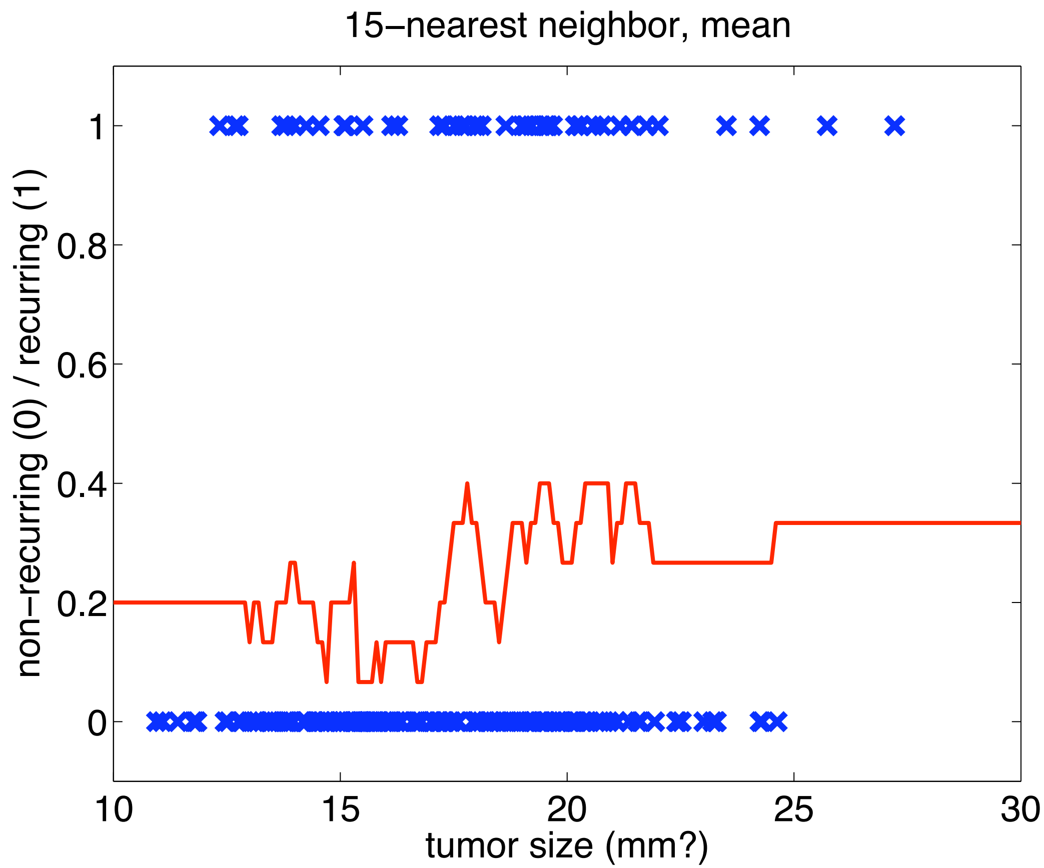
Classification, 5-nearest neighbor



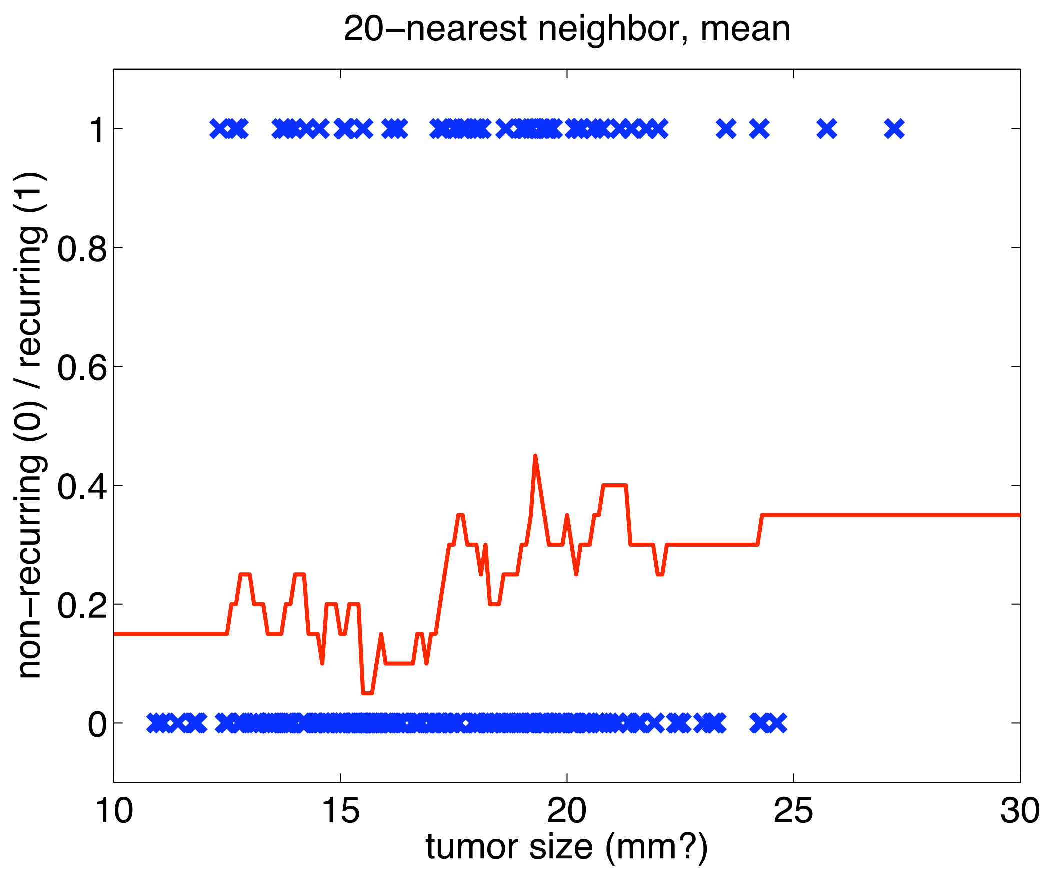
Classification, 10-nearest neighbor



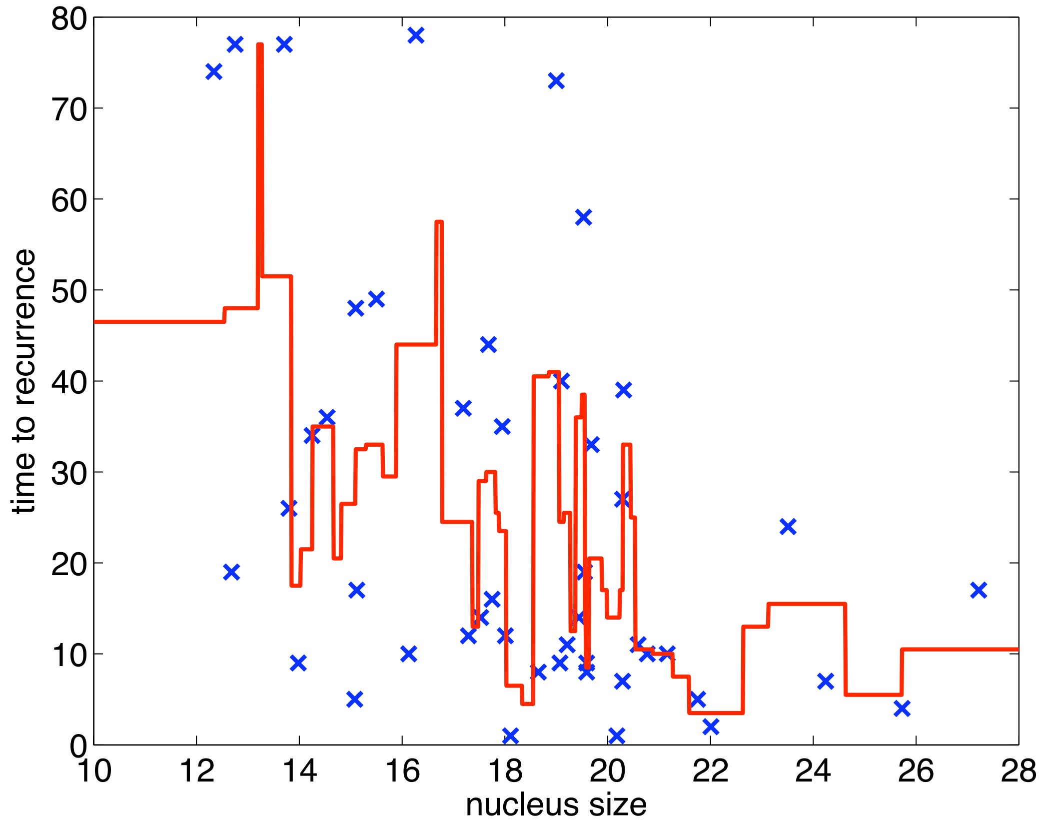
Classification, 15-nearest neighbor



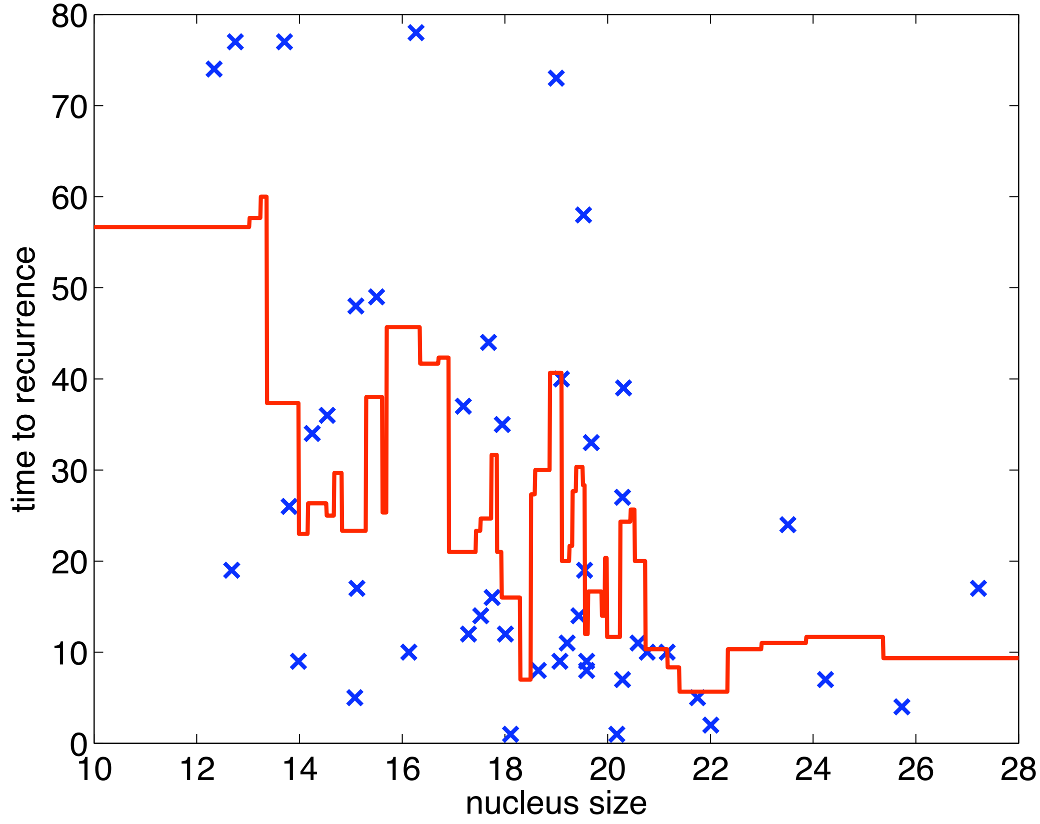
Classification, 20-nearest neighbor



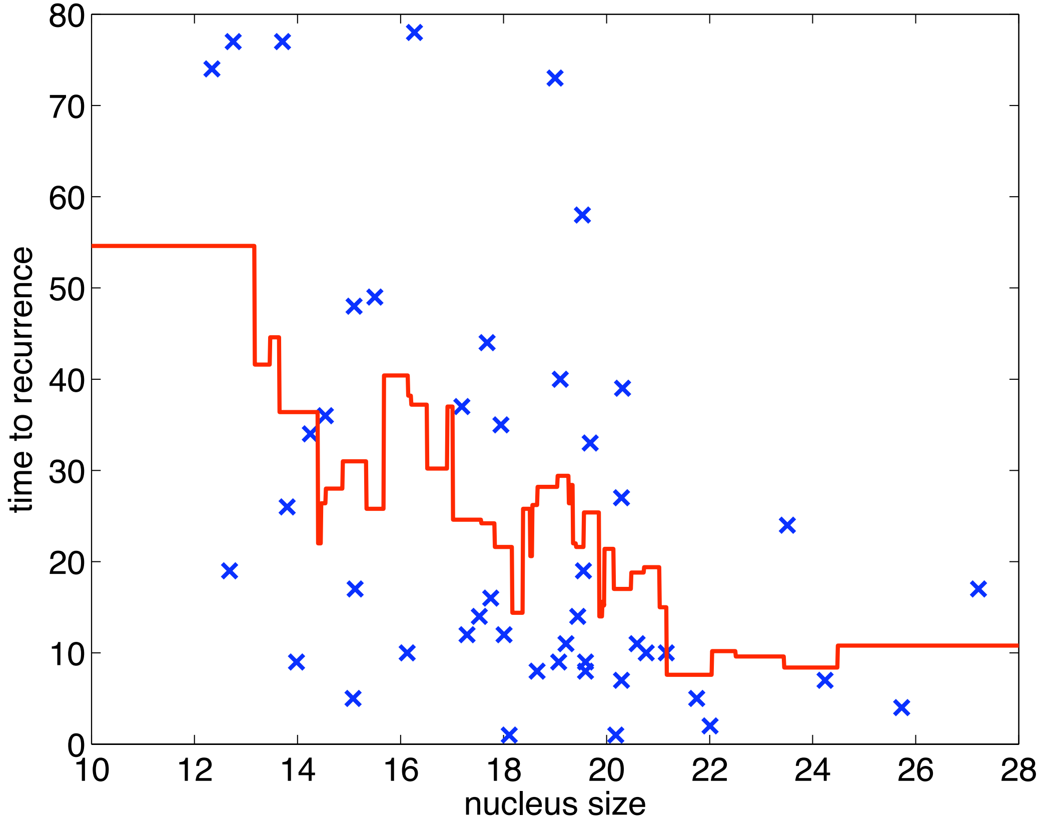
Regression, 2-nearest neighbor



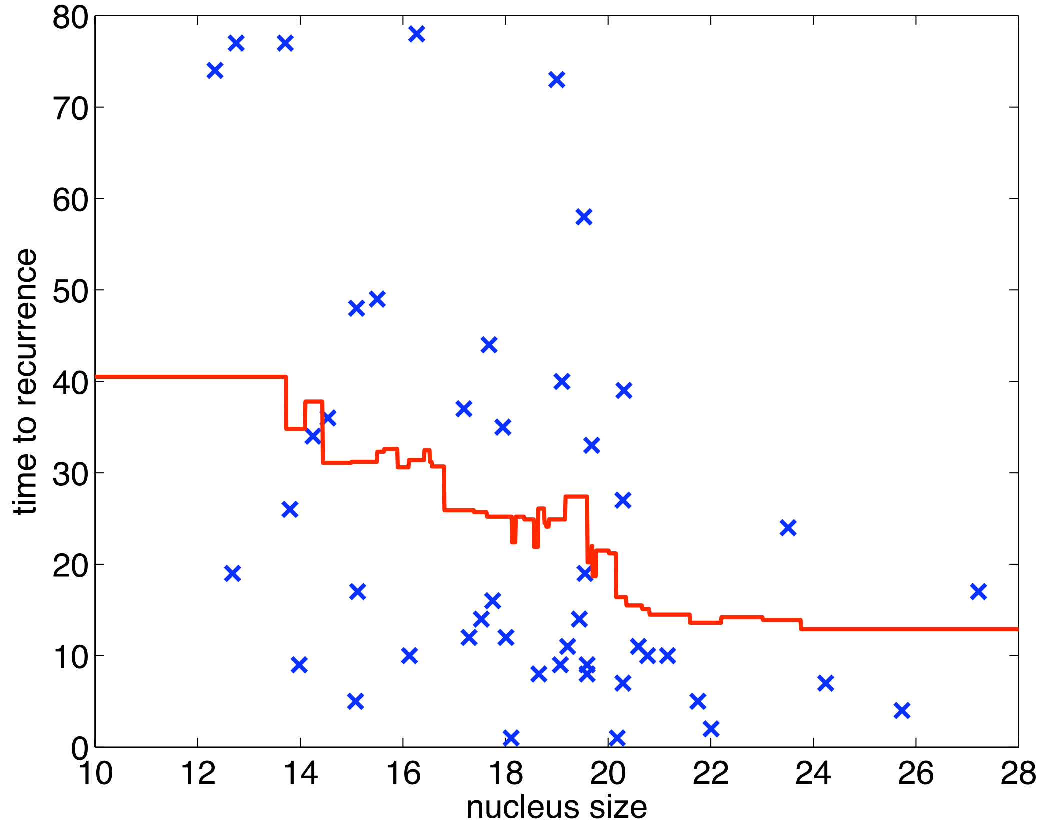
Regression, 3-nearest neighbor



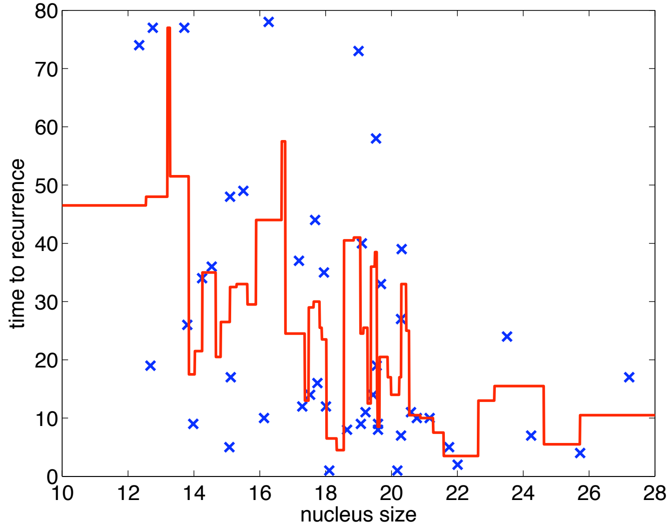
Regression, 5-nearest neighbor



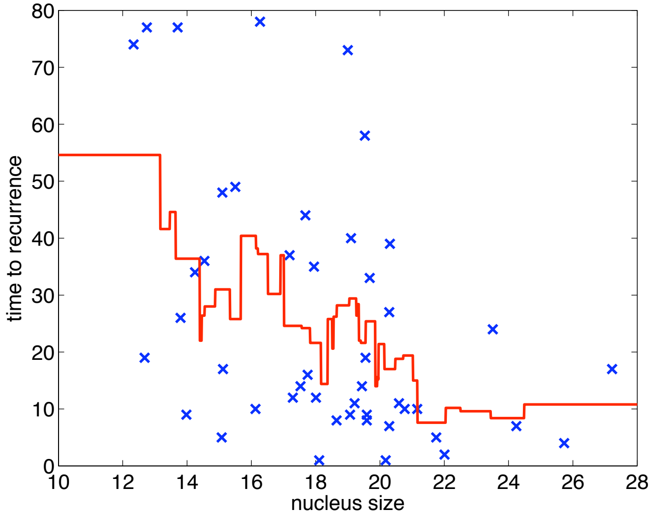
Regression, 10-nearest neighbor



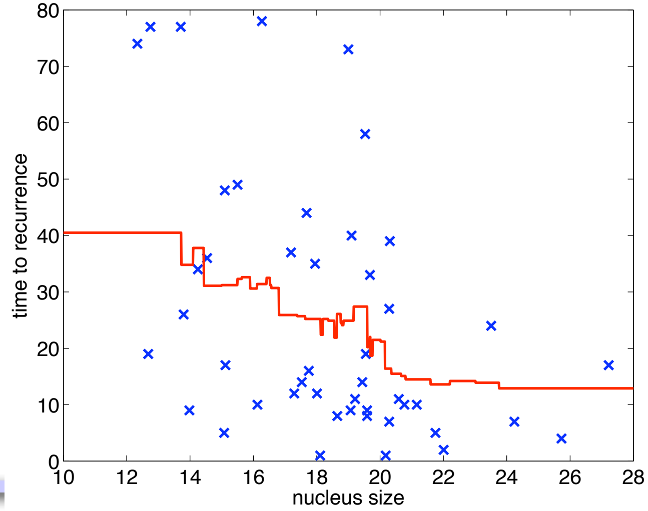
What is the best regressor?



K=2



K=5



K=10

Bias-variance trade-off

- What happens if *k* is **low**?

- What happens if *k* is **high**?

Bias-variance trade-off

- What happens if *k* is **low**?

  Very non-linear functions can be approximated, but we also capture the noise in the data. Bias is low, variance is high.

- What happens if *k* is **high**?

  The output is much smoother, less sensitive to data variation. High bias, low variance.

- A validation set can be used to pick the best *k*.

Limitations of *k*-nearest neighbor (kNN)

- A lot of discontinuities!
- Sensitive to small variations in the input data.
- Can we fix this but still keep it (fairly) local?

k-nearest neighbor (kNN)

- **Given**: Training data X, distance metric d on X.
- **Learning**: Nothing to do! (Just store the data).
- **Prediction**:
  - For $\boldsymbol{x} \in X$, find the k nearest training samples to $\boldsymbol{x}$.
  - Let their indices be $i_1, i_2, \ldots, i_k$.
  - Predict: y = mean/median of $\{y_{i1}, y_{i2}, \ldots, y_{ik}\}$ for regression

    y = majority of $\{y_{i1}, y_{i2}, \ldots, y_{ik}\}$ for classification, or empirical probability of each class.

Distance-weighted (kernel-based) NN

- **Given**: Training data X, distance metric d on X, weighting function $w : R \rightarrow R$.

- **Learning**: Nothing to do! (Just store the data).

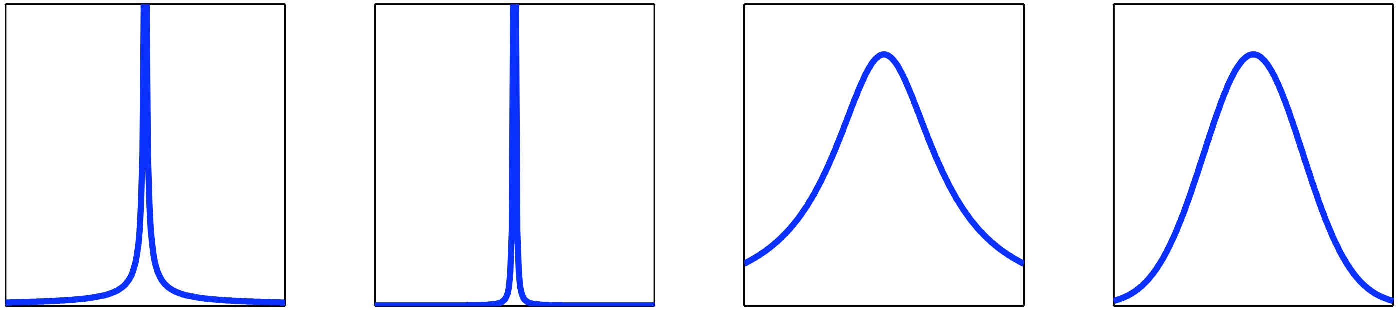
- **Prediction**:
  - Given input $\boldsymbol{x}$.
  - For each x_i compute $w_i = w(d(x_i,x))$.
  - Predict: $y = \sum_i w_i y_i \;/\; \sum_i w_i$.

Distance-weighted (kernel-based) NN

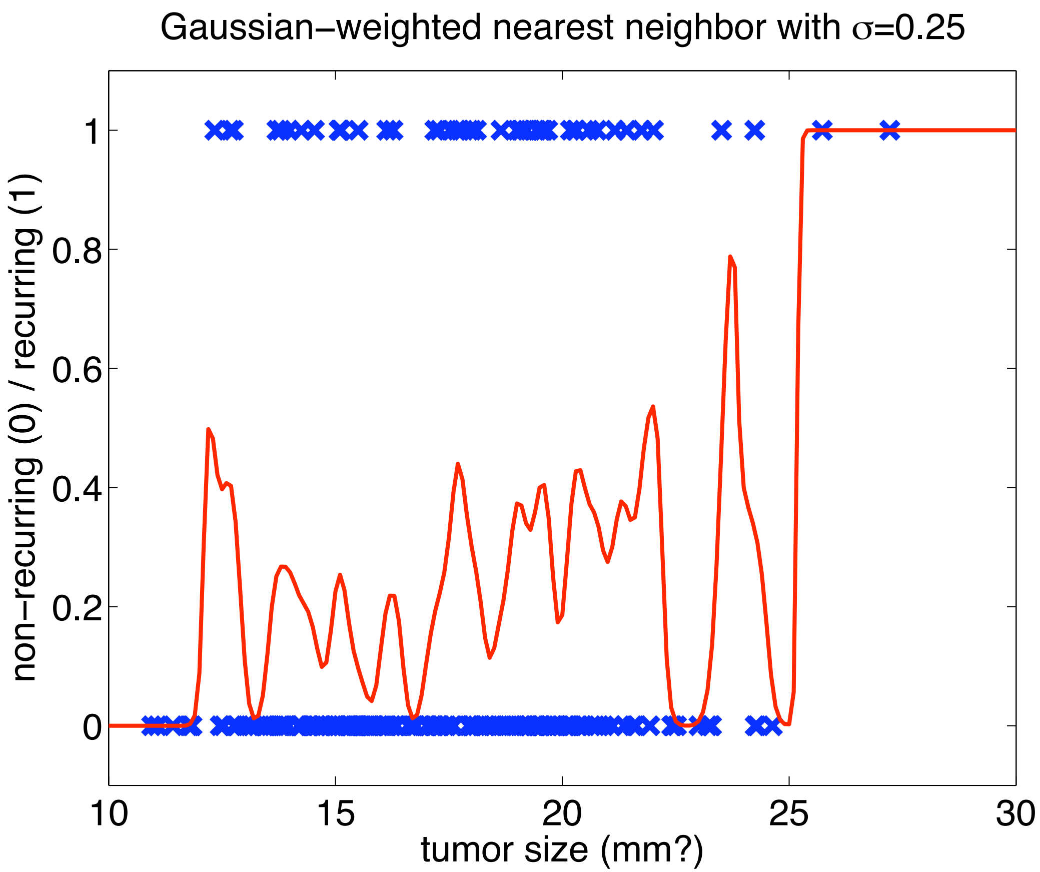
- **Given**: Training data X, distance metric d on X, weighting function $w : R \rightarrow R$.
- **Learning**: Nothing to do! (Just store the data).
- **Prediction**:
  - Given input $\boldsymbol{x}$.
  - For each x_i compute $w_i = w(d(x_i,x))$.
  - Predict: $y = \sum_i w_i y_i \ / \ \sum_i w_i$.
- How should we weigh the distances?

Some weighting functions

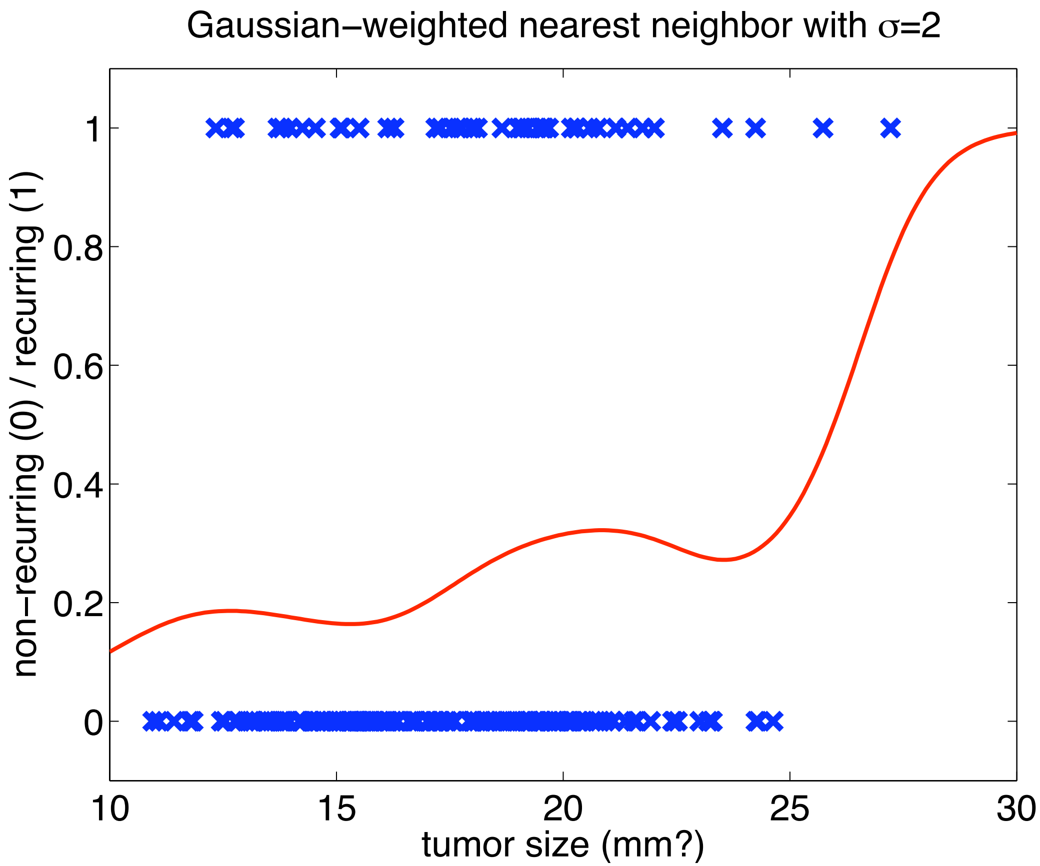
$$\frac{1}{d(\mathbf{x}_i, \mathbf{x})} \qquad \frac{1}{d(\mathbf{x}_i, \mathbf{x})^2} \qquad \frac{1}{c + d(\mathbf{x}_i, \mathbf{x})^2} \qquad e^{-\frac{d(\mathbf{x}_i, \mathbf{x})^2}{\sigma^2}}$$



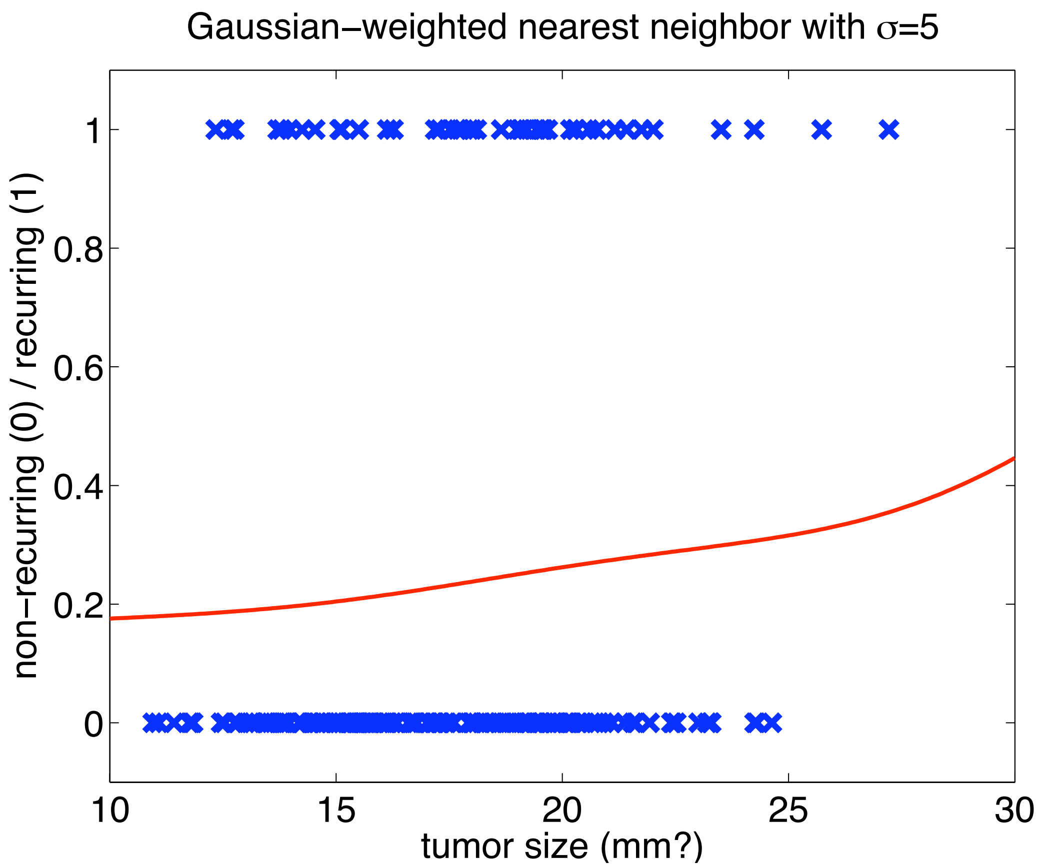
Gaussian weighting, small σ



Gaussian weighting, medium σ



Gaussian weighting, large σ



All examples get to vote! Curve is smoother, but perhaps too smooth?

Scaling up

- kNN in high-dimensional feature spaces?

- kNN with larger number of datapoints?

Scaling up

- kNN in high-dimensional feature spaces?
  – In high dim spaces, the distance between points appears similar.
  – A few points ("hubs") show up repeatedly in the top kNN [*Radovanovic et al., 2009*].

- kNN with larger number of datapoints?

Scaling up

- kNN in high-dimensional feature spaces?
  - In high dim spaces, the distance between points appears similar.
  - A few points (“hubs”) show up repeatedly in the top kNN [*Radovanovic et al., 2009*].

- kNN with larger number of datapoints?
  - Can be implemented efficiently, *O(log n)* at retrieval time, if we use smart data structures:
    - Condensation of the dataset (Use prototypes)
    - Hash tables in which the hashing function is based on the distance metric.
    - KD-trees (Tutorial: *http://www.autonlab.org/autonweb/14665*)

Instance based learning

- Instance-based learning refers to techniques where previous samples are used directly to make predictions
- What makes instance based methods different?
  - Model is typically *non-parametric* (no fixed parameter vector)
  - Algorithms are typically *lazy*

Lazy vs eager learning

- **Lazy learning**: Wait for query before generalization.
  - E.g. Nearest neighbour.
- **Eager learning**: Generalize before seeing query.
  - E.g. Logistic regression, LDA, decision trees, neural networks.
- Which is faster?
  - Training time?
  - Query answering time?

Pros and cons of lazy and eager learning

- Eager learners create global approximation.
- Lazy learners create many local approximations.
- If they use the same hypothesis space, a lazy learner can represent more complex functions (e.g., consider H = linear function).

Pros and cons of lazy and eager learning

- Eager learners create global approximation.
- Lazy learners create many local approximations.
- If they use the same hypothesis space, a lazy learner can represent more complex functions (e.g., consider H = linear function).

- Lazy learning has much faster training time.
- Eager learner does the work off-line

Pros and cons of lazy and eager learning

- Eager learners create global approximation.
- Lazy learners create many local approximations.
- If they use the same hypothesis space, a lazy learner can represent more complex functions (e.g., consider H = linear function).

- Lazy learning has much faster training time.

- Lazy learner typically has slower query answering time (depends on number of instances and number of features) and requires more memory (must store all the data).
- Eager learner does the work off-line

Non-parametric method

- Representation for parametric method is specified in advance
  - Fixed size representation
- Representation for non-parametric methods depends on dataset
  - Size of representation typically linear in # of examples

Pros and cons of non-parametric method

- Representation for parametric method is specified in advance
  - Good if a good representation is known in advance
  - Can easily leverage knowledge about structure
- Representation for non-parametric methods depends on dataset
  - High resolution where much data available / decisions are complex
  - If little is known data distribution (no good representation known)
  - Still requires a good distance metric
- Non-parametric methods often require complex computations
- Non-parametric methods typically larger storage requirement

Lazy / eager and non-parametric

- Lazy / eager: Generalization before or after seeing query?
- Parametric or not: fixed # of parameters or determined by data?

- Usually, parametric methods are also eager
- Often, non-parametric are also lazy
  - But consider decision trees!

When to use instance-based learning

- Instances map to points in R^n . Or else a given distance metric.
- Not too many attributes per instance (e.g. <20), otherwise all points look at a similar distance, and noise becomes a big issue.
- Not too many irrelevant attributes: easily fooled!
  (for most distance metrics.)
- Structure of model not known in advance
- Uneven spread of data: Provides variable resolution approximation (based on density of points).

Application

Hays & Efros, Scene Completion Using Millions of Photographs, CACM, 2008.

http://graphics.cs.cmu.edu/projects/scene-completion/scene_comp_cacm.pdf

Original Input Alternative completions

What you should know

- Difference between **eager** vs **lazy** learning.
- Key idea of **non-parametric** learning.
- The **k-nearest neighbor** algorithm for classification and regression, and its properties.
- The distance-weighted NN algorithm

What you should know

- Difference between **eager** vs **lazy** learning.
- Key idea of **non-parametric** learning.
- The **k-nearest neighbor** algorithm for classification and regression, and its properties.
- The distance-weighted NN algorithm and locally-weighted linear regression.

Project 1 follow-up

- Please follow instructions carefully!
  - I spent ~5 hours since Friday cleaning up your submissions.
  - Some submitted by email a few minutes/seconds late.
  - Some submitted a single tar (w/report, predictions, code).
  - Some did not include their collaborators as co-authors.
  - Some could not compress their code sufficiently.
  - SUBMIT EARLY!   SUBMIT OFTEN!

Project 2

- Available today. **Due Oct. 23rd.**

- **Text classification task:**
  - Devise a machine learning algorithm to analyze short conversations and automatically classify them according to the language of the conversation.
  - Conversations taken from your collected corpuses

Tips for analyzing text

- Natural Language toolkit: http://www.nltk.org/
- Common features?
  - **Bag of words**

Document 1

The quick brown fox jumped over the lazy dog's back.

Document 2

Now is the time for all good men to come to the aid of their party.

Term	Document 1	Document 2
aid	0	1
all	0	1
back	1	0
brown	1	0
come	0	1
dog	1	0
fox	1	0
good	0	1
jump	1	0
lazy	1	0
men	0	1
now	0	1
over	1	0
party	0	1
quick	1	0
their	0	1
time	0	1

Stopword List

Stopword List
for
is
of
the
to

Tips for analyzing text

- Natural Language toolkit: http://www.nltk.org/
- Common features?
  - Bag of words
  - **Term frequency – inverse document frequency (TF-IDF)**

TF*(t,d)* = frequency of a word *t* in a document *d*

IDF(t,D) = measure of how much information the word t provides across corpus of documents *D*

TF-IDF(*t,d,D*) = TF(*t,d*) x IDF(*t,D*)

Tips for analyzing text

- Natural Language toolkit: http://www.nltk.org/
- Common features?
  - Bag of words
  - Term frequency – inverse document frequency (TF-IDF)
  - **Hashing**

    => Turn a word into a fixed-length vector using a hashing function.
  - **Word embeddings** (*more on this later in the course.)*

- **Dimensionality reduction:** don't consider all words, limit size of hash table / embedding dimension. (*more on this also later.)*

Locally weighted regression