k-Nearest Neighbor

Instructor: Jessica Wu -- Harvey Mudd College

The instructor gratefully acknowledges Eric Eaton (UPenn), David Kauchak (Pomona), and Andrew Moore (CMU), and the many others who made their course materials freely available online.

Data Representation

Learning Goals

- Describe how to represent complex data
- View data graphically

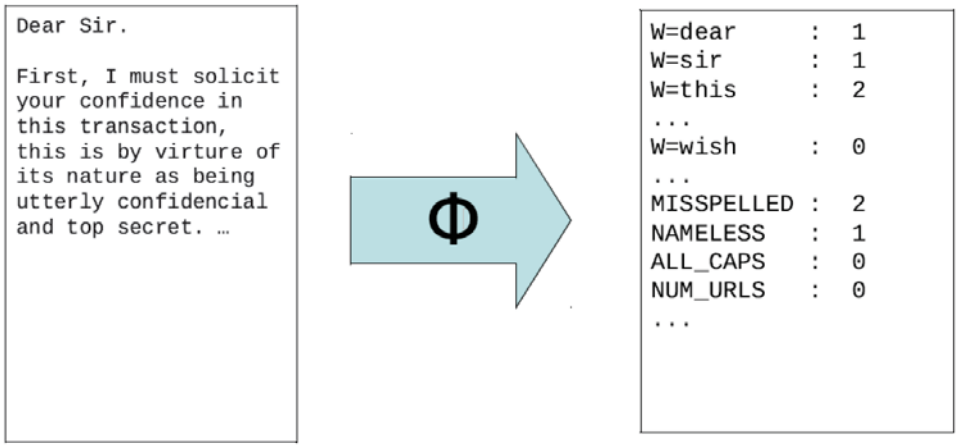
Data Representation

- Most learning algorithms require data in some numeric representation
  - e.g. each input pattern is a vector
- If data naturally has numeric (real-valued) features, just represent as vector of (real) numbers
  - e.g. 28x28 image by 784x1 vector of pixel intensities
- If data has non-numeric representation …
  - let's look at some examples

Based on slide by Piyush Rai

Data to Features

- Text document



Based on slide by Piyush Rai

Data to Features

Let's consider dataset similar to Tennis Playing example

Y	Out	T	R
P	Sunny	Low	Yes
N	Sunny	High	Yes
N	Sunny	High	No
P	Overcast	Low	Yes
P	Overcast	High	No
P	Overcast	Low	No
N	Rainy	Low	Yes
P	Rainy	Low	No

⇒

Y	⟨Out , T , R⟩
1	⟨ ? , 0 , 1 ⟩
0	⟨ ? , 1 , 1 ⟩
0	⟨ ? , 1 , 0 ⟩
1	⟨ ? , 0 , 1 ⟩
1	⟨ ? , 1 , 0 ⟩
1	⟨ ? , 0 , 0 ⟩
0	⟨ ? , 0 , 1 ⟩
1	⟨ ? , 0 , 0 ⟩

- Features are categorical (Low/High, Yes/No, Overcast/Rainy/Sunny, etc)
- Features with only 2 possible values
  - can be represented as 0/1
- Features with more than 2 possible values
  - can we map Sunny=0, Overcast=1, Rainy=2?

Based on slide by Piyush Rai

Data to Features

- Well, we could map Sunny=0, Overcast=1, Rainy=2…
- But such a mapping may not always be appropriate
  - imagine feature values being red, blue, green
  - red=0, blue=1, green=2 implies red more similar to blue than to green
- **Solution**: for feature with K > 2 possible values, create binary features, one for each possible value

Y	Out	T	R
P	Sunny	Low	Yes
N	Sunny	High	Yes
N	Sunny	High	No
P	Overcast	Low	Yes
P	Overcast	High	No
P	Overcast	Low	No
N	Rainy	Low	Yes
P	Rainy	Low	No

⇒

Y	⟨S? , O? , R? , T , R⟩
1	⟨ 1 , 0 , 0 , 0 , 1 ⟩
0	⟨ 1 , 0 , 0 , 1 , 1 ⟩
0	⟨ 1 , 0 , 0 , 1 , 0 ⟩
1	⟨ 0 , 1 , 0 , 0 , 1 ⟩
1	⟨ 0 , 1 , 0 , 1 , 0 ⟩
1	⟨ 0 , 1 , 0 , 0 , 0 ⟩
0	⟨ 0 , 0 , 1 , 0 , 1 ⟩
1	⟨ 0 , 0 , 1 , 0 , 0 ⟩

Data Visualization

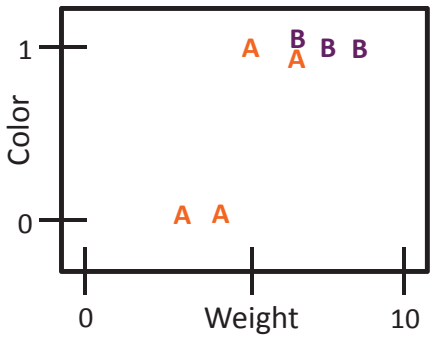
Let's visualize this data

Weight	Color	Label
4	Red	Apple
5	Yellow	Apple
6	Yellow	Banana
3	Red	Apple
7	Yellow	Banana
8	Yellow	Banana
6	Yellow	Apple

Turn features into numerical values

(simple mapping used here to keep visualization simple)

Weight	Color	Label
4	0	Apple
5	1	Apple
6	1	Banana
3	0	Apple
7	1	Banana
8	1	Banana
6	1	Apple



We can view examples as points in an *d*-dimensional space where *d* is number of features

Based on slide by David Kauchak

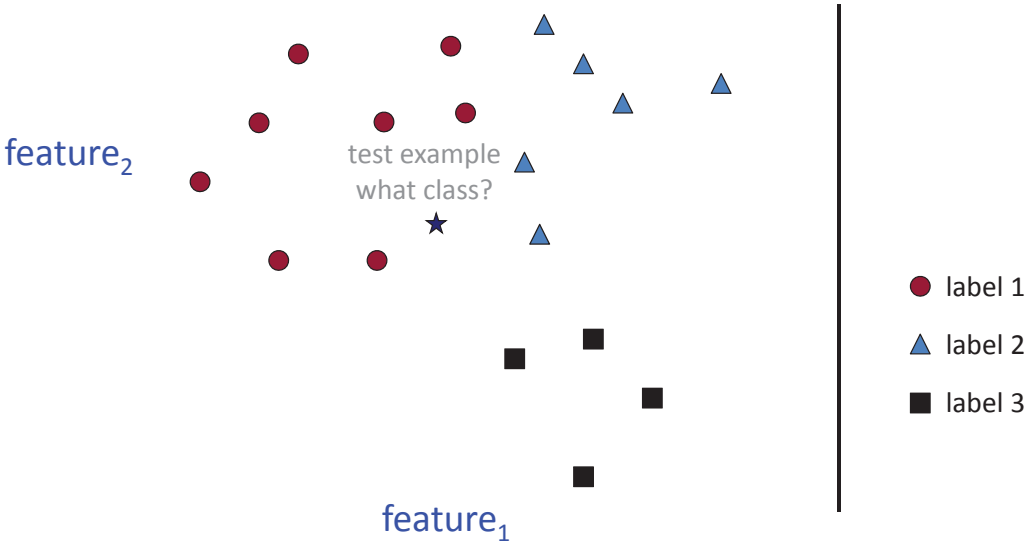
(This slide intentionally left blank.)

k-Nearest Neighbor

Learning Goals

- Describe kNN algorithm
- Describe impact of k in kNN
- Describe kNN variants (optional)

Examples in a Feature Space



Another classification algorithm?

To classify example ***x***:

Label ***x*** with label of closest example to ***x*** in training set

Based on slide by David Kauchak

k-Nearest Neighbor (*k*-NN)

To classify example ***x***:

- Find ***k*** *nearest* neighbors of ***x***
- Choose as label the majority label within ***k*** nearest neighbors

How do we measure "nearest"?

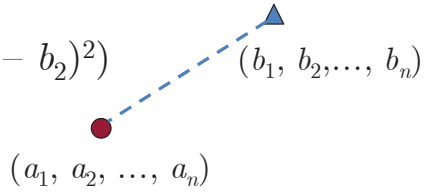
common approach : standard Euclidean distance metric

two-dimensional:

$$dist(\boldsymbol{a},\boldsymbol{b}) = \text{sqrt}((a_1 - b_1)^2 + (a_2 - b_2)^2)$$

n-dimensional:

$$dist(\boldsymbol{a},\boldsymbol{b}) = \text{sqrt}(\textstyle\sum_i (a_i - b_i)^2)$$



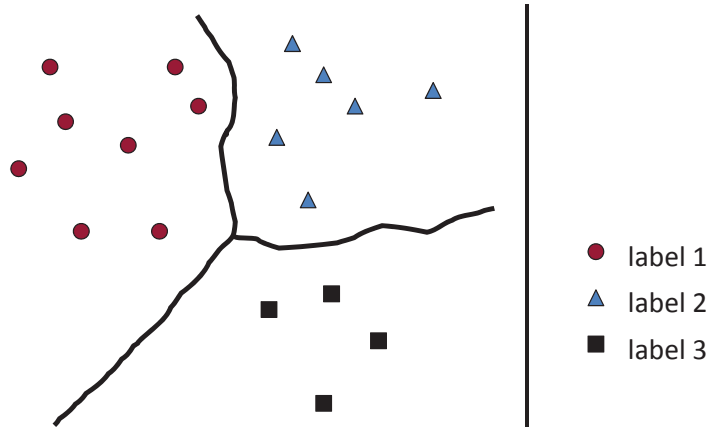
Based on slide by David Kauchak

Other Distance Measures

- **Binary**-valued features
  - Hamming distance $dist(\boldsymbol{a},\boldsymbol{b}) = \sum_i I(a_i \neq b_i)$ counts number of features where two examples disagree
- **Mixed** feature types (some real, some binary)
  - mixed distance measures
  - e.g. Euclidean for real part, Hamming for binary part
- Can also assign **weights** to features $dist(\boldsymbol{a},\boldsymbol{b}) = \sum_i w_i \cdot d(a_i,b_i)$

Based on slide by Piyush Rai

k-NN Decision Boundaries

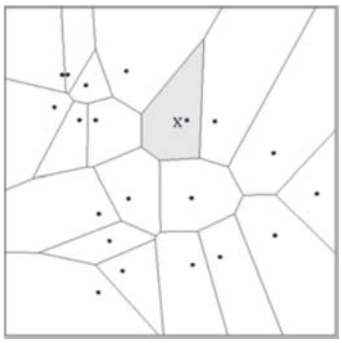


Where are decision boundaries for k-NN?
k-NN gives **locally defined** decision boundaries between classes
(forms subset of Voronoi diagram for training data)

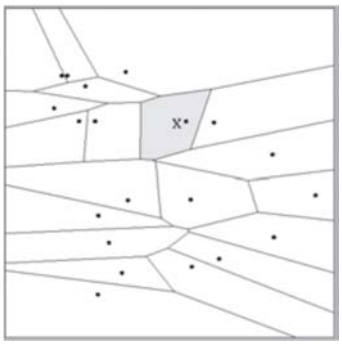
Based on slide by David Kauchak

k-NN Decision Boundaries

- Can be changed by different distance metrics



$dist(\boldsymbol{a},\boldsymbol{b}) = (a_1 - b_1)^2 + (a_2 - b_2)^2$



$dist(\boldsymbol{a},\boldsymbol{b}) = (a_1 - b_1)^2 + (3a_2 - 3b_2)^2$

- Become more complex as more examples are stored

Based on slide by Eric Eaton
(originally by Andrew Moore)

k-Nearest Neighbor (*k*-NN)

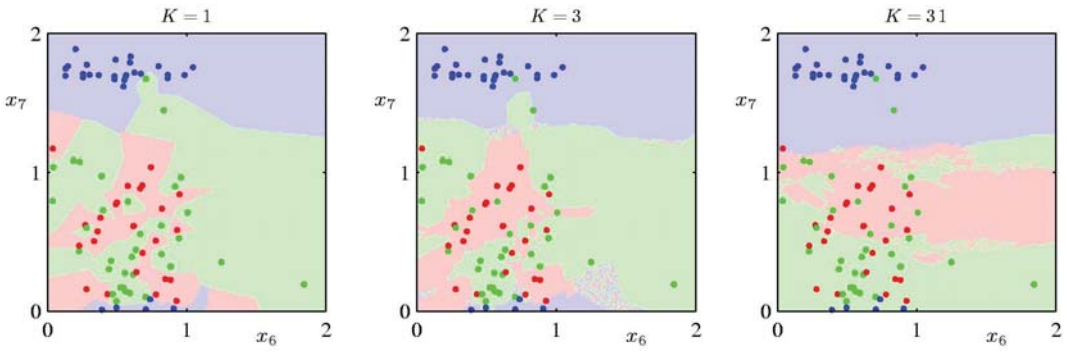
To classify example ***x***:

- Find ***k*** nearest neighbors of ***x***
- Choose as label the majority label within ***k*** nearest neighbors

How do we choose *k*?

Based on slide by David Kauchak

Impact of *k*



What is role of *k*?

How does it relate to overfitting and underfitting?

How did we control this for decision trees?

Based on slide by David Kauchak

k-Nearest Neighbor (*k*-NN)

To classify example ***x***:

- Find ***k*** nearest neighbors of ***x***
- Choose as label the majority label within ***k*** nearest neighbors

How do we choose *k*?

- Often data-dependent and heuristic-based
  - common heuristic : choose 3,5,7 (odd number)
- Use validation data
- In general, *k* too small or too big is bad

Based on slide by David Kauchak and Piyush Rai

k-Nearest Neighbor (*k*-NN)

To classify example ***x***:

- Find ***k*** nearest neighbors of ***x***
- Choose as label the majority label within ***k*** nearest neighbors

Any variants?

- Fixed distance
  - instead of *k*-NN, count majority from all examples within fixed distance (radius-based neighbors)
- Weighted
  - instead of treating all examples equally, weight "vote" of examples so that closer examples have more vote/weight (often use some sort of exponential decay)

Based on slide by David Kauchak

kNN Problems and ML Terminology

Learning Goals

- Describe how to speed-up kNN
- Define non-parametric and parametric and describe differences
- Describe curse of dimensionality

Speeding up *k*-NN

- *k*-NN is a **"lazy" learning algorithm**
  - does virtually nothing at training time
- But **classification / prediction can be costly** when training set is large
  - for *n* training examples and *d* features, how many computations required for each test example?
- Two strategies for alleviating this weakness
  - edited nearest neighbor : do not retain every training instance
  - k-d tree : use smart data structure to look up NN

Based on slide by David Page

Aside : Non-parametric vs Parametric

- **non-parametric** method
  - not based on parameterized families of probability distributions – make no assumptions about distributions of variables being assessed
  - complexity grows with amount of training data
  - (not *none*-parametric)

both DT and kNN are non-parametric

- **parametric** method
  - makes inferences about parameters of underlying data-generating distribution
  - has fixed number of parameters

Curse of Dimensionality

- Our intuitions about space/distance do not scale with dimensions!
- NN breaks down in high dimensional spaces because "neighborhood" becomes very large
- Ex: Suppose we have 5000 points uniformly distributed in unit hypercube and want to apply 5-NN to test example at origin
  - on average, need to explore 5/5000 = 0.001 of volume
  - 1D: must go distance of 0.001 on average
  - 2D: must go sqrt(0.001) ≈ 0.0316 to get square that contains 0.001 of volume
  - *n*-D : in *n* dimensions, must go $(0.001)^{1/n}$ >> 0.001

Based on slides by David Kauchak and David Sontag

Summary: *k*-NN

When to consider

- examples map to points in $\mathbb{R}^d$
- small number (<20) attributes per instance
- lots of training data

Advantages

- "training" is very fast
- simple to implement
- learn complex target functions
- adapts well to online learning
- do not lose information
- robust to noisy training data (when $k > 1$)

Disadvantages

- slow prediction
- easily fooled by irrelevant attributes
- sensitive to range of feature values
- **not much insight** into problem domain because no explicit model

Based on slides by David Sontag and David Page