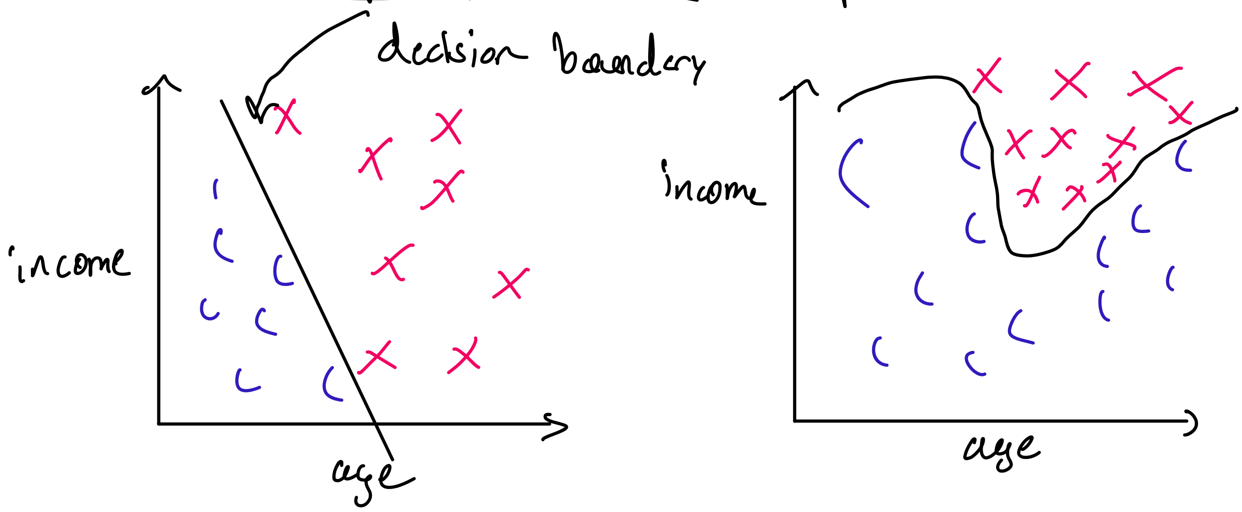
Classifiers

- Given a sample of n observations w/ d features/predictors each.
- Some observations belong to class C; others do not

example: bank loans = observations
features = income & age
Classes = [defaulted, not defaulted]

goal: predict whether someone will fault or not based on income & age

- represent each object in d dimensional space
↳ sample point / feature vector / independent variables



decision boundary: separates classes

Overfitting: when decision boundaries become "snake-like"
↳ real test: will boundary classify test data well?

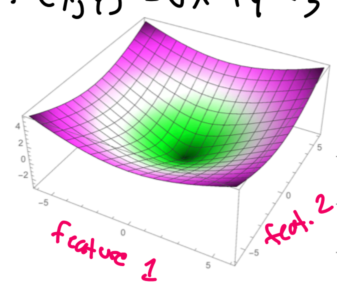
Some classifiers compute a decision function: function $f(x)$ that maps sample point $\vec{x}$ to scalar s.t. $f(x)$ is positive if sample is in class C and vice versa

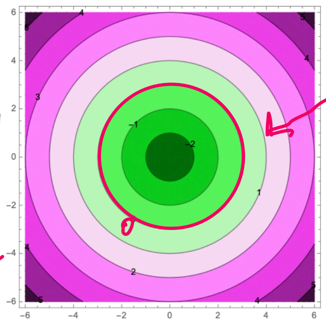
aka "predictor functions" or "discriminant"

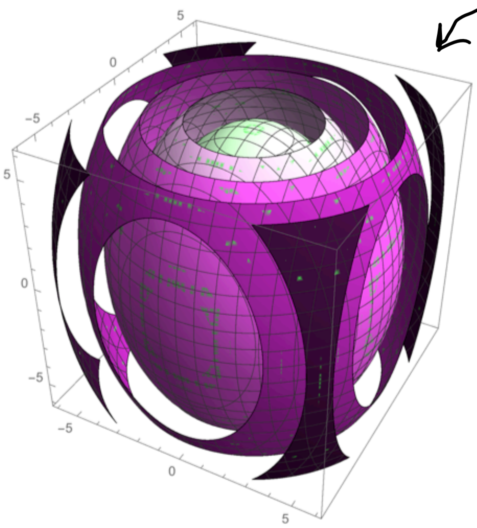
For such classifiers, decision boundary is defined by $\boxed{\{x \in \mathbb{R}^d, f(x) = 0\}}$ infinite set of points

$\{x : f(x) = 0\}$ is also called an isosurface of $f(\cdot)$ for isovalue 0
* f has other isosurfaces & isovalues e.g. $\{x : f(x) = 1\}$

$f(x,y) = \sqrt{x^2+y^2-3}$







2D isocontours in 3D space

linear Classifier: decision boundary is linear (line/plane/hyperplane...)
↳ most use linear decision function
SVM's don't have linear

Math Review

Vectors: $x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = [\, x_1 \;\; x_2 \;\; x_3 \;\; x_4 \;\; x_5 \,]^T$

↳ point in 5D space

Conventions: uppercase roman: Matrix, random Variable, Set
lowercase roman: Vector
greek: Scalar
other scalars: n = # sample points, d = # features/dimension of sample points
i, j, k = integer indices

Inner product: $x \cdot y = x_1 y_1 + \dots + x_d y_d$
also written in matrix notation: $x^T y$

clearly, $f(x) = w \cdot x + \alpha$ is a linear function (linear in x)

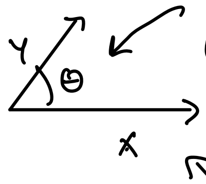
Euclidean Norm: $\|x\| = \sqrt{x^T x} = \sqrt{x_1^2 + x_2^2 + \dots + x_d^2}$

$\|x\|$ is length of $\vec{x}$ (euclidean length)

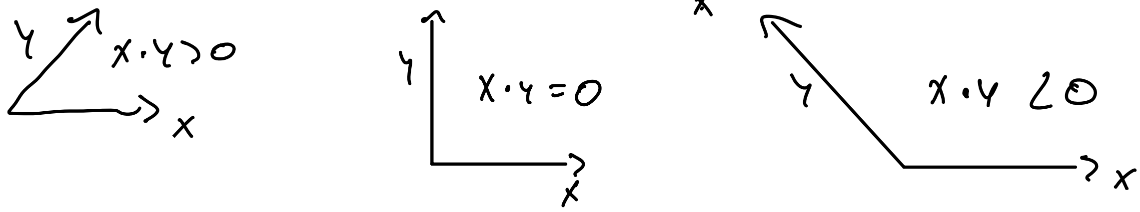
Given $\vec{x}$, $\frac{\vec{x}}{\|\vec{x}\|}$ is a unit vector (length $= 1$)

normalization

Use dot products to compute angles:



$\cos\theta = \frac{x \cdot y}{\|x\| \cdot \|y\|} = \frac{x}{\|x\|} \cdot \frac{y}{\|y\|}$

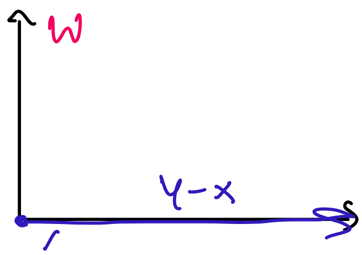


given $f(x) = w \cdot x + \alpha$, decision boundary is

$$H = \{x : wx = -\alpha\}$$

Set H is a hyperplane (line in 2D, plane in 3D)

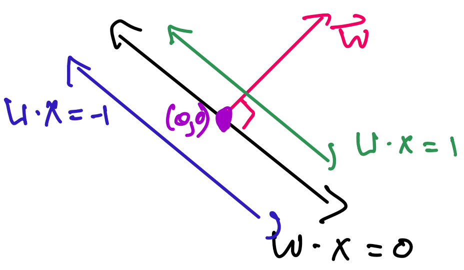
- dimension $d-1$
- flat
- infinite
- cuts d-space into 2



theorem: Let x, y be 2 points on H. Then $w \cdot (y - x) = 0$

Proof: $w(y-x) = -\alpha + \alpha = 0 \Rightarrow w \perp x \ \forall x \in H$

$\rightarrow$ w is called normal vector of H (perpendicular)



If $\vec{w}$ is a unit vector, then $w \cdot x + \alpha$ is the signed distance from point x to hyperplane H i.e. positive on w's side of H
negative on opposite side of w

Moreover, distance from H to origin is α

$\Rightarrow \alpha = 0$ iff H passes through origin

The coefficients in w, plus α, are the weights (or parameters) of the classifier. or regression coefficients

Input data is linearly seperable if there exists a hyperplane that seperates all samples by their class
↳ If not seperable, accuracy won't be 100%.

e.g. Simple Classifier: Centroid Method

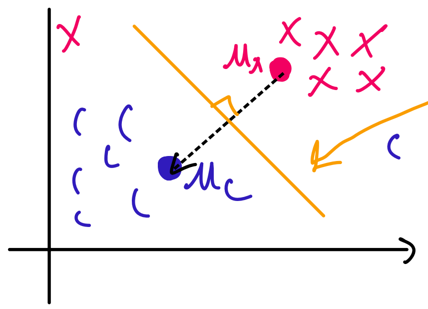
1) - compute mean μ_c of all points in class C
- compute μ_x of all points not in class C

midpoint of mean vectors

2) decision function $f(x) = \underbrace{(\mu_c - \mu_x)}_{\text{normal vector}} \cdot \vec{x} - (\mu_c - \mu_x) \cdot \frac{\mu_c + \mu_x}{2}$

→ decision boundary passes through midpoint between μ_c & μ_x

decision boundary is H-plane that bisects line segment w/ end points μ_c, μ_x → - when $\vec{x} = \frac{\mu_c + \mu_x}{2}$, $f(x) = 0$ → decision boundary passes midpoint



Perceptron algorithm (Frank Rosenblatt, 1957)
- slow
- Correct for linearly seperable points
- Uses a numerical optimalization algorithm, namely GRADIENT DESCENT

Consider n training points $X_1, X_2 \dots X_n$

each row is a datapoint

each training point X_i has a label y_i

$$\text{label} = \begin{cases} 1 & X_i \text{ in class } C \\ -1 & \text{e/w} \end{cases}$$

For simplicity, consider only boundaries that pass through origin

goal: find weights $\vec{w}$ s.t. $\vec{X}_i \cdot \vec{w} \geq 0$ if $y_i = 1$

$\vec{X}_i \cdot \vec{w} < 0$ if $y_i = -1$

equivalently: $y_i X_i \cdot \vec{w} \geq 0$ ← constraint

Idea: define risk function R that is positive if some constraints are violated, 0 otherwise

Then, use optimization algo that minimizes R

Loss Function:

scalar, same sign

$$L(z, y_i) = \begin{cases} 0 & \text{if } y_i z \geq 0 \\ -y_i z & \text{otherwise} \end{cases}$$

classifier's prediction

Risk Function: (Cost function) (obj. function)

$$R(w) = \frac{1}{n} \sum_{i=1}^{n} L(X_i \cdot w, y_i)$$

$$= \frac{1}{n} \sum_{i \in V} -y_i(X_i \cdot w)$$, V's set of misclassified training points

$$R(X) = \begin{cases} 0 & \text{all training points classified correctly} \\ \text{positive} & \text{o/w} \end{cases}$$

goal: Solve this optimization problem

Find w that minimizes $R(w)$

$R(w)$