L1: Entropy is a measure of the purity of a dataset (interval) S The higher the entropy, the lower the purity of the dataset

$$entropy(S) = -\sum_i P_i . \log_2 P_i$$ Pi - proportion of examples from class i

- Ex.: Consider a split between 70 and 71. What is the entropy of the left and right datasets (intervals)?
- values of temperature:

64	65	68	69	70	71	72	73	74	75	80	81	83	85
yes	no	yes	yes	yes	no	no	no	yes	yes	no	yes	yes	no

$$entropy(S_{left}) = -\frac{4}{5}\log_2\frac{4}{5} - \frac{1}{5}\log_2\frac{1}{5} = 0.722\,bits$$
$$entropy(S_{right}) = -\frac{4}{9}\log_2\frac{4}{9} - \frac{5}{9}\log_2\frac{5}{9} = 0.991\,bits$$

Total entropy of the split = weighted average of the interval entropies

$$totalEntropy = \sum_i^n w_i \, entropy(S_i)$$

w_i – proportion of values in interval i, n – number of intervals

Algorithm: evaluate all possible splits and choose the best one (with the lowest total entropy); repeat recursively until stopping criteria are satisfied (e.g. user specified number of splits is reached)

Normalization and standardization

Performed for each attribute

Normalization (also called min-max scaling):

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Standardization:

$$x' = \frac{x - \mu(x)}{\sigma(x)}$$

x – original value
x' – new value

x – all values of the attribute; a vector
min(x) and max(x) – min and max values of the attribute (of the vector x)
μ(x) - mean value of the attribute
σ(x) - standard deviation of the attribute

Euclidean and Manhattan distance

Distance measures for numeric attributes

- A, B – examples with attribute values $a_1, a_2, ..., a_n$ & $b_1, b_2, ..., b_n$
- E.g. A= [1, 3, 5], B=[1, 6, 9]

Euclidean distance (L2 norm) – most frequently used

$$D(A,B) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + ... + (a_n - b_n)^2}$$

D(A,B) = sqrt $((1-1)^2+(3-6)^2+(5-9)^2)=5$

Manhattan distance (L1 norm)

$$D(A,B) = |a_1 - b_1| + |a_2 - b_2| + ... + |a_n - b_n|$$

D(A,B)=|1-1|+|3-6|+|5-9|=7

Weighted distance – each attribute is assigned a weight according to its importance (requires domain knowledge)

- Weighted Euclidean:

$$D(A,B) = \sqrt{w_1|a_1 - b_1|^2 + w_2|a_2 - b_2|^2 + ... + w_n|a_n - b_n|^2}$$

Hamming distance = Manhattan for binary vectors

- Counts the number of different bits

$$D(A,B) = |a_1 - b_1| + |a_2 - b_2| + ... + |a_n - b_n|$$

A = [1 0 0 0 0 0 0 0 0 0]
B = [0 0 0 0 0 0 1 0 0 1]
D(A,B) = 3

Similarity coefficients

f00: number of matching 0-0 bits
f01: number of matching 0-1 bits
f10: number of matching 1-0 bits
f11: number of matching 1-1 bits

Calculate these coefficients for the example above!
Answer: f01 = 2, f10 = 1, f00 = 7 , f11 = 0

Minkowski distance – generalization of Euclidean & Manhattan

$$D(A,B) = (|a_1 - b_1|^q + |a_2 - b_2|^q + ... + |a_n - b_n|^q)^{1/q}$$

q – positive integer

Simple Matching Coefficient (SMC) - matching 1-1 and 0-0 / num. attributes

SMC = (f11+f00)/(f01+f10+f11+f00)

Ex.: A = [1 0 0 0 0 0 0 0 0 0]
B = [0 0 0 0 0 0 1 0 0 1]
f01 = 2, f10 = 1, f00 = 7 , f11 = 0
SMC = (0+7) / (2+1+0+7) = 0.7

An alternative: Jaccard coefficient

- counts matching 1-1 and ignores matching 0-0

J=f11/(f01+f10+f11)

A = [1 0 0 0 0 0 0 0 0 0]
B = [0 0 0 0 0 0 1 0 0 1]
f01 = 2, f10 = 1, f00 = 7 , f11 = 0
J = 0 / (2 + 1 + 0) = 0 (A and B are dissimilar)

Useful for sparse data (both binary and non-binary)
Widely used for classification of text documents

$$\cos(A,B) = \frac{A \bullet B}{\|A\|\|B\|}$$

- • - vector dot product, ||A|| - length of vector A

Geometric representation: measures the angle between A and B

- Cosine similarity=1 => angle(A,B)=0°
- Cosine similarity =0 => angle (A,B)=90°

Two document vectors:

d_1 = 3 2 0 5 0 0 0 2 0 0
d_2 = 1 0 0 0 0 0 0 1 0 2

$$\cos(A,B) = \frac{A \bullet B}{\|A\|\|B\|}$$

$d_1 \bullet d_2$ = 3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5
$||d_1||$ = $(3*3+2*2+0*0+5*5+0*0+0*0+0*0+0*0+2*2+0*0+0*0)^{1/2}$ = $(42)^{1/2}$ = 6.481
$||d_2||$ = $(1*1+0*0+0*0+0*0+0*0+0*0+0*0+0*0+1*1+0*0+2*2)^{1/2}$ = $(6)^{1/2}$ = 2.245
=> $\cos(d_1, d_2)$ = 0.3150

Pearson correlation coefficient between data objects (instances) x and y with dimensionality n

$$\text{corr}(\mathbf{x},\mathbf{y}) = \frac{\text{covar}(\mathbf{x},\mathbf{y})}{\text{std}(\mathbf{x})\,\text{std}(\mathbf{y})}$$

where:

$$mean(\mathbf{x}) = \frac{\sum_{k=1}^{n} x_k}{n} \qquad std(\mathbf{x}) = \sqrt{\frac{\sum_{k=1}^{n}(x_k - mean(\mathbf{x}))^2}{n-1}}$$

$$\text{co var}(\mathbf{x},\mathbf{y}) = \frac{1}{n-1}\sum_{k=1}^{n}(x_k - mean(x))(y_k - mean(y))$$

Range: [-1, 1]

- -1: perfect negative correlation
- +1: perfect positive correlation
- 0: no correlation

L2 KNN:

•categorical (nominal) - their values belong to a pre-specified, finite set of possibilities
• numeric (continuous) - their values are numbers

What will be the prediction of the Nearest Neighbor algorithm using the Euclidean distance for the following new example: a1=2, a2=4, a3=2?

	a1	a2	a3	class
1	1	3	1	yes
2	3	5	2	yes
3	3	2	2	no
4	5	2	3	no

D(new, ex1) = sqrt[$(2-1)^2$+$(4-3)^2$+$(2-1)^2$]=sqrt(3) yes
D(new, ex2) = sqrt[$(2-3)^2$+$(4-5)^2$+$(2-2)^2$]=sqrt(2) yes
D(new, ex3) = sqrt[$(2-3)^2$+$(4-2)^2$+$(2-2)^2$]=sqrt(5) no
D(new, ex4) = sqrt[$(2-5)^2$+$(4-2)^2$+$(2-3)^2$]=sqrt(14) no

The closest nearest neighbor is ex. 2, hence the nearest neighbor algorithm predicts class=yes for the new example

Training
Classification (prediction for a new example)
• Compare each unseen example with each training example
• If m training examples with dimensionality n => lookup for 1 unseen example takes m*n computations, i.e. O(mn)
Variations more efficiently: KD-trees & ball trees

Choice of k
K-Nearest Neighbor is very sensitive to to the value of k
• rule of thumb: k ≤ sqrt(#training_examples)
• commercial packages typically use k=10
•more nearest neighbors increases the robustness to noisy examples
also for **regression** : average value of the class values (numerical) of the k nearest neighbours

Nominal Data:
difference = 0 if attribute values are the same
difference = 1 if they are not

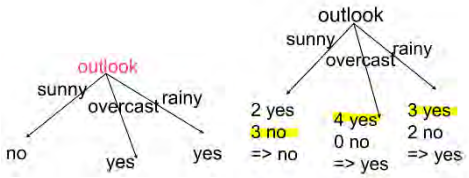
Example: 2 attributes = temperature and windy
temperature values: low and high **windy** values: yes and no ex.1= {high, no} ex.2 = {high, yes} d(A,B) =$(0+1)^{1/2}$=1 (Euclidean distance)

Weighted nearest neighbor
Idea: Closer neighbors should count more than distant neighbors
• Distance-weighted nearest-neighbor algorithm
• Find the k nearest neighbors
• Weight their contribution based on their distance to the new example
• bigger weight if they are closer
• smaller weight if they are further
• e.g. the vote can be weighted according to the distance – weight w = $1/distance^2$

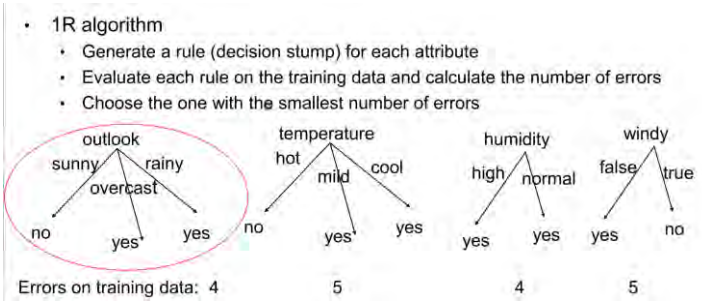
Decision boundary: Each training example has an associated **Voronoi region**; it contains the data points for which this is the closest example

Discussion:
- Often very accurate
- Slow for big datasets
- Distance-based - **requires normalization**
- Not effective for high-dimensional data (data with many features) -Solution – dimensionality reduction and feature selection
- Sensitive to the value of k

1-rule
Generates 1 rule that tests the value of a single attribute



Which is the **best rule (i.e. the best attribute)**?
• The one with the **smallest error rate** (i.e. with the highest accuracy) on training data



Final rule - rule 1:
if outlook=sunny then play=no
elseif outlook=overcast then play=yes
elseif outlook=rainy then play=yes

1R – discussion
• Simple and efficient algorithm, easy to understand
• Numerical datasets require discretization
• 1R has an in-built procedure to do this

Rule-Based Algorithms: PRISM - rule-based covering algorithm – Accuracy on training data always 100%
consider each class in turn and
• construct a set of if-then rules that cover all examples from this class and do not cover any examples from the other classes

Which test to add at each step?
The one that maximizes accuracy p/t:
• t: total number of examples (from all classes) covered by the rule (t comes from total)
• p: examples from the class under consideration, covered by the rule (p comes from positive)
• t-p: number of errors made by the rule
• Select the test that maximises the accuracy p/t

- Start with an empty rule: `if ? then recommendation = hard`

p/t (accuracy)

- 9 possible tests for the 4 attributes based on num. attribute values (3+2+2+2):

age = young	2/8
age= pre-presbyoptic	1/8
age = presbyoptic	1/8
spectacle prescription = myope	3/12
spectacle prescription = hypermetrope	1/12
astigmatism = no	0/12
astigmatism = yes	**4/12**
tear production rate = reduced	0/12
tear production rate = normal	4/12

age=young in 8 ex. and in 2 of them class=hard

- Best test (highest accuracy): `astigmatism = yes`
- Note that there is a tie: both `astigmatism = yes` and `tear production rate = normal` have the same accuracy 4/12; we choose the first one randomly

Current rule

`if astigmatism = yes then recommendation = hard`

Not "perfect" - covers 12 examples but only 4 of them are from class `hard` => refinement is needed

age	spectacle prescription	astigmatism	tear production rate	recommended lenses
young	myope	yes	reduced	none
young	myope	yes	normal	hard
young	hypermetrope	yes	reduced	none
young	hypermetrope	yes	normal	hard
pre-presbyopic	myope	yes	reduced	none
pre-presbyopic	myope	yes	normal	hard
pre-presbyopic	hypermetrope	yes	reduced	none
pre-presbyopic	hypermetrope	yes	normal	none
presbyopic	myope	yes	reduced	none
presbyopic	myope	yes	normal	hard
presbyopic	hypermetrope	yes	reduced	none
presbyopic	hypermetrope	yes	normal	none

- Further refinement by adding tests:
  `if astigmatism = yes and ? then recommendation = hard`
- Possible tests:

age = young	2/4
age= pre-presbyoptic	1/4
age = presbyoptic	1/4
spectacle prescription = myope	3/6
spectacle prescription = hypermetrope	1/6
tear production rate = reduced	0/6
tear production rate = normal	**4/6**

- Best test: `tear production rate = normal`

Current rule:

`if astigmatism = yes & tear production = normal then recommendation = hard`

Examples covered by the rule

age	spectacle prescription	astigmatism	tear production rate	recommended lenses
young	myope	yes	normal	hard
young	hypermetrope	yes	normal	hard
pre-presbyopic	myope	yes	normal	hard
pre-presbyopic	hypermetrope	yes	normal	none
presbyopic	myope	yes	normal	hard
presbyopic	hypermetrope	yes	normal	none

The rule is again not "perfect" – 2 examples classified as `none` => further refinement is needed

Further refinement:

`if astigmatism = yes & tear production = normal and ? then recommendation = hard`

Possible tests

age = young	2/2
age= pre-presbyoptic	1/2
age = presbyoptic	1/2
spectacle prescription = myope	**3/3**
spectacle prescription = hypermetrope	1/3

Best test: tie between the 1st and 4th; choose the one with the greater coverage (4th)

New rule:

`if astigmatism = yes & tear production = normal & spectacle prescription = myope then recommendation = hard`

Does the rule cover all hard examples? No, only 3/4, so we will need another rule

- Delete these 3 examples and start again
- Stop as all examples from class hard are covered
- Follow the same process for the other 2 classes (soft and none)

L3 Linear Regression :

Prediction error/residual

- Prediction error (residual) = observed-predicted value = $= \varepsilon = y_i - \hat{y}_i$

Performance index: sum of squared prediction errors (SSE): $SSE = \sum_i (y_i - \hat{y}_i)^2$

Our goal: select the line which minimizes SSE

- Can be solved using the **method of the least Squares**

The least squares method finds the best fit to the data but doesn't tell us how good this fit it

- E.g. SSE=12; is this large or small?

R^2 measures the *goodness of fit* of the regression line found by the least squares method:

$$R^2 = \frac{SSR}{SST}$$

Values between 0 and 1; the higher the better

- = 1: the regression line fits perfectly the training data
- close to 0: poor fit

SST = SSR + SSE

SSE: sum of squared prediction errors (actual – predict)

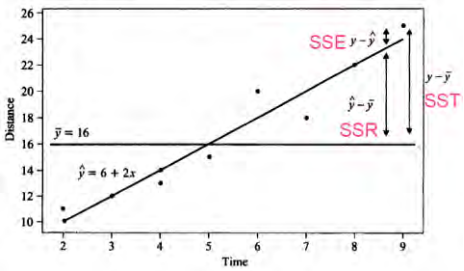
SST: sum of squared total errors (actual – mean)

$$SST = \sum_{i=1}^{n} (y_i - \bar{y})^2$$ = actual value – mean value

$$SST = \sum_{i}^{n} (y_i - \bar{y})^2 = (n-1)\,\mathrm{var}(y)$$ Can be used as a baseline - predicting y without knowing x

SSR: sum of squared regression errors (predict – mean)

$$SSR = \sum_{i}^{n} (\hat{y}_i - \bar{y})^2$$



r - **correlation coefficient**; measures linear relationship between 2 vectors x and y (positive relationship or negative) $r = \pm\sqrt{R^2}$

R^2 – **coefficient of determination**; measures how well the regression line represents the data . in multiple regression, R^2 : multiple coefficient of determination

Mean Absolute Error (MAE): $MAE = \frac{1}{n}\sum_{i=1}^{n} |\hat{y}_i - y_i|$

Mean Squared Error (MSE): $MSE = \frac{1}{n}\sum_{i=1}^{n} (\hat{y}_i - y_i)^2$

Root Mean Squared Error (RMSE):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}$$