Machine Learning

Measuring Distance

Why measure distance?

- Clustering requires distance measures.
- Local methods require a measure of "locality"
- Search engines require a measure of similarity

What is a "metric"?

- A function of two values with these four qualities.

$$d(x,y)=0 \quad \text{iff} \quad x=y \qquad \text{(reflexivity)}$$

$$d(x,y)\geq 0 \qquad \text{(non-negative)}$$

$$d(x,y)=d(y,x) \qquad \text{(symmetry)}$$

$$d(x,y)+d(y,z)\geq d(x,z) \qquad \text{(triangle inequality)}$$

What is a norm ||v|| ?

- Loosely, it is a function that applies a positive value to all vectors (except the 0 vector) in a vector space.

- 3 properties:

For all $a \in F$ and $u,v \in V$, a function $p : V \rightarrow F$

$p(av) = |a|\, p(v)$ (positive scalability)

$p(u) = 0$ iff u is the zero vector

$p(u) + p(v) \geq p(u+v)$ (triangle inequality)

2 definitions (AKA why this is confusing)

- A vector norm

  assigns a strictly positive value to all vectors v in a vector space….except the 0 vector, which has a 0 assigned to it. (see previous slide)

$$||v|| \geq 0$$

- A normal vector

  A vector is **normal** to another object if they are perpendicular to each other. So, a **normal vector** is perpendicular to the tangent plane of a surface at some point *P*.

Metric == Norm??

- Every norm determines a metric.

  Given a normed vector space, we can make a metric by saying

$$d(\mathbf{u}, \mathbf{v}) \equiv \|\mathbf{u} - \mathbf{v}\|$$

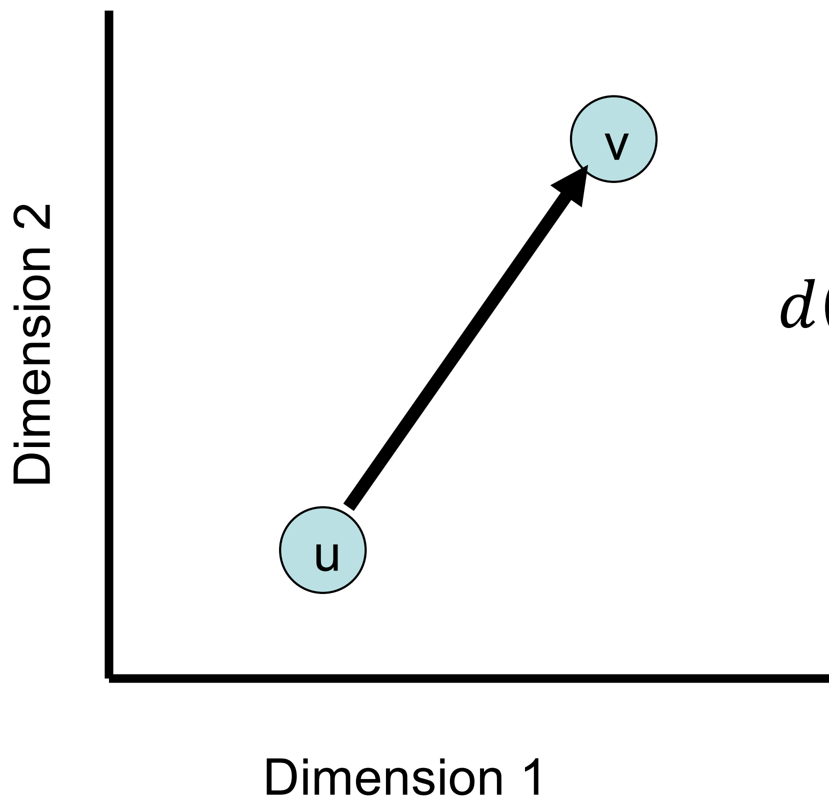
- Some metrics determine a norm.

  <u>If</u> the metric is on a vector space, you can define a norm by saying…

$$\|\mathbf{u}\| \equiv d(\mathbf{u}, \mathbf{0})$$

Euclidean Distance

- What people intuitively think of as “distance”
- Is it a metric?
- Is it a norm?



$$d(\mathbf{u}, \mathbf{v}) = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2}$$

Generalized Euclidean Distance

n = the number of dimensions

$$d(\mathbf{u},\mathbf{v}) = \left[\sum_{i}^{n} |u_i - v_i|^2\right]^{1/2}$$

Where…

$$\mathbf{u} = [u_1, u_2, u_3, \ldots, u_n]$$

$$\mathbf{v} = [v_1, v_2, v_3, \ldots, v_n]$$

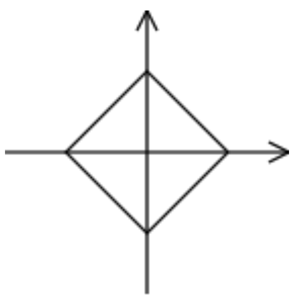
$$\mathbf{u}, \mathbf{v} \in \mathcal{R}^n$$

L^p norms

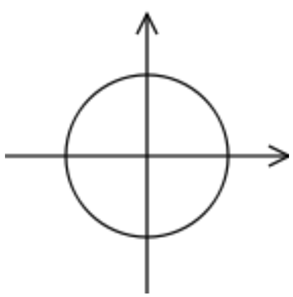
- L^p norms are all special cases of this:

$$d(\mathbf{u},\mathbf{v}) = \left[\sum_{i}^{n} |u_i - v_i|^p\right]^{1/p}$$

p changes the norm



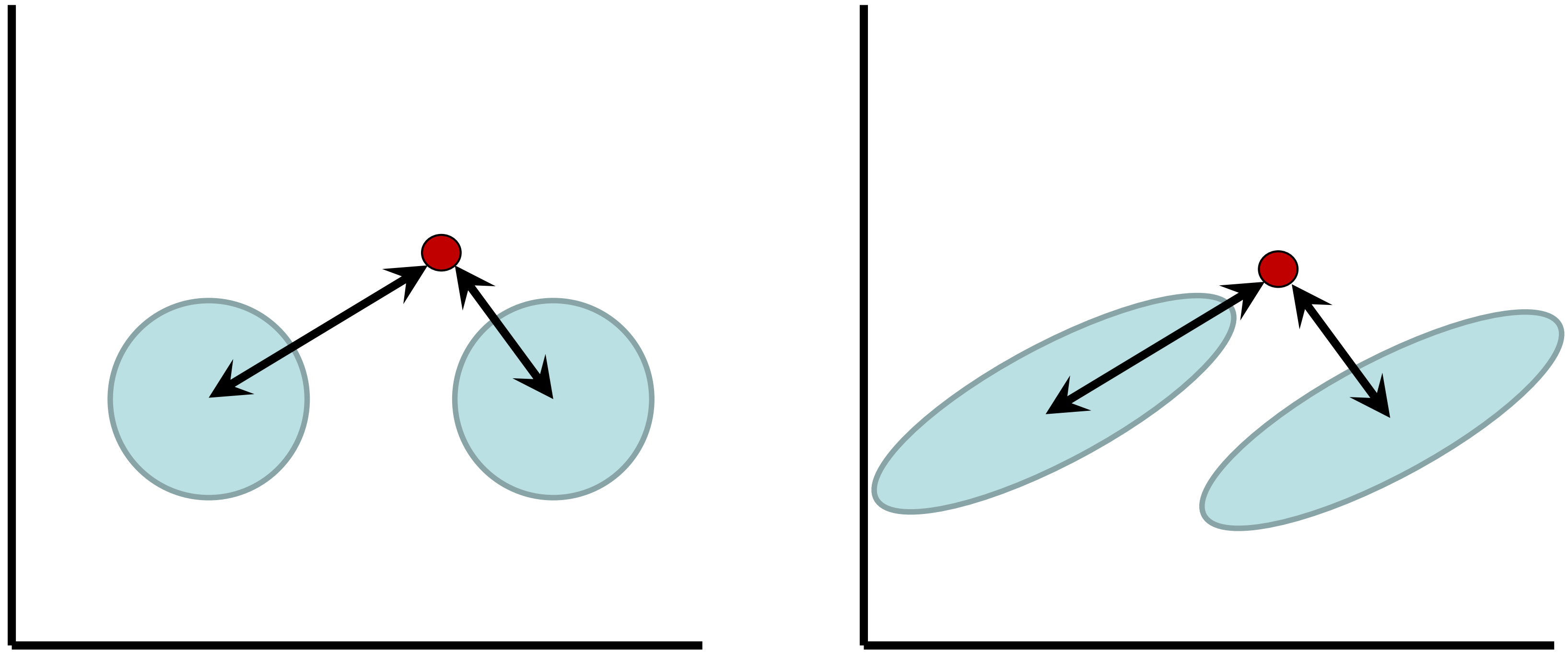
$$\|\mathbf{u}\|_1 = \mathrm{L}^1 \text{ norm} = \text{Manhattan Distance}: p = 1$$



$$\|\mathbf{u}\|_2 = \mathrm{L}^2 \text{ norm} = \text{Euclidean Distance}: p = 2$$

$$\text{Hamming Distance}: p = 1 \text{ and } \forall i, u_i, v_i \in \{0,1\}$$

Weighting Dimensions



- Put point in the cluster with the closest center of gravity
- Which cluster <u>should</u> the red point go in?
- How do I measure distance in a way that gives the “right” answer for both situations?

Weighted Norms

- You can compensate by weighting your dimensions....

$$d(\mathbf{u},\mathbf{v}) = \left[\sum_{i}^{n} w_i |u_i - v_i|^p\right]^{1/p}$$

This lets you turn your circle of equal-distance into an elipse with axes parallel to the dimensions of the vectors.

Cosine Similarity $s(\mathbf{u}, \mathbf{v})$

Sometimes, you don't want to think about magnitude of a vector, just the direction.

$$s(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u}^T \mathbf{v}}{||\mathbf{u}||_2 ||\mathbf{v}||_2}$$

$$= \frac{\sum_{i=1}^{n}(u_i)(v_i)}{\sqrt{\sum_{i=1}^{n}(u_i - 0)^2}\sqrt{\sum_{i=1}^{n}(v_i - 0)^2}}$$

$$||\mathbf{u}||_2 = d(\mathbf{u}, \mathbf{0}) = \sqrt{\sum_{i=1}^{n}(u_i - 0)^2}$$

Cosine Distance

$$s(\mathbf{u},\mathbf{v}) = \frac{\mathbf{u}^T\mathbf{v}}{||\mathbf{u}||_2||\mathbf{v}||_2}$$

Cosine similarity goes as low as -1 and maximizes at 1, when x == y.

To make it a distance measure (but still not a metric), make sure goes down when things are more similar and that the most similar pair gets a distance of 0

$$cosine\ distance \qquad d(\mathbf{u},\mathbf{v}) = 1 - s(\mathbf{u},\mathbf{v})$$

Cosine Distance

- What is the distance to the 0 vector?
- What is the distance between 2 vectors with the same angle, but different magnitudes?
- How do these things relate to the definition of being a metric?

Pearson Correlation Coefficient

- Measure of correlation between two variables
- Related to, but not identical to cosine similarity
- Pearson correlation coefficient

  Range (-1, 1)

  A perfect positive correlation: 1

  A perfect negative correlation: -1

In Python,

```
>> import scipy.stats
>> scipy.stats.pearsonr(array1, array2)
```

Pearson Sample Correlation $r_{\mathbf{xy}}$

Mean: $\mu_{\mathbf{x}} = \frac{\sum_{i=1}^{n} x_i}{n}$

$$r_{\mathbf{xy}} = \frac{\sum_{i=1}^{n}(x_i - \mu_x)\left(y_i - \mu_y\right)}{\sqrt{\sum_{i=1}^{n}(x_i - \mu_x)^2}\sqrt{\sum_{i=1}^{n}\left(y_i - \mu_y\right)^2}}$$

Example correlations

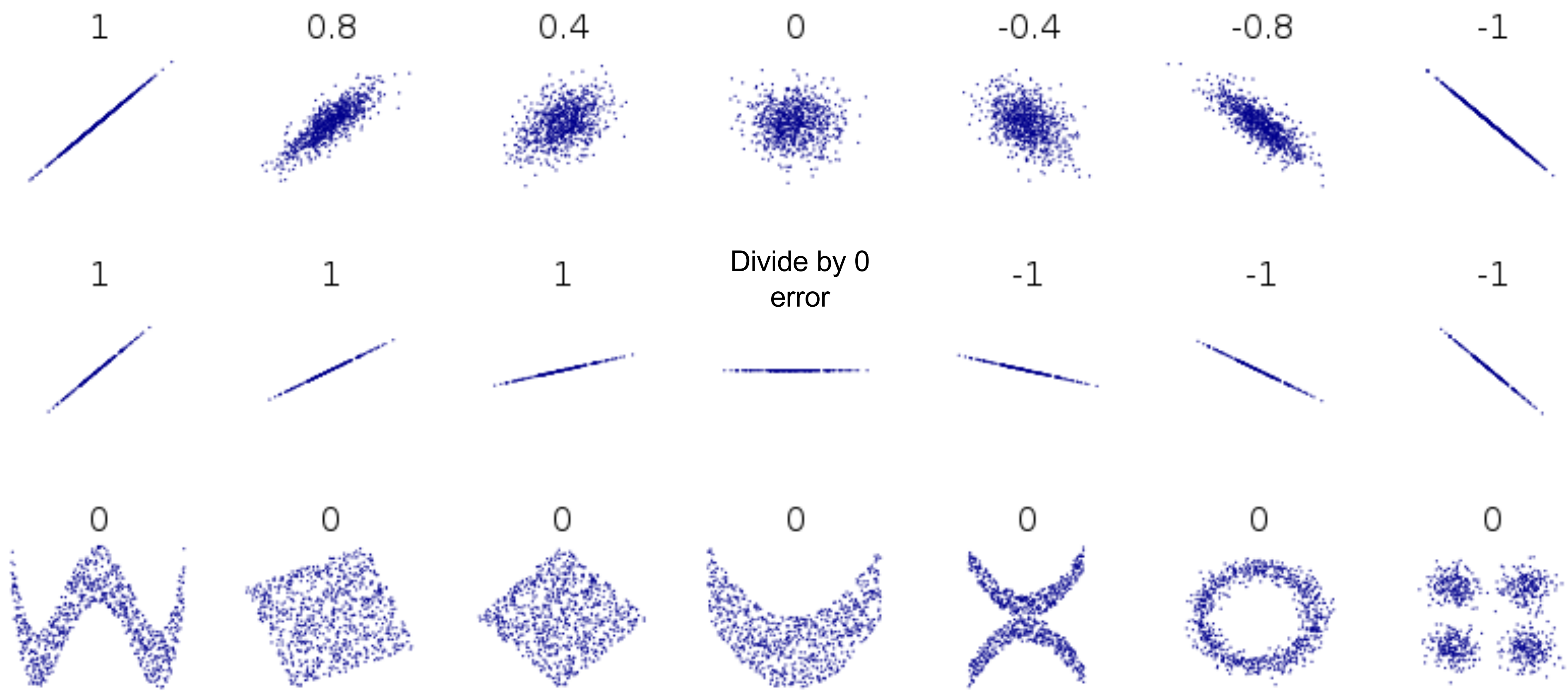


Image from Wikipedia: contribured by DenisBoigelot

Pearson vs Cosine

Pearson Correlation Coefficient

$$r_{\mathbf{xy}} = \frac{\sum_{i=1}^{n}(x_i - \mu_x)\left(y_i - \mu_y\right)}{\sqrt{\sum_{i=1}^{n}(x_i - \mu_x)^2}\sqrt{\sum_{i=1}^{n}\left(y_i - \mu_y\right)^2}}$$

Cosine Similarity, where we add in some 0s, so the relationship becomes clear

$$s(\mathbf{u},\mathbf{v}) = \frac{\sum_{i=1}^{n}(u_i - 0)(v_i - 0)}{\sqrt{\sum_{i=1}^{n}(u_i - 0)^2}\sqrt{\sum_{i=1}^{n}(v_i - 0)^2}}$$

Metric, or not?

- Driving distance with 1-way streets

- Categorical Stuff :
  - Is distance (Jazz to Blues to Rock) no less than distance (Jazz to Rock)?

Categorical Variables

- Consider feature vectors for genre & vocals:
  - Genre: {Blues, Jazz, Rock, Zydeco}
  - Vocals: {vocals,no vocals}

s1 = {rock, vocals}

s2 = {jazz, no vocals}

s3 = { rock, no vocals}

- Which two songs are more similar?

One Solution:Hamming distance

Blues	Jazz	Rock	Zydeco	Vocals
0	0	1	0	1
0	1	0	0	0
0	0	1	0	0

s1 = {rock, vocals}

s2 = {jazz, no_vocals}

s3 = { rock, no_vocals}

Hamming Distance = number of bits different between binary vectors

Hamming Distance

$$d(\vec{x},\vec{y}) = \sum_{i=1}^{n} | x_i - y_i |$$

$$\text{where } \vec{x} =< x_1, x_2, \ldots, x_n >,$$

$$\vec{\mathrm{y}} =< y_1, y_2, \ldots, y_n \}$$

$$\text{and } \quad \forall i (x_i, y_i \in \{0,1\})$$

Defining your own distance (an example)

How often does artist *x* quote artist *y*?

Quote Frequency

	Beethoven	Beatles	Kanye
Beethoven	7	0	0
Beatles	4	5	0
Kanye	?	1	2

Let's build a distance measure!

Defining your own distance (an example)

	Beethoven	Beatles	Kanye
Beethoven	7	0	0
Beatles	4	5	0
Kanye	?	1	2

$$\text{Quote frequency } Q_f(x, y) = \text{value in table}$$

$$\text{Distance } d(x, y) = 1 - \frac{Q_f(x, y)}{\sum_{z \in Artists} Q_f(x, z)}$$

Missing data

- What if, for some category, on some examples, there is no value given?

- Approaches:
  - Discard all examples missing the category
  - Fill in the blanks with the mean value
  - Only use a category in the distance measure if both examples give a value

(one way of) handling missing attributes

$$w_i = \begin{cases} 0, & \text{if both } x_i \text{ and } y_i \text{ are defined} \\ 1, & \text{else} \end{cases}$$

$$d(\vec{x}, \vec{y}) = \frac{n}{n - \sum_{i=1}^{n} w_i} \left[\sum_{i=1}^{n} \phi(x_i, y_i) \right]$$

A scaling factor that adds weight to the distance, as there are fewer attributes used

A distance measure that works on individual attributes

One more distance measure

- Kullback–Leibler (KL) divergence
  - a non-symmetric measure of the difference between two probability distributions
  - not a metric, since it is not symmetric
  - Here's the definition of KL divergence for discrete probability distributions P and Q

$$D_{KL}(P \,||\, Q) = \sum_i \ln\left(\frac{P(i)}{Q(i)}\right) P(i)$$

KL Divergence as Cross Entropy

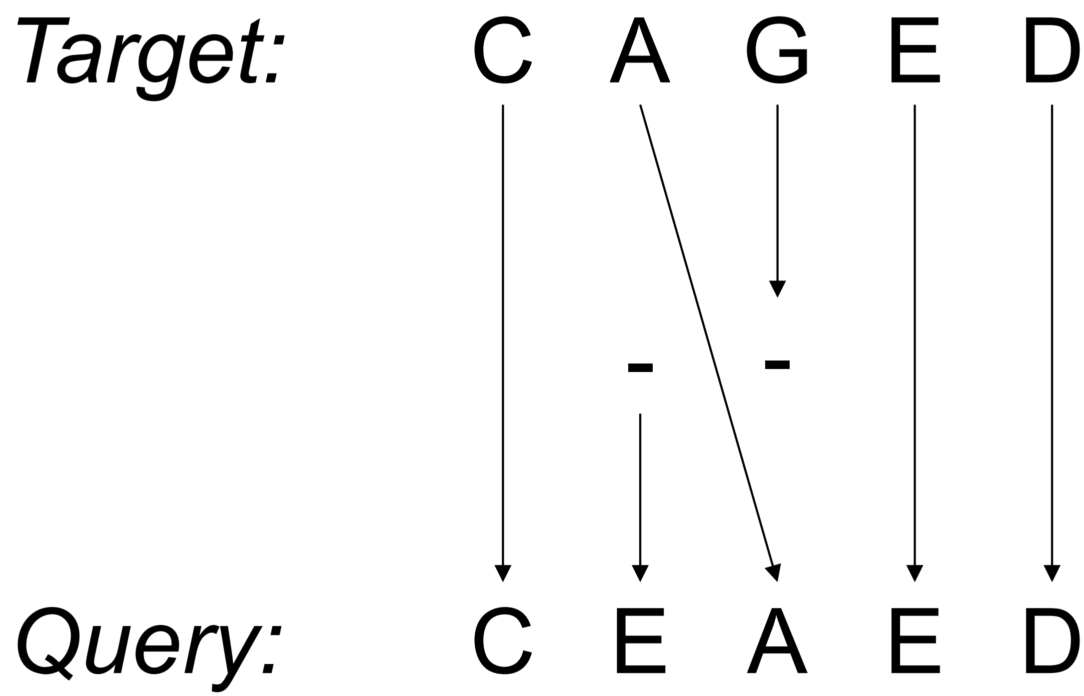
$$D_{KL}(P \| Q) = \sum_i \ln\left(\frac{P(i)}{Q(i)}\right) P(i)$$

$$= \sum_i \left(\ln(P(i)) - \ln(Q(i))\right) P(i)$$

$$= \sum_i P(i) \ln P(i) - \sum_i P(i) \ln Q(i)$$

Edit Distance

- Query = string from finite alphabet
- Target = string from finite alphabet
- Cost of Edits = Distance



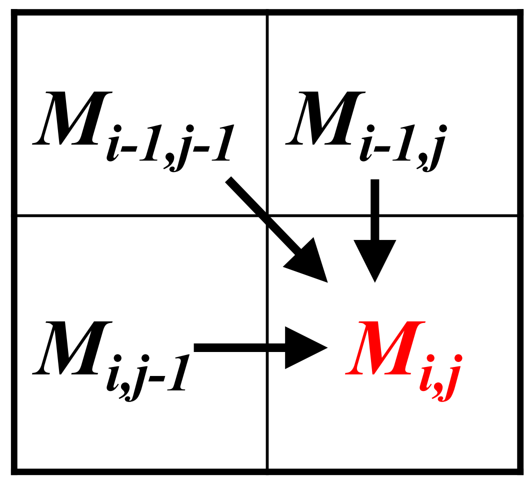
Levenshtein edit distance

$$M_{0,0} = 0$$

3 possible operations

$$M_{i,j} = \min \begin{cases} M_{i-1,j} + 1 & \textbf{Insertion} \\ M_{i,j-1} + 1 & \textbf{Deletion} \\ M_{i-1,j-1} + \mu(s_i, q_j) & \textbf{Substitution} \end{cases}$$

$$\mu(s_i, q_j) = \begin{cases} 0 & \text{if } s_i = q_j \\ 1 & \text{otherwise} \end{cases}$$



Pseudocode of Levenshtein (after Wagner and Fischer)

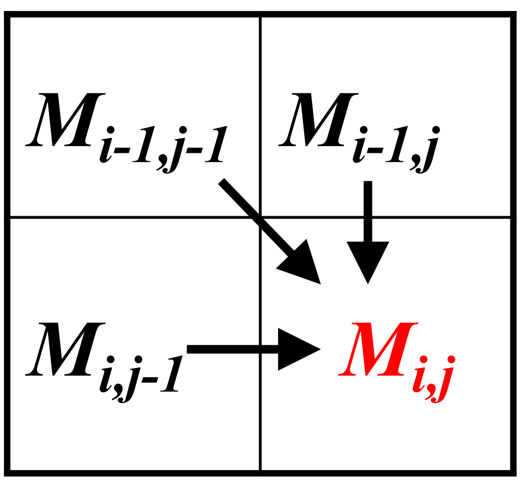
```
return int LevenshteinDistance(char s[1..m], char t[1..n], deletionCost,  insertionCost,
substitutionCost)

// A standard approach is to set deletionCost = insertionCost = substitutionCost = 1

  declare int M[0..m, 0..n]    // M has (m+1) by (n+1) values
  for i from 0 to m
          M[i, 0] := i*deletionCost  // distance of any 1st string to an empty 2nd string
  for j from 0 to n
          M[0, j] := j*insertionCost // distance of any 2nd string to an empty 1st string

  for j from 1 to n
     for i from 1 to m
        if s[i] = t[j] then
           M[i, j] := M[i-1, j-1]   // no operation cost, because they match
        else
           M[i, j] := minimum(M[i-1, j] + deletionCost,
                              M[i, j-1] + insertionCost,
                              M[i-1, j-1] + substitutionCost)
  return M[m,n]
```

Working through an example

$$M_{i,j} = \min \begin{cases} M_{i-1,j} + 1 \\ M_{i,j-1} + 1 \\ M_{i-1,j-1} + \mu(s_i, q_j) \end{cases}$$

$$\mu(s_i, q_j) = \begin{cases} 0 & \text{if } s_i = q_j \\ 1 & \text{otherwise} \end{cases}$$

$$M_{0,0} = 0$$



		F	R	O	G
	0	**1**,-,-	2,-,-	3,-,-	4,-,-
D	-,-,1	2,**1**,2	**2,2**,3	3,3,4	4,4,5
O	-,-,2	3,2,2	3,2,3	3,**2**,4	**3**,4,5
G	-,-,3	4,3,3	4,3,3	4,3,3	4,**2**,4

Edit distance is 2 edits

Working through an example

- The final edit cost is the lowest value calculated for the lower right-hand corner of the matrix.
- Tracing a path from the lower right to the beginning shows 2 minimal-cost alignments, each with 1 substitution and one deletion:

```
FROG    FROG

D–OG    –DOG
```

		F	R	O	G
	0	**1**,-,-	2,-,-	3,-,-	4,-,-
D	-,-,1	2,**1**,2	**2,2**,3	3,3,4	4,4,5
O	-,-,2	3,2,2	3,2,3	3,**2**,4	**3**,4,5
G	-,-,3	4,3,3	4,3,3	4,3,3	4,**2**,4

del F

sub D for F

sub D for R

del R

O = O

G= G

Edit distance is 2 edits

(Somewhat more) General Edit Distance

$$M_{i,j} = \min \begin{cases} M_{i-1,j} + \mu(-, q_j) & \textbf{Insert} \\ M_{i,j-1} + \mu(s_i, -) & \textbf{Delete} \\ M_{i-1,j-1} + \mu(s_i, q_j) & \textbf{Match} \end{cases}$$

$\mu(s_i, q_j) =$ whatever you want.

The distance between s_i and q_j on a keyboard?

The probability of substituting s_i for q_j?

Final notes on edit distance

- Used in many applications
  - Gene sequence matching (google: BLAST)
  - Spell checking
  - Music melody matching
- There are many variants of the algorithms
- The parameter weights strongly affect performance
- You need to pick the algorithm and parameters that make sense for your problem.

Some take-away thoughts

- Many machine learning methods are helped by having a distance measure
- Some methods require metrics
- Not all measures are metrics
- Some common distance measures:

  "P-norms": Euclidean, Manhattan

  "Edit distance": Levenshtein

  KL Divergence

  Mahalanobis