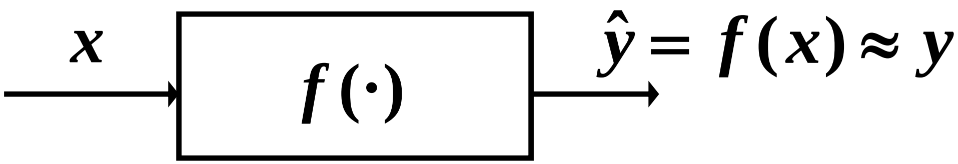
Metric-Based Classifiers

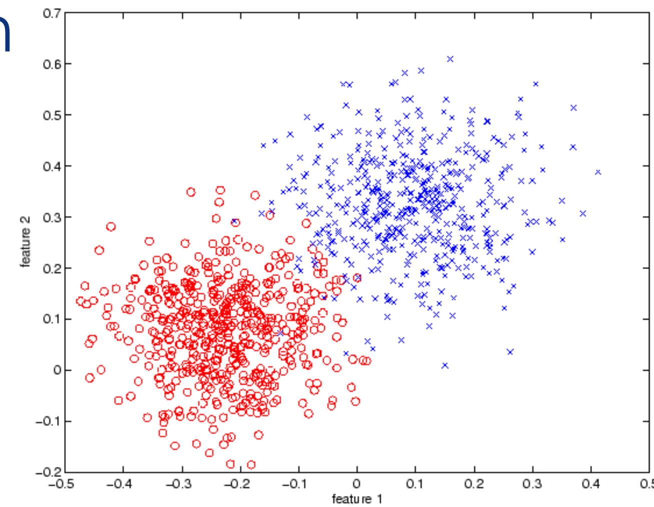
Nuno Vasconcelos

(Ken Kreutz-Delgado)

Statistical Learning from Data

- **Goal:** Given a relationship between a feature vector x and a vector y, and iid data samples (x_i, y_i), find an approximating function $f(x) \approx y$

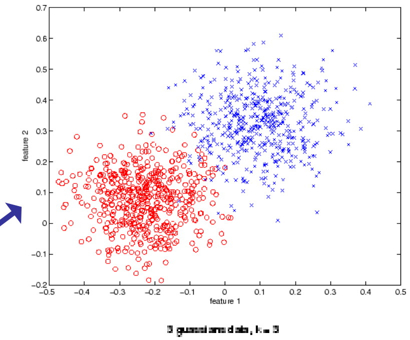


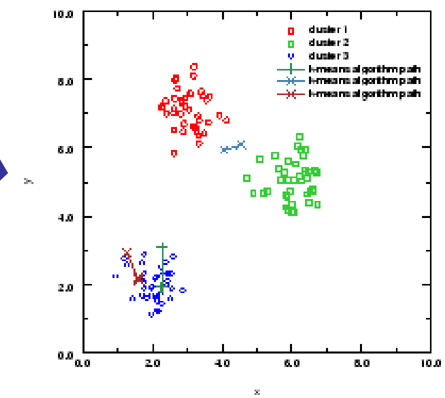


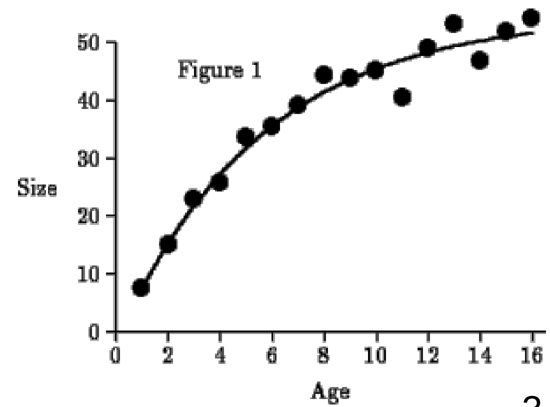
- This is called training or learning.
- Two major types of learning:
  - Unsupervised (aka Clustering) : only *X* is known.
  - Supervised (Classification or Regression): both *X* and target value *Y* are known during training, only *X* is known at test time.

Supervised Learning

- Feature Vector X can be anything, but the type of *Y* dictates the type of supervised learning problem
  - *Y* in {0,1} is referred to as detection
  - *Y* in {0, ..., M-1} is referred to as (M-ary) classification
  - *Y* continuous is referred to as regression
- Theories are quite similar, and algorithms similar most of the time
- We will emphasize classification, but will talk about regression when particularly insightful





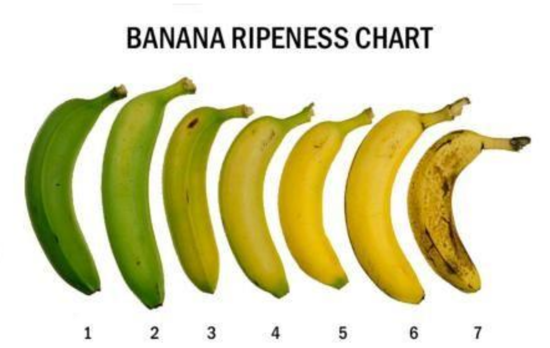


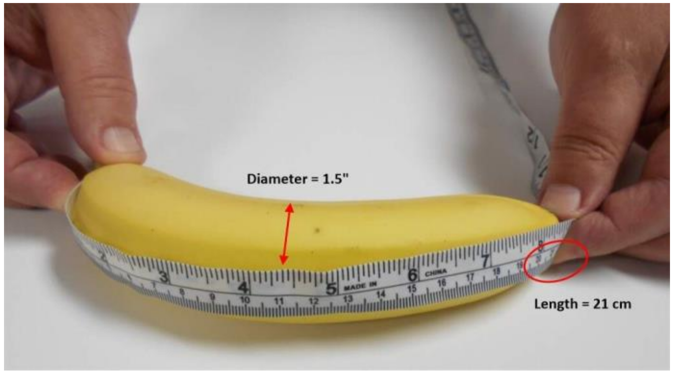
Example

- Classifying fish:
  - fish roll down a conveyer belt
  - camera takes a picture
  - goal: is this a salmon or a sea-bass?
- Q: what is *X*? What features do I use to distinguish between the two fish?
- Feature Selection is somewhat of an art-form. Frequently, the best is to ask “domain experts”.
- E.g. use length and width of scales as features

Bananas

- Any object can be mapped into a vector space.
- E.g. bananas: I can measure
  - Ripeness r
  - Weight w
  - Length l
  - Diameter d
  - Color c

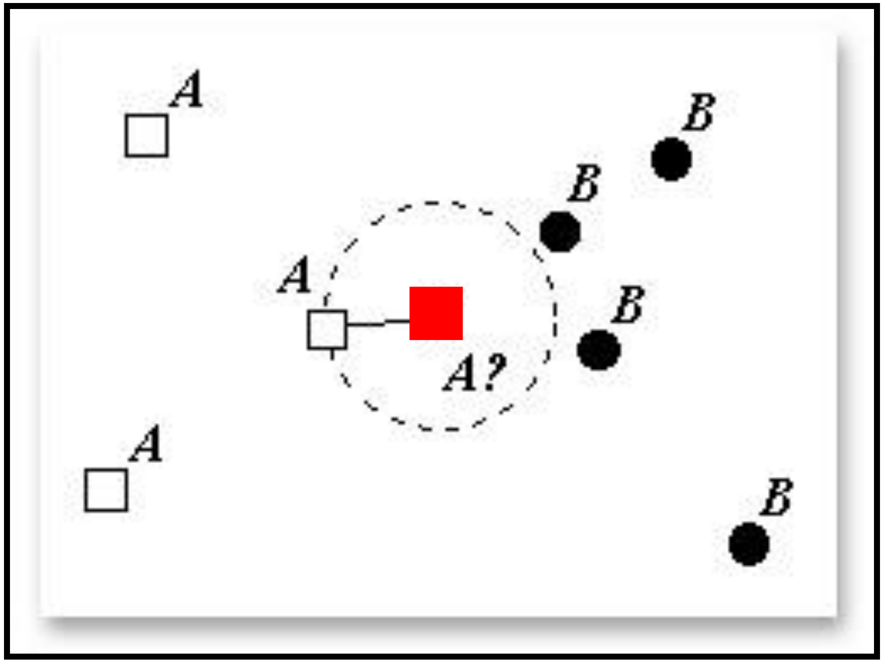




  - and represent a banana by the vector $v = (r, w, l, d, c)^T$
  - The five measurements are called features.

Nearest Neighbor Classifier

- The simplest possible classifier that one could think of:
  - It consists of assigning to a new, unclassified vector the same class label as that of the closest vector in the labeled training set
  - E.g. to classify the unlabeled point "Red":
    - measure Red's distance to all other labeled training points
    - If the closest point to Red is labeled "A = square", assign it to the class A
    - otherwise assign Red to the "B = circle" class



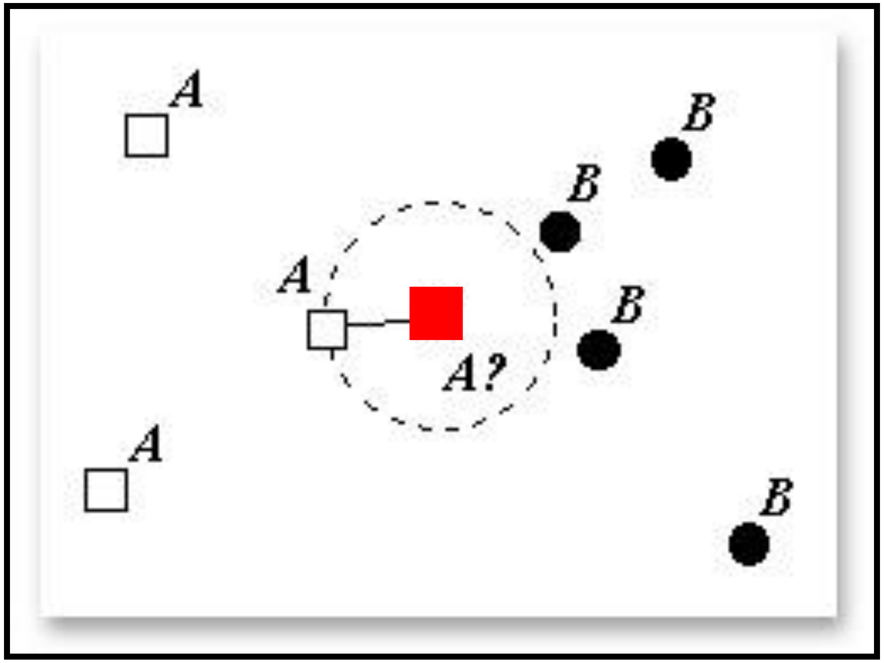
- This works a lot better than what one might expect, particularly if there are a **lot** of labeled training points

Nearest Neighbor Classifier

- To define this classification procedure rigorously, define:
  - a Training Set $\mathcal{D} = \{(x_1,y_1), \ldots, (x_n,y_n)\}$
  - x_i is a vector of observations, y_i is the class label
  - a new vector x to classify
- The Decision Rule is

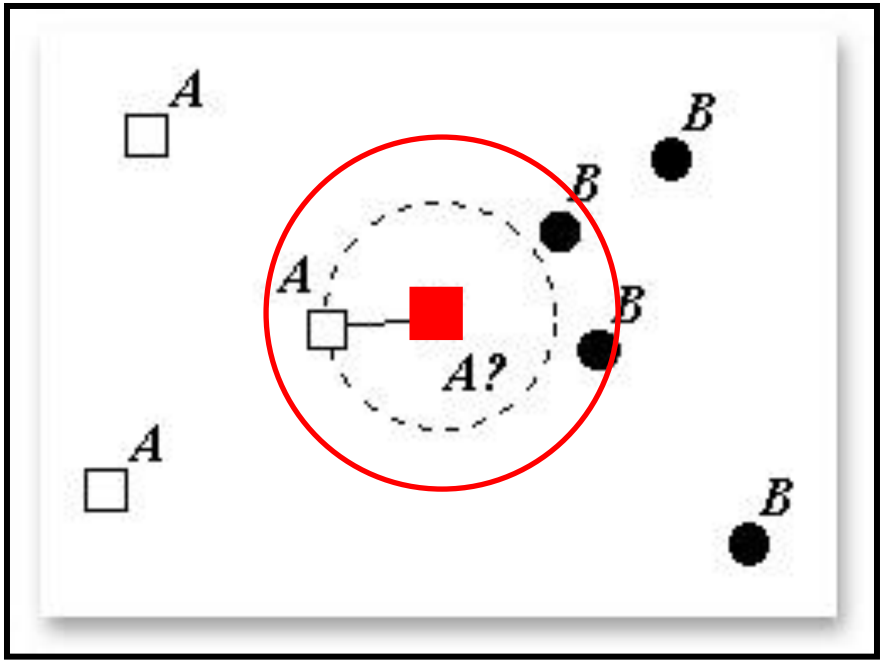
$$
\boxed{\begin{aligned}
&set \quad y = y_{i*} \\
&where \\
&i* = \arg\min_{i \in \{1,\ldots,n\}} d(x, x_i)
\end{aligned}}
$$

  - argmin means: "the *i* that minimizes the distance"



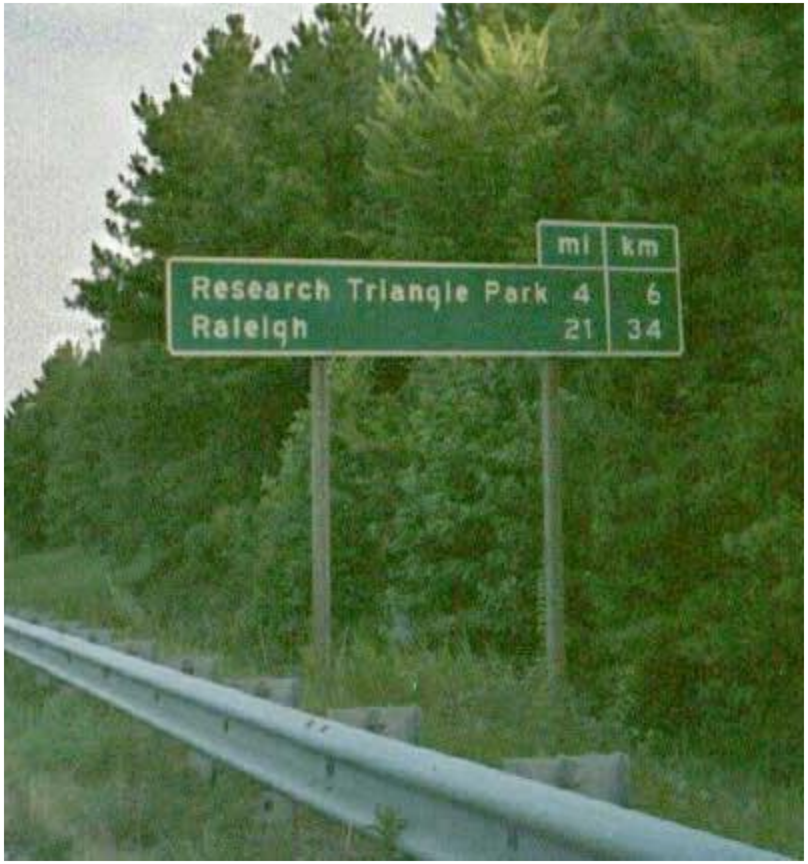
k-Nearest Neighbor (*k*-NN) Classifier

- Instead of the single NN, assigns to the majority vote of the k nearest neighbors
- In this example
  - NN = 1-NN rule says "A"
  - but 3-NN rule says "B"
- Usually best performance for $k > 1$, but there is no universal number
- When k is "too large," the performance degrades (too many neighbors are no longer near)
- *k* should be odd, to prevent ties



Nearest Neighbor Classifier

- We will use *k* = 1 for simplicity
- There are only two components required to design a NN classifier
  - Which Features do we use, i.e. what is x?
  - What Metric d(x,y)?
- Both can have great impact on classification error
- Feature selection is a problem for all classifiers
  - will talk about this later
- A suitable metric can make big difference



Inner Product

- **Definition:** an inner product on a vector space $\mathcal{H}$ is a bilinear form

$$<.,.>: \mathcal{H} \times \mathcal{H} \rightarrow \Re$$
$$(x,x') \rightarrow <x,x'>$$

such that

$$i) \;\; <x,x> \geq 0, \;\; \forall x \in \mathcal{H}$$
$$ii) \; <x,x> = 0 \text{ if and only if } x = 0$$
$$iii) \; <x,y> = <y,x> \text{ for all } x \text{ and } y$$

- Conditions i) and ii) make the inner product a natural measure of similarity
- This is made more precise with introduction of a norm

Metrics

- Any inner product defines (induces) a norm via

$$||x||^2 = <x,x>$$

- The norm has the following properties
  - Positive-Definiteness: $||x|| \geq 0, \ \forall x$, and $||x|| = 0$ iff $x = 0$
  - Homogeneity: $||\lambda x|| = |\lambda| \, ||x||$
  - Triangle Inequality: $||x + y|| \leq ||x|| + ||y||$

- This naturally defines a metric

$$\boxed{d(x,y) = ||x-y||}$$

  which is a measure of the distance between x and y

- Always remember that the metric depends on the choice of the inner product $<x,x>$!

Metrics

- we have seen some **examples**:
  - $\mathbb{R}^d$

Inner Product :

$$\langle x, y \rangle = x^T y = \sum_{i=1}^{d} x_i y_i$$

Euclidean norm:

$$\|x\| = \sqrt{x^T x} = \sqrt{\sum_{i=1}^{d} x_i^2}$$

Euclidean distance:

$$d(x,y) = \|x-y\| = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

-- **Continuous functions**

Inner Product :

$$\langle f(x), g(x) \rangle = \int f(x) g(x) dx$$

norm^2 = 'energy':

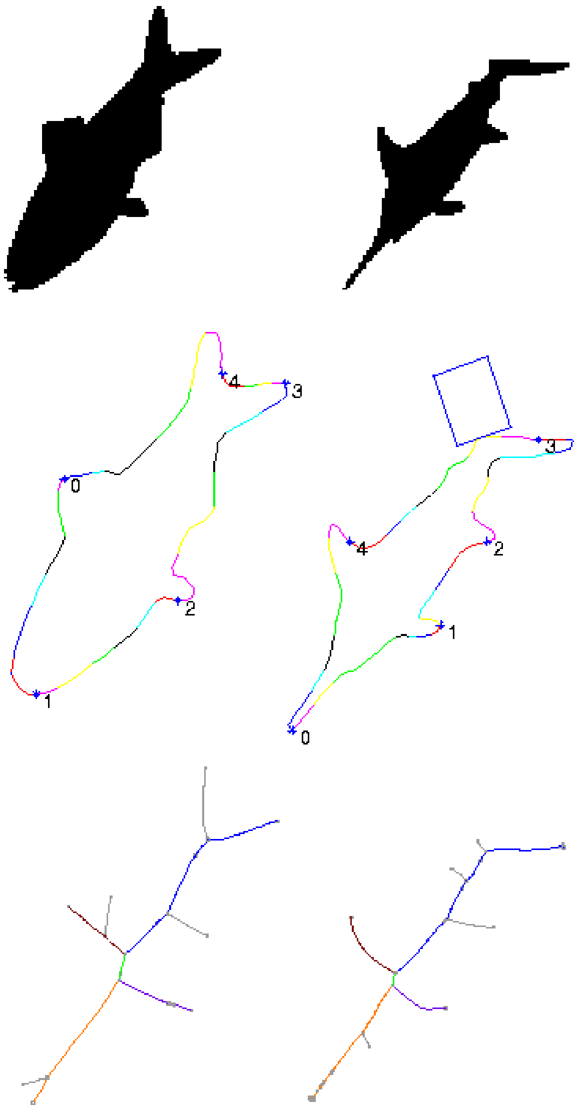
$$\|f(x)\| = \sqrt{\int f^2(x) dx}$$

Distance^2 = 'energy' of difference:

$$d(f,g) = \sqrt{\int [f(x) - g(x)]^2 dx}$$

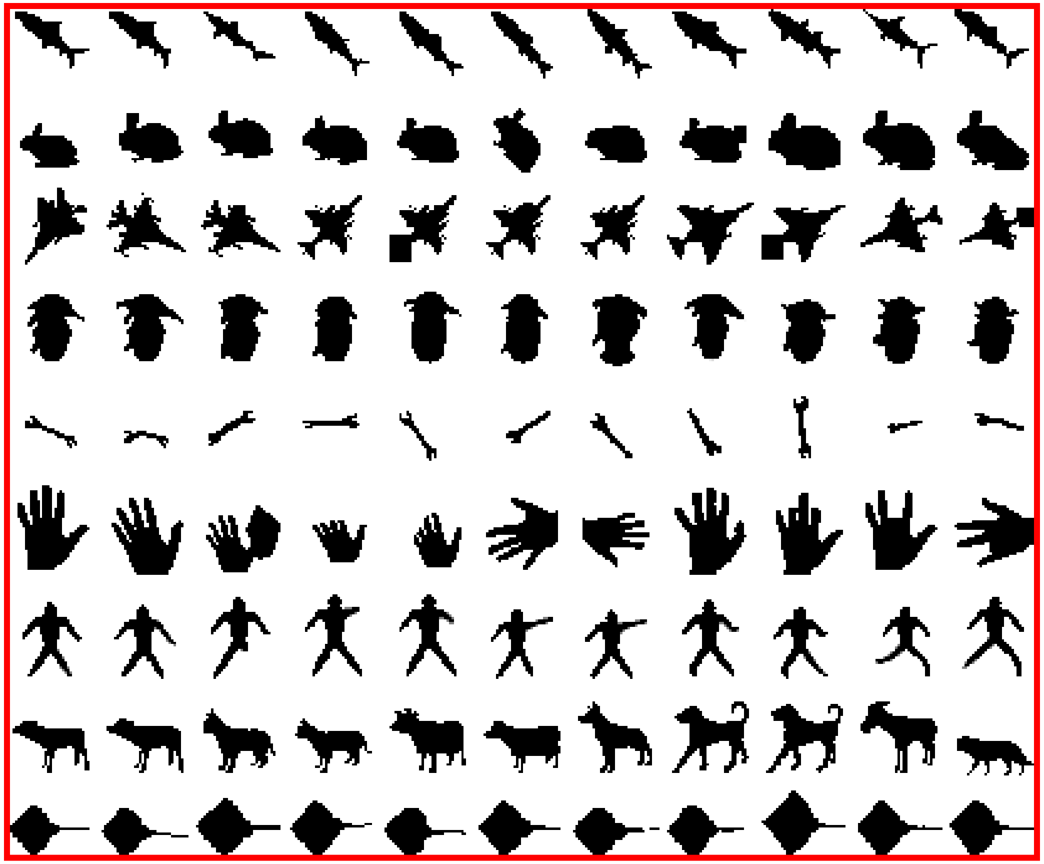
Metrics

- There is an infinity of possible metrics
  - Sometimes it pays off to build one for the specific problem
  - E.g. how do I compare the ***shape*** of two fish?
  - example:
    - find contour
    - compute “skeleton”
    - what is the “energy” that I would need to transform one into the other?
  - This has an “evolutionary motivation” – change requires “effort”.



Nearest Neighbor Classification

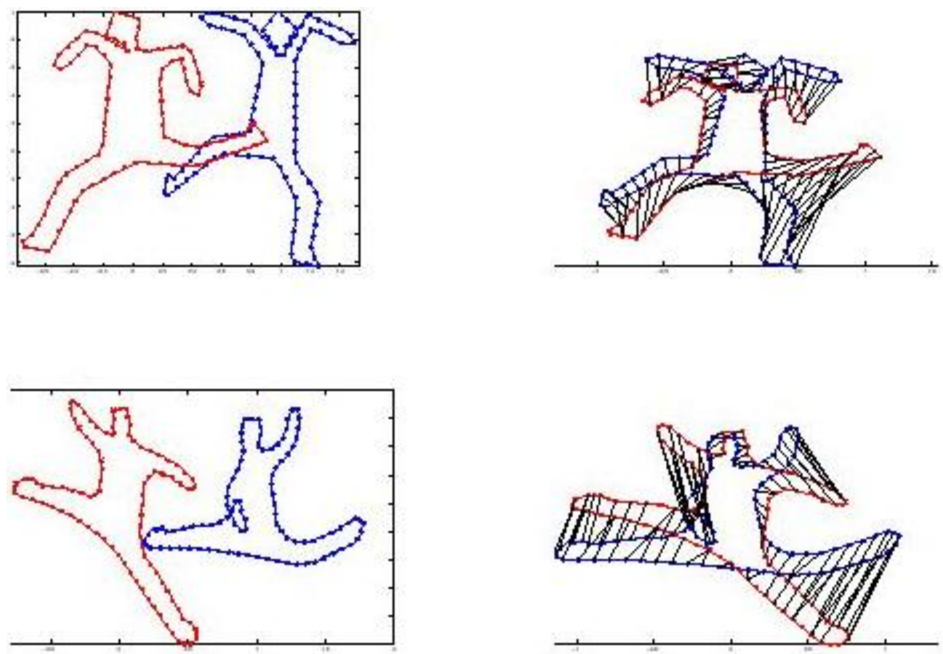
- Finding appropriate features and a metric can be hard, but
  - Once you have the metric you have the classifier
  - The right metric can make a big difference



- Example:
  - Shape retrieval system
  - "What are the new shapes most similar to this class shapes?"
  - Works fairly well

Metrics

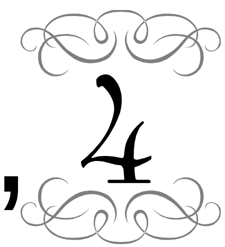
- Many useful metrics are based on this idea of "energy" (inner product induced norm) minimization
  - The less I have to 'work' to transform A into B, the closer they are
  - Sometimes you can ignore transformations that are irrelevant
  - E.g. to understand action, we don't care about relative position or scale



  - We compensate for this and compute "energy" (the value of the square of the norm) between aligned images

‘Energy-Based’ Metrics

- Note that these are just the energy metric in some suitably normalized vector space
- E.g. a metric invariant to rotation

d( , )=0

can be implemented as an ‘energy’ metric after finding the rotation that aligns the images

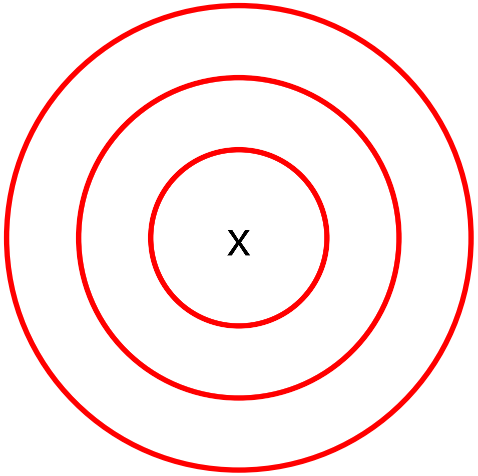
$$d\left(f(\vec{x}), g(\vec{x})\right) = \min_{\theta} d\left(f(R(\theta)\vec{x}), g(\vec{x})\right)$$

$$\text{where} \quad R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \quad d^2 = \text{energy}$$

Euclidean Distance

- So, let's consider the Euclidean distance

$$\boxed{d(x,y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}}$$



- What are equidistant points to x?

$$\boxed{d(x,y) = r \Leftrightarrow \sum_{i=1}^{d} (x_i - y_i)^2 = r^2}$$

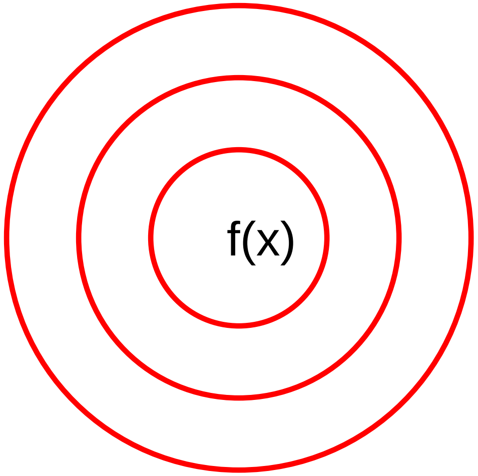
  - e.g. $(x_1 - y_1)^2 + (x_2 - y_2)^2 = r^2$

- The equidistant points to x (aka "level sets") are located on spheres around x
  - Set of points y such that d(x,y) = r is the sphere of radius r centered on x

Euclidean Distance

- The same holds in the continuous case

$$d(f,g) = \sqrt{\int [f(x) - g(x)]^2 dx}$$



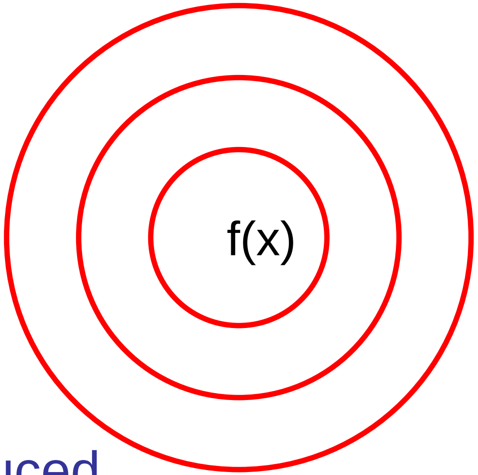
where

$$d(f,g) = r \Leftrightarrow \int [f(x) - g(x)]^2 dx = r^2$$

- This is still the "sphere" of radius r centered on f(x) but now, we are in an infinite dimensional space, so it is impossible to visualize
  - If you think about it, we already couldn't visualize the case of the Euclidean distance for d = 4

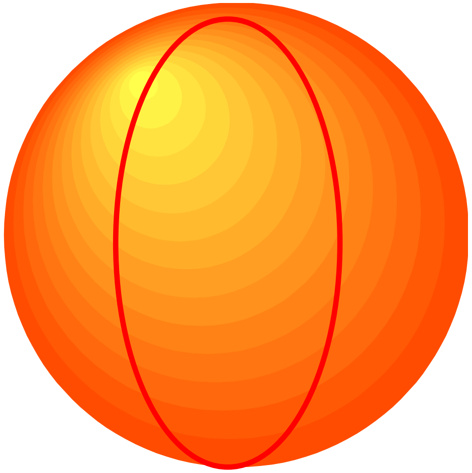
Euclidean Distance

- For intuition, we'll continue with the Euclidean distance
  - Seems like a natural distance
  - We know it is a metric
  - Why would we need something else?
- Remember that underlying the metric there is
  - A vector space
  - An associate inner product, if the metric is induced
- The Euclidean distance works for "flat" spaces
  - E.g. hyperplanes (e.g. the 3D world)
  - "The shortest path between two points is a line"
- But there are many problems that involve non-flat spaces

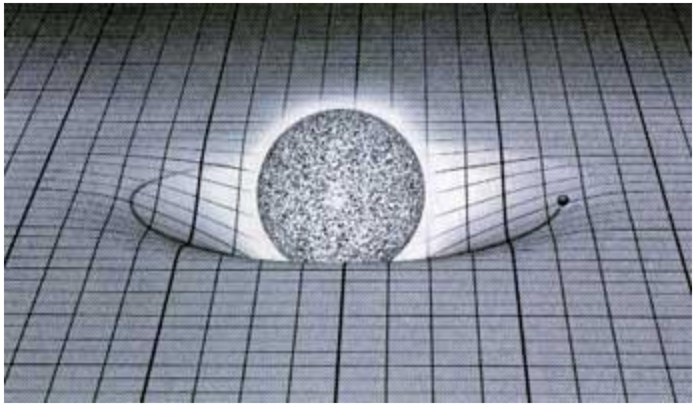


Non-flat Spaces

- What if your space is a sphere?
  - Clearly, the shortest distance is not a line
  - The problem is the assumption that the space has no curvature



- To deal with this you have to use a different geometry
  - Riemannian instead of Euclidean geometry
  - Einstein realized this, and a lot of his relativity work was the development of this different geometry
  - Much of relativity followed easily once he got the geometry right



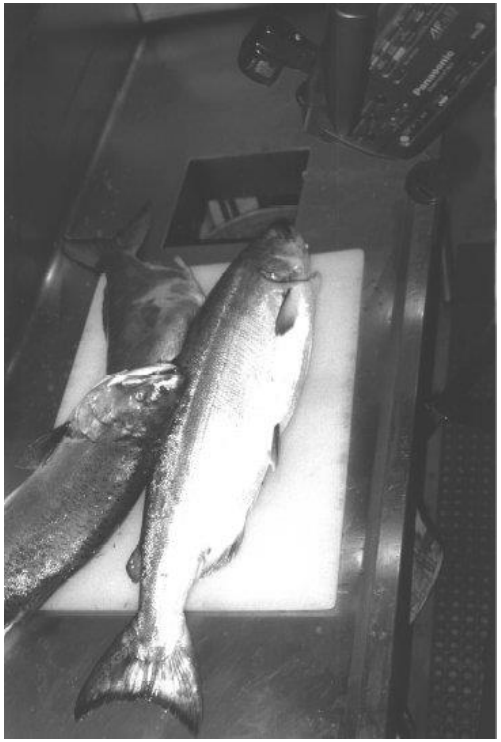
- We will certainly not go into this in any great depth

Inner Products

- So, we will (mostly) work in flat spaces
- What about the inner product? What are potential problems?
- Fish example:
  - Features are L = fish length, W = scale width
  - Let's say I measure L in meters and W in millimeters
    - Typical L: 0.70m for salmon, 0.40m for sea-bass
    - Typical W: 35mm for salmon, 40mm for sea-bass
  - I have three fish
    - $F_1 = (.7,35)$ $F_2 = (.4, 40)$ $F_3 = (.75, 37.8)$
    - F_1 clearly salmon, F_2 clearly sea-bass, F_3 looks like salmon
    - yet

$$d(F_1,F_3) = 2.8 \quad > \quad d(F_2,F_3) = 2.23$$

  - There seems to be something wrong here!



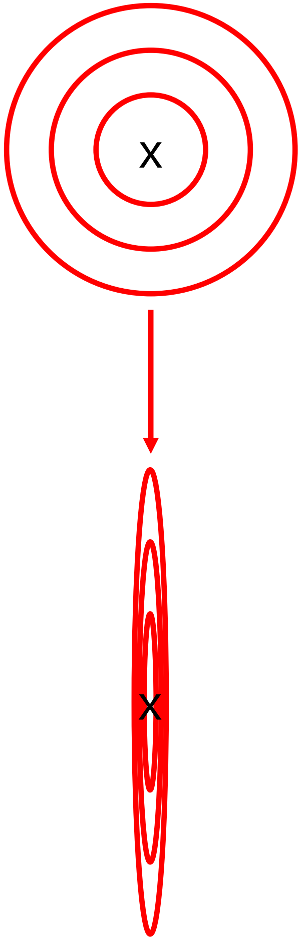
Inner Products

- Suppose the scale width is now also measured in meters:
  - I have three fish
    - $F_1 = (.7,.035)$ $F_2 = (.4, .040)$ $F_3 = (.75, .0378)$
    - and now

      $d(F_1,F_3) = .05 \quad << \quad d(F_2,F_3) = 0.35$

    which seems to be right
- The problem is that the Euclidean distance depends on the units (or scaling) of each axis
  - e.g. if I multiply the second coordinate by 1,000 (say, by changing units from meters to millimeters)

$$d(x,y) = \sqrt{[x_1 - y_1]^2 + [1,000(x_2 - y_2)]^2}$$



its influence on the relative distance increases one thousand-fold!

- Often the “right” units are not clear (e.g. car speed vs weight)

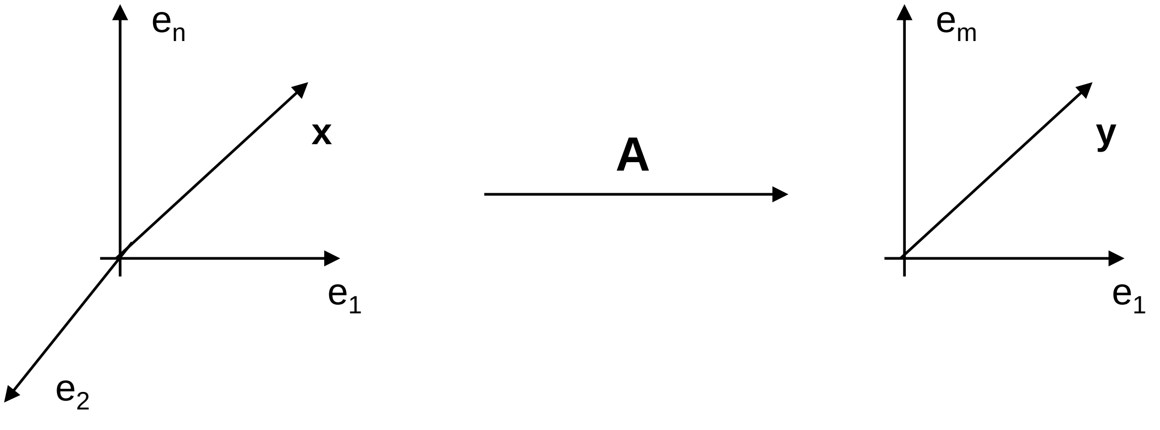
Inner Products

- Perhaps one can transform the problem to a better posed form?

- Remember, an $m \times n$ matrix is an operator that maps a vector from R^n to a vector in R^m

- E.g. the equation $y = Ax$ sends x in R^n to y in R^m

$$\boxed{\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} a_{11} & & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}}$$



Inner Products

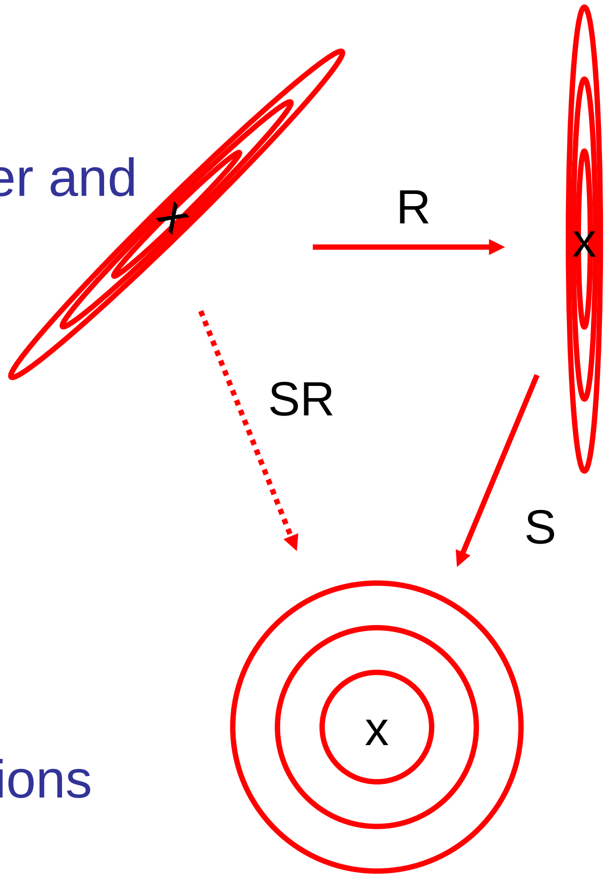
- Suppose I apply a transformation to the feature space

$$x' = Ax$$

- Examples:
  - We already saw that $A = R$, for R proper and orthogonal, is equivalent to a rotation
  - Another important case is scaling, A = S with S diagonal:

$$\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \lambda_n \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \lambda_1 x_1 \\ \vdots \\ \lambda_n x_n \end{bmatrix}$$

  - We can combine two such transformations by taking $A = SR$



(Weighted) Inner Products

- Thus, in general one can rotate *and* scale by applying some matrix $A = SR$, *to form transformed vectors* $\mathbf{x}' = \mathbf{A}\mathbf{x}$

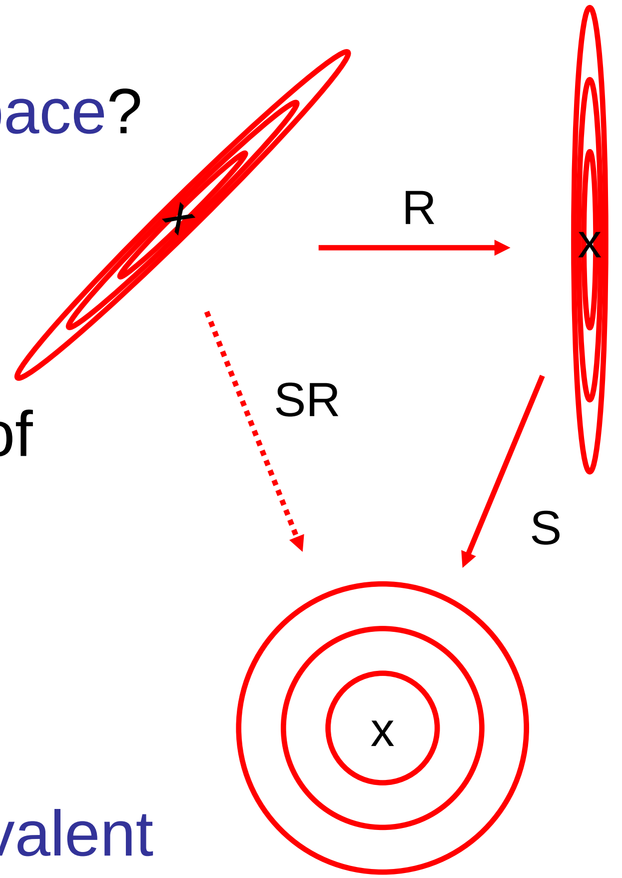
- What is the inner product in the new space?

$$(\mathbf{x}')^T \mathbf{y}' = (\mathbf{A}\mathbf{x})^T \mathbf{A}\mathbf{y} = \mathbf{x}^T \underbrace{\mathbf{A}^T \mathbf{A}}_{\mathbf{M}} \mathbf{y}$$

- The inner product in the new space is of weighted form in the old space

$$\boxed{\langle x', y' \rangle = x^T M y}$$

- Using a weighted inner product is equivalent to working in the transformed space



(Weighted) Inner Products

- Can I use any weighting matrix M ? – NO!
- **Recall:** an inner product is a bilinear form such that

  i) $\langle x,x\rangle \geq 0, \ \forall x \in \mathcal{H}$

  ii) $\langle x,x\rangle = 0$ *if and only if* $x = 0$

  iii) $\langle x,y\rangle = \langle y,x\rangle$ *for all* x *and* y

- From iii), M must be Symmetric since

$$\langle x, y\rangle = x^T M y = (y^T M^T x)^T = y^T M^T x$$
$$\langle y, x\rangle = y^T M x$$
$$\langle x, y\rangle = \langle y, x\rangle, \text{if and only if } M = M^T$$

- from i) and ii), M must be Positive Definite

$$\boxed{\langle \mathbf{x}, \mathbf{x}\rangle = \mathbf{x}^T \mathbf{M} \mathbf{x} > \mathbf{0}, \quad \forall \mathbf{x} \neq \mathbf{0}}$$

Positive Definite Matrices

- **Fact:** Each of the following is a necessary and sufficient condition for a real symmetric matrix A to be positive definite:
  - i) $x^T A x > 0, \quad \forall x \neq 0$
  - ii) All eigenvalues, λ_i, of A are real and satisfy $\lambda_i > 0$
  - iii) All upper-left submatrices A_k have strictly positive determinant
  - iv) There is a matrix R with independent columns such that $A = R^T R$

- Definition of upper left submatrices:

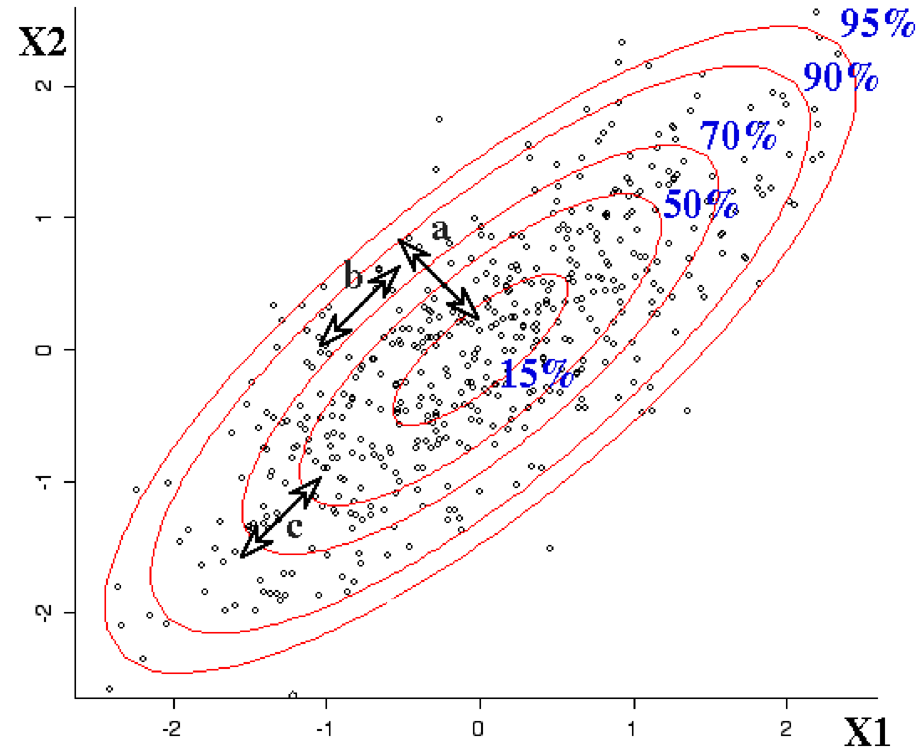
$$A_1 = a_{1,1} \qquad A_2 = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} \qquad A_3 = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \qquad \cdots$$

- Note: from property iv), using a positive definite A to weight an inner product is equal to working in a transformed space.

$$\langle x', y' \rangle = x^T A y = x^T R^T R y = \langle Rx, Ry \rangle$$

Metrics

- What is a good weighting matrix M ?
  - Let the data tell us!
  - Use the inverse of the covariance matrix M = Σ^{-1}



$$\Sigma = E\left[(x-\mu)(x-\mu)^T\right]$$
$$\mu = E[x]$$

- Mahalanobis Distance: $$\boxed{d(x,y) = \sqrt{(x-y)^T \Sigma^{-1} (x-y)}}$$

- This distance is adapted to the covariance (“scatter”) of the data and thereby provides a “natural” rotation and scaling for the data

The Multivariate Gaussian

- In fact, for Gaussian data, the Mahalanobis distance tells us all we could statistically know about the data
  - The pdf for a d-dimensional Gaussian of mean μ and covariance Σ is

$$\boxed{P_X(x) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left\{ -\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu) \right\}}$$

  - Note that this can be written as

$$\boxed{P_X(x) = \frac{1}{K} \exp\left\{ -\frac{1}{2} d^2(x,\mu) \right\}}$$

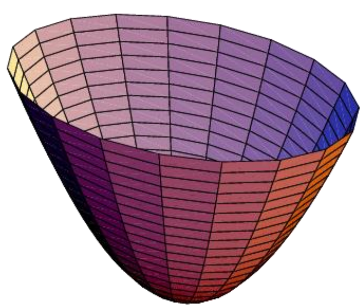
  - I.e. a Gaussian is just the exponential of the negative of the square of the Mahalanobis distance
  - The constant *K* is needed only to ensure the density integrates to 1

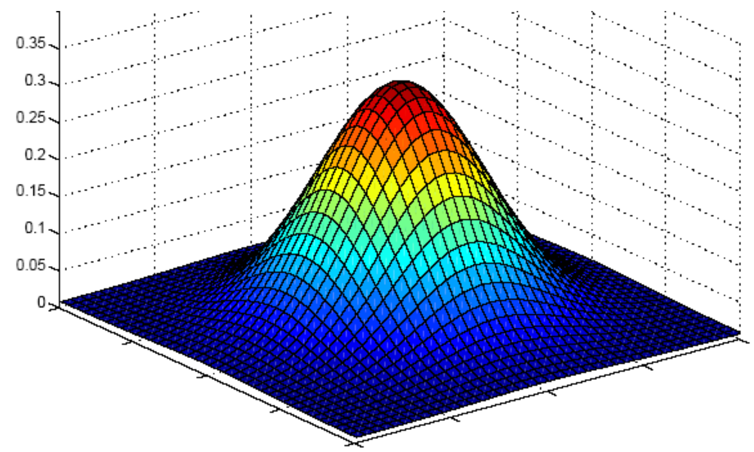
The Multivariate Gaussian

- Using Mahalanobis = assuming Gaussian data
- Mahalanobis distance: Gaussian pdf:

$$\boxed{d^2(x,\mu) = (x-\mu)^T \Sigma^{-1} (x-\mu)}$$

$$\boxed{P_X(x) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left\{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right\}}$$





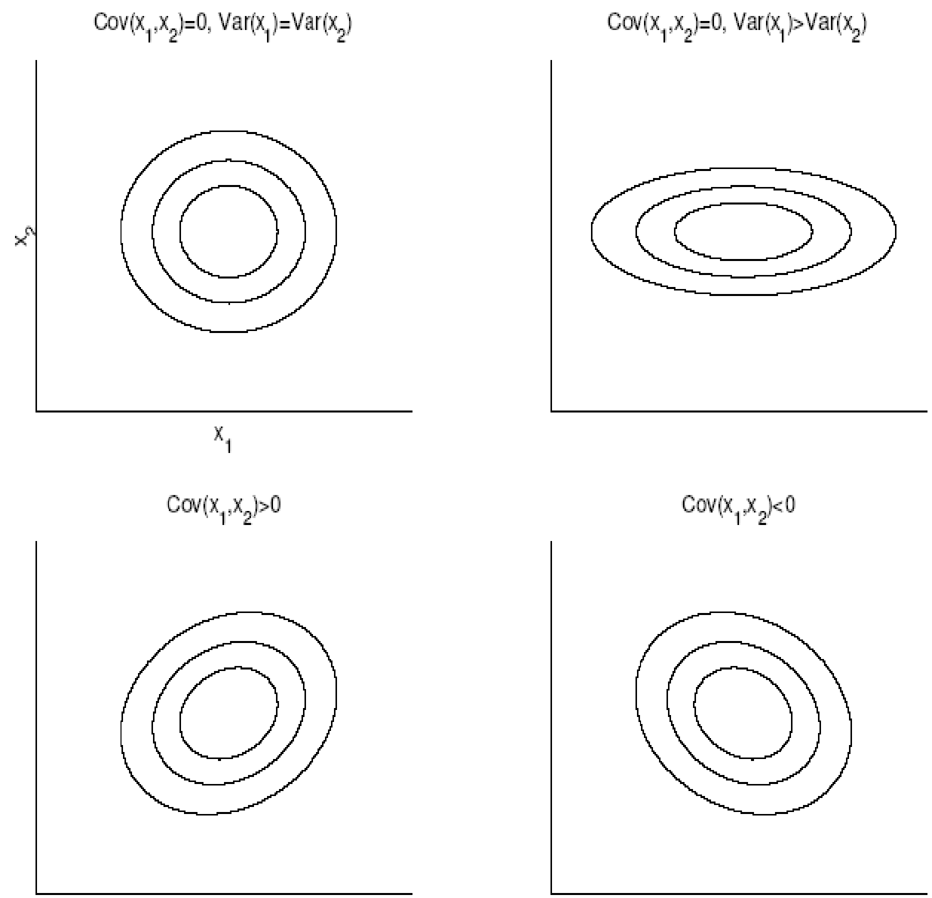
– Points of high probability are those of small Mahalanobis distance to the center (mean) of a Gaussian density
– This can be interpreted as the right norm for a certain type of space

The Multivariate Gaussian

- Defined by two parameters
  - Mean just shifts center
  - Covariance controls shape
  - in 2D, $X = (X_1, X_2)^T$

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{bmatrix}$$

  - σ_i^2 is variance of X_i
  - $\sigma_{12} = \mathrm{cov}(X_1, X_2)$ controls how dependent X_1 and X_2 are
  - Note that, when $\sigma_{12} = 0$:



$$P_{X_1,X_2}(x_1, x_2) = P_{X_1}(x_1) P_{X_2}(x_2) \Leftrightarrow$$ X_i are independent

The Multivariate Gaussian

- The best way to understand the role of the different parameters is by experimenting
- On MATLAB make a plot of the Mahalonabis distance

$$\boxed{d^2(x,\mu) = (x-\mu)^T \Sigma^{-1} (x-\mu)}$$

- Start with $\mu = 0$ and $\Sigma = I$
- The consider different values of μ and see how the plot changes
- Finally consider different covariances

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{bmatrix}$$

- Note that σ_1^2 and σ_2^2 “stretch” the covariance, while σ_{12} changes the orientation.

Any questions?