K-nearest-neighbor: an introduction to machine learning

Xiaojin Zhu

jerryzhu@cs.wisc.edu

Computer Sciences Department

University of Wisconsin, Madison

Outline

- Types of learning
- Classification: features, classes, classifier
- K-nearest-neighbor, instance-based learning
- Distance metric
- Training error, test error
- Overfitting
- Held-out set, k-fold cross validation, leave-one-out

machine learning

- Learn: to improve automatically with experience
  - speech recognition
  - autonomous vehicles
  - classify astronomical objects
  - play backgammon
  - predict heart attack
  - predict stock prices
  - filter spam
  - ...

there are 3 types of learning

1. supervised learning

   1. **classification**: You're given images of scanned handwritten digits (x) and their corresponding labels (y discrete). Design a ZIP-code reader for the post office to label new scanned digit images.
   2. **regression**: Given the infrared absorption spectrum (x) of diabetic people's blood, and the amount of glucose (y continuous) in the blood. Estimate glucose amount from a new spectrum.

   "supervised" because y was given during training.

x are called [instances | examples | points]. Each instance is represented by a [feature | attribute] vector.

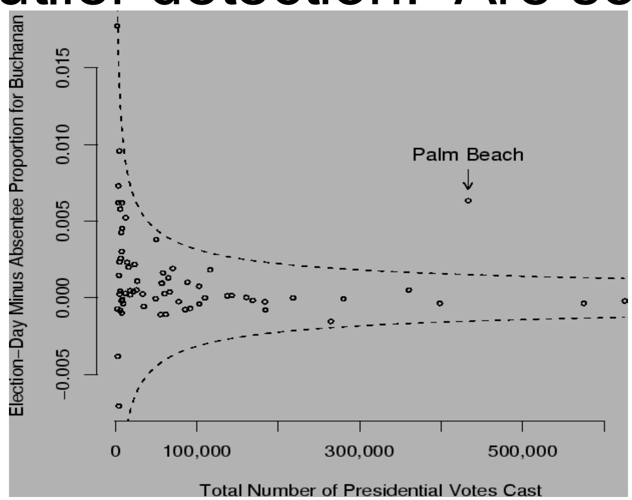
there are 3 types of learning

1. unsupervised learning
    - clustering: 'cut out' in digital photo: how can we group pixels (x) into meaningful regions (clusters)?

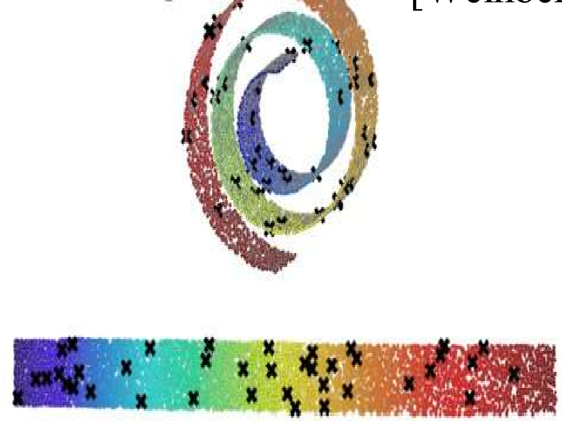
[Shi & Malik 2000]

    - outlier detection: Are some of x abnormal?

[Wand et al. 2001]



[Weinberger et al. 2005]



    - dimensionality reduction, and more...

"unsupervised" because we only have (x1...xn)

there are 3 types of learning

1. reinforcement learning

   world: you're in observable world state x1

   robot: i'll take action a1

   world: you receive reward r1.   now you're in state x2

   robot: i'll take action a2

   world: you receive reward r2. now you're in state x3

   ...

   What's the optimal action ***a*** when in state ***x***, in order to maximize life-long reward?

   "reinforcement" because learning is via

   action - reward

there are more than 3 types of learning

1. supervised learning: (x,y)
   - classification
   - regression
2. unsupervised learning: x
   - clustering, outlier detection, dimensionality reduction
3. reinforcement learning: (x,a,r)

5. semi-supervised learning: (x,y),x
6. multi-agent reinforcement learning as game playing: the world is full of other robots (recall the tragedy of the Commons)

...

the design cycle of classification

1. collect data and labels (the real effort)
2. choose features (the real ingenuity)
3. pick a classifier (some ingenuity)
4. train the classifier (some knobs, fairly mechanical)
5. evaluate the classifier (needs care)

If classifier no good, goto 1 — 4

1-nearest-neighbor (1NN) classifier

- Training set: $(x_1,y_1), (x_2,y_2), \ldots, (x_n,y_n)$
- Assume $x_i=(x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(d)})$ is a d-dimensional feature vector of real numbers, for all i
- y_i is a class label in $\{1\ldots C\}$, for all i
- Your task: determine y_{new} for x_{new}

1NN algorithm:

8. Find the closest point x_j to x_{new} w.r.t. Euclidean distance $[(x_j^{(1)} - x_{new}^{(1)})^2 + \ldots + (x_j^{(d)} - x_{new}^{(d)})^2]^{1/2}$
9. Classify by $y^{1nn} = y_j$

Instance-based learning

- 1NN is a special case of instance-based learning
  - Find similar instances to x_{new} in training data
  - Fit the label within the similar instances
  - 100-year old idea
- Variations in instance-based learning
  - How many neighbors to consider? (1 for 1NN)
  - What distance to use? (Euclidean for 1NN)
  - How to combine neighbors' labels? (the same)
- Instance-based learning methods which you should know the name, but we won't cover today:
  - Nearest-neighbor regression
  - Kernel regression, locally weighted regression

K-nearest-neighbor (kNN)

- Obvious extension to 1NN: consider k neighbors

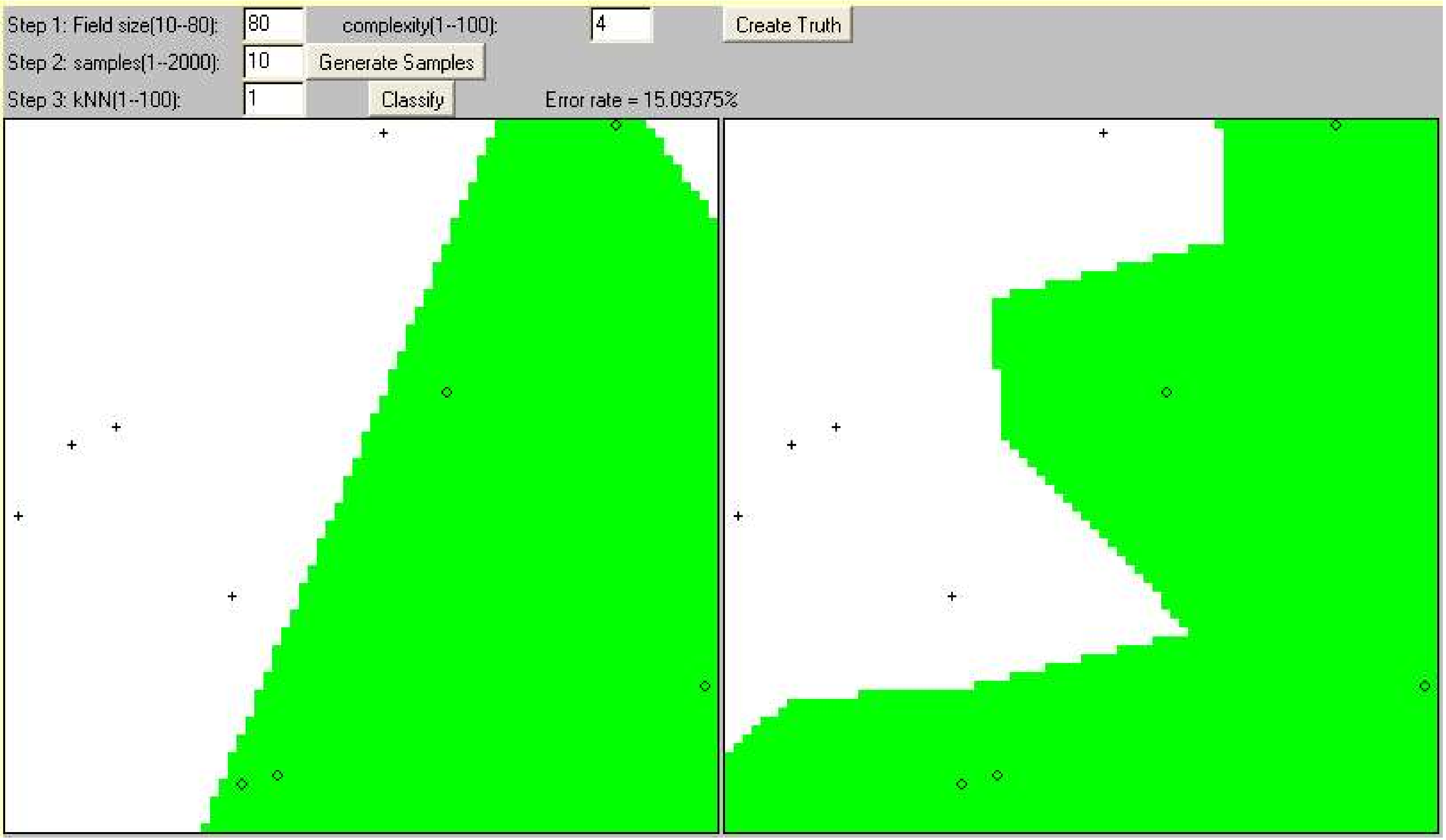
kNN algorithm:

5. Find k closest training points to x_{new} w.r.t. Euclidean distance
6. Classify by y^{knn} = majority vote among the k points

•How many neighbors to consider? (k)
•What distance to use? (Euclidean)
•How to combine neighbors' labels? (majority vote)

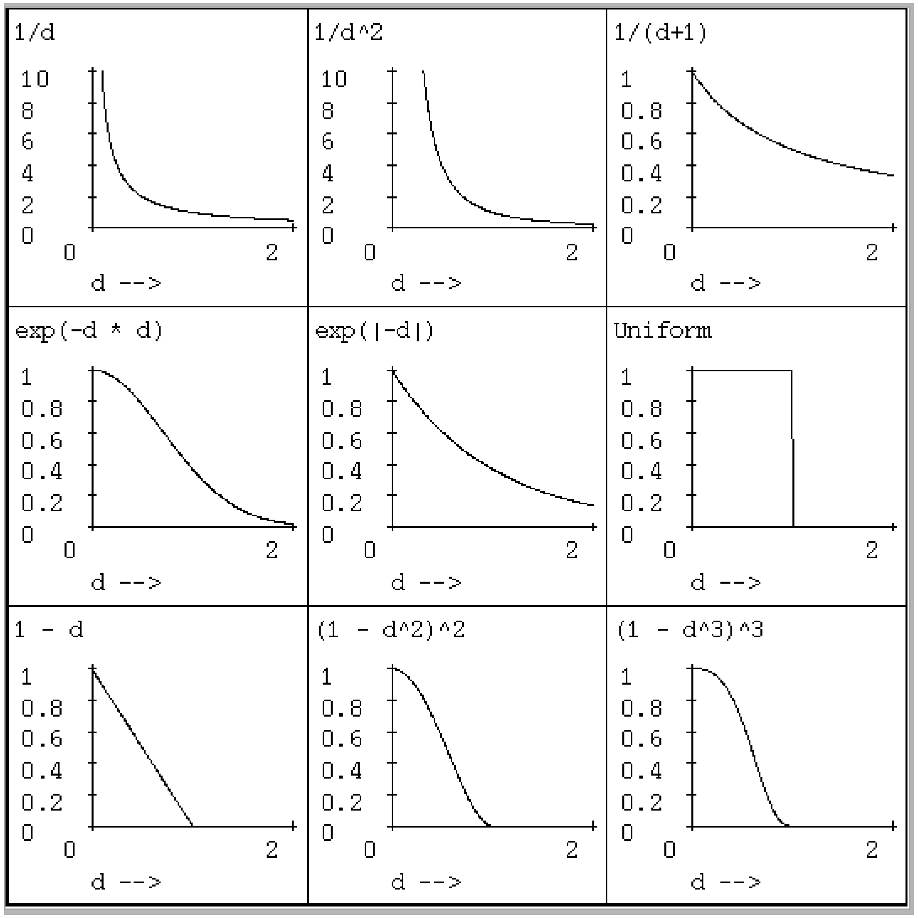
kNN demo

http://www.cs.cmu.edu/~zhuxj/courseproject/knndemo/KNN.html



Weighted kNN

- Near neighbors should count more than far neighbors
- Each neighbor cast vote with weight w_i, depending on the distance d. Many choices…



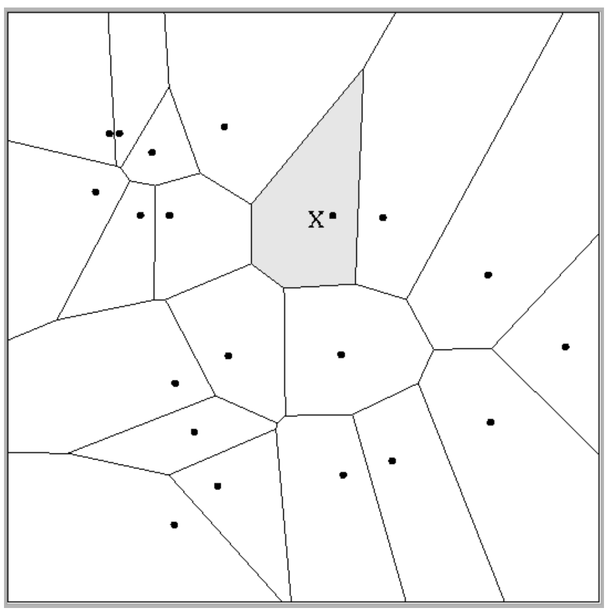
Copied from Andrew Moore

•How many neighbors to consider? (k)
•What distance to use? (Euclidean)
•How to combine neighbors' labels? (weighted majority vote)

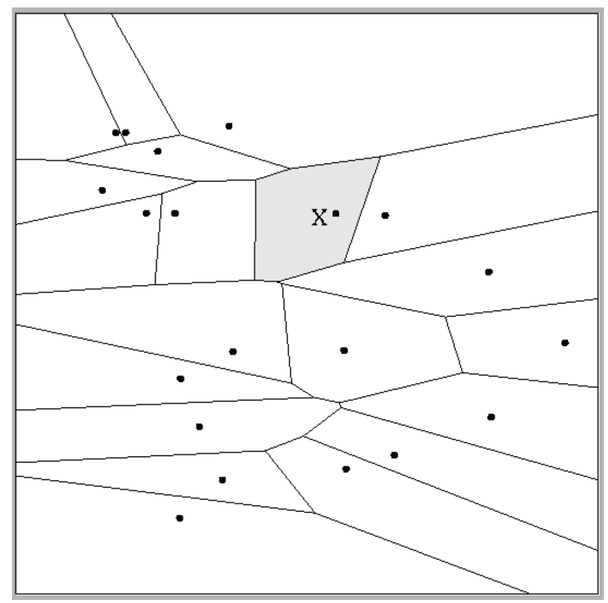
The distance metric

- The answer shouldn't change if one uses metric or English units for the features
- An Euclidean distance with different scaling factors ($a_1 \ldots a_d$) for different dimensions:

$$[a_1(x_j^{(1)} - x_{new}^{(1)})^2 + \ldots + a_d\,(x_j^{(d)} - x_{new}^{(d)})^2]^{1/2}$$



$Dist^2(\mathbf{x}_i,\mathbf{x}_j) = (x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2$

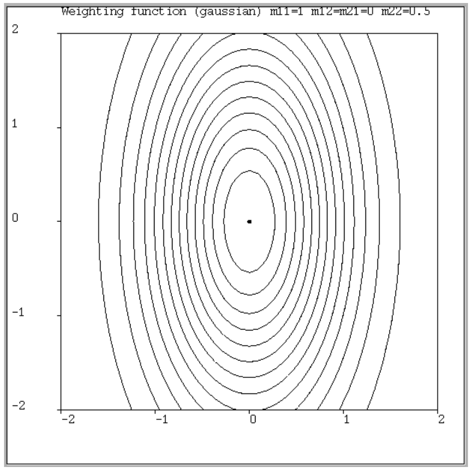


$Dist^2\,(\mathbf{x}_i,\mathbf{x}_j) = (x_{i1} - x_{j1})^2 + (3x_{i2} - 3x_{j2})^2$

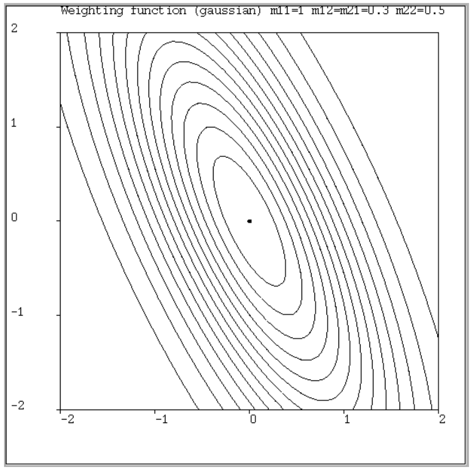
Copied from Andrew Moore

Some notable distance metrics

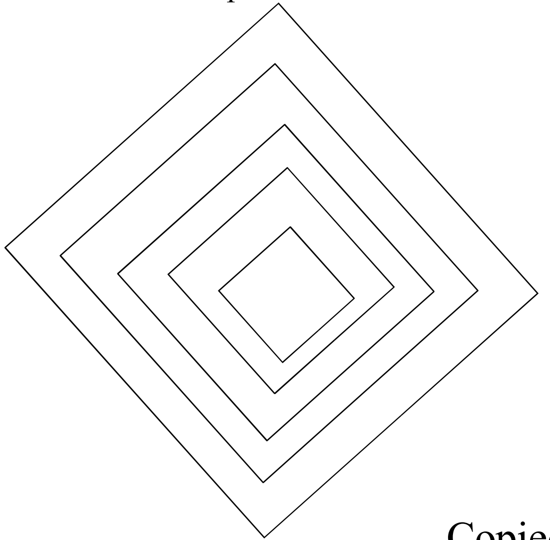
Scaled Euclidean (diagonal covariance)



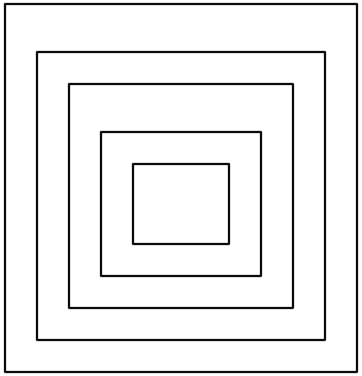
Mahalanobis (full covariance)



L_1 norm



L_∞(max) norm



Copied from Andrew Moore

Distance metric

- However choosing an appropriate distance metric is a hard problem

Mars climate orbiter, lost in 1999 due to a mix-up of metric and English units within NASA.

Important questions:

2. How to choose the scaling factors $(a_1 \ldots a_d)$?
3. What if some feature dimensions are simply noise??
4. How to choose K?

These are the parameters of the classifier.

Evaluate classifiers

- During training
  - You're given training set (x_1,y_1), (x_2,y_2), …, (x_n,y_n). You write kNN code. You now have a classifier, a black box which generates y^{knn} for any x_{new}
- During testing
  - For new test data $x_{n+1}\ldots x_{n+m}$, your classifier generates labels $y_{n+1}^{knn}\ldots\ y_{n+m}^{knn}$
- Test set accuracy: the correct performance measure
  - You need to know the true test labels $y_{n+1}\ldots\ y_{n+m}$
  - $acc = (\ \Sigma\ y_i^{knn}{=}{=}y_i)\ /\ m$, where $i=n+1, \ldots, n+m$
  - Test set error = 1 – acc

Parameter selection

- During training
  - You're given training set $(x_1,y_1), (x_2,y_2), \ldots, (x_n,y_n)$.
  - You now hav
  - es y_{new} for
  - x_{n+m}, your
    generates labels $y_{n+1}^{knn} \ldots y_{n+m}^{knn}$
- Test set accuracy: the correct performan measure
  - You need to know the true test labels y
  - acc = $(\Sigma\, y_i^{knn}==y_i) / m$, wher
  - Test set error = 1 – acc

Good idea: pick the parameters (e.g. K) that maximize test set accuracy

But most of the time we only have the training set, when we design a classifier.

Can we use training set accuracy instead?

Parameter selection by training set accuracy?

- You're given training set (x_1,y_1), (x_2,y_2), …, (x_n,y_n).
- You build a kNN classifier on them.
- You compute the training set accuracy

$$acc_{train} = (\Sigma\ y_i^{knn}==y_i) / n, \text{ where } i=1, \ldots, n$$

- You use acc_{train} to select K

There is something very wrong, can you see why?

Hint: consider K=1

Overfitting

- Overfitting is when a learning algorithm performs too good on the training set, compared to its true performance on unseen test data.

Never use training accuracy to select parameters, you'll overfit.

- Overfitting fits [noise | idiosyncrasy] in training data, not the general underlying regularity.
- Ideas?

Leave-one-out cross validation (LOOCV)

1. For each training example (x_i, y_i)
   - Train a classifier with all training data except (x_i, y_i)
   - Test the classifier's accuracy on (x_i, y_i)
2. LOOCV accuracy = average of all n accuracies

- Selected parameters based on LOOCV accuracy
- Easy to compute for kNN, but (as we'll see later) expensive for many other classifiers.

The holdout set method

1. Randomly choose, say 30%, of the training data, set it aside
2. Train a classifier with the remaining 70%
3. Test the classifier's accuracy on the 30%

- Selected parameters based on holdout set accuracy
- Easy to compute, but has higher variance (not as good an estimator of future performance). Waste data.

K-fold cross validation (CV)

1. Randomly divide training data into k chunks (folds)
2. For each fold:
   - Train a classifier with all training data except the fold
   - Test the classifier's accuracy on the fold
3. K-fold CV accuracy = average of all k accuracies

- Selected parameters based on k-fold CV accuracy
- Between LOOCV and holdout set. Often used. People tend to select k=10 or 5

These methods will not completely prevent overfitting.

What you should know

- Types of learning
- K-nearest-neighbor, instance-based learning
- Distance metric
- Training error, test error
- Overfitting
- Held-out set, k-fold cross validation, leave-one-out