UCB - CS189
Introduction to Machine Learning
Fall 2015
Lecture 12: Gaussian classifier

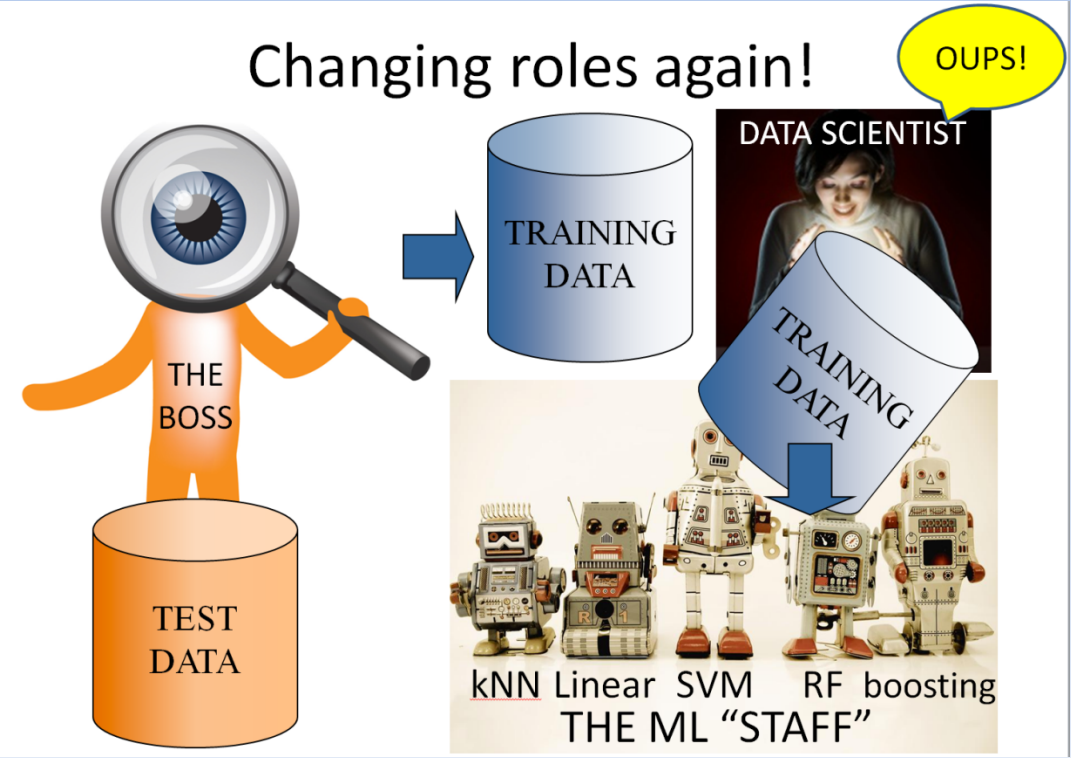
Isabelle Guyon

ChaLearn

Come to my office hours…

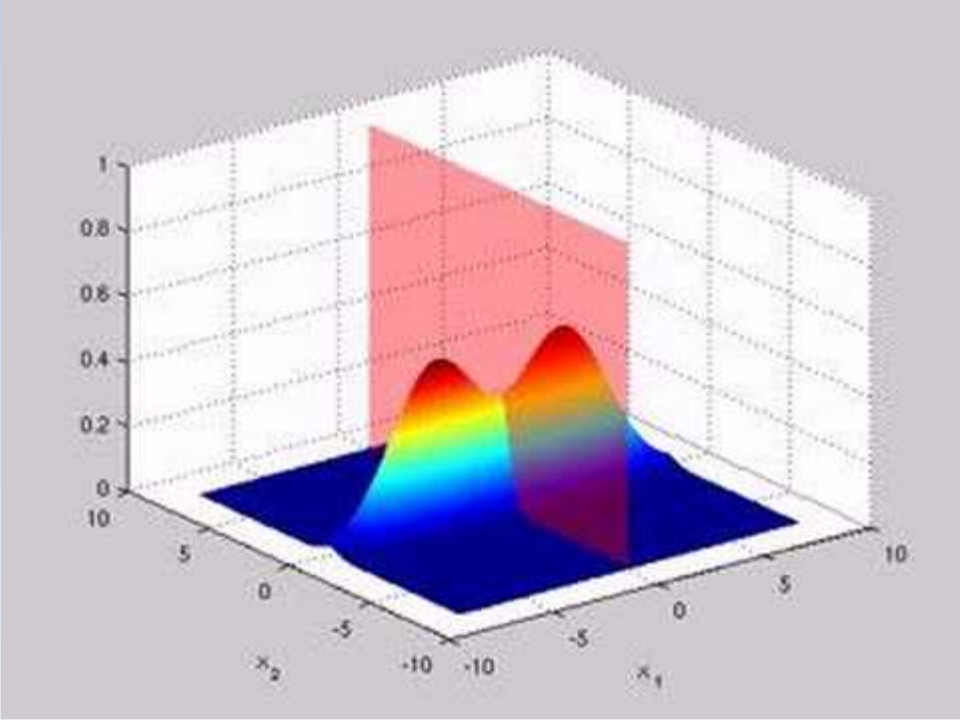
Wed 2:30-4:30 Soda 329

Last time: embedded methods



Come to my office hours…
Wed 2:30-4:30 Soda 329

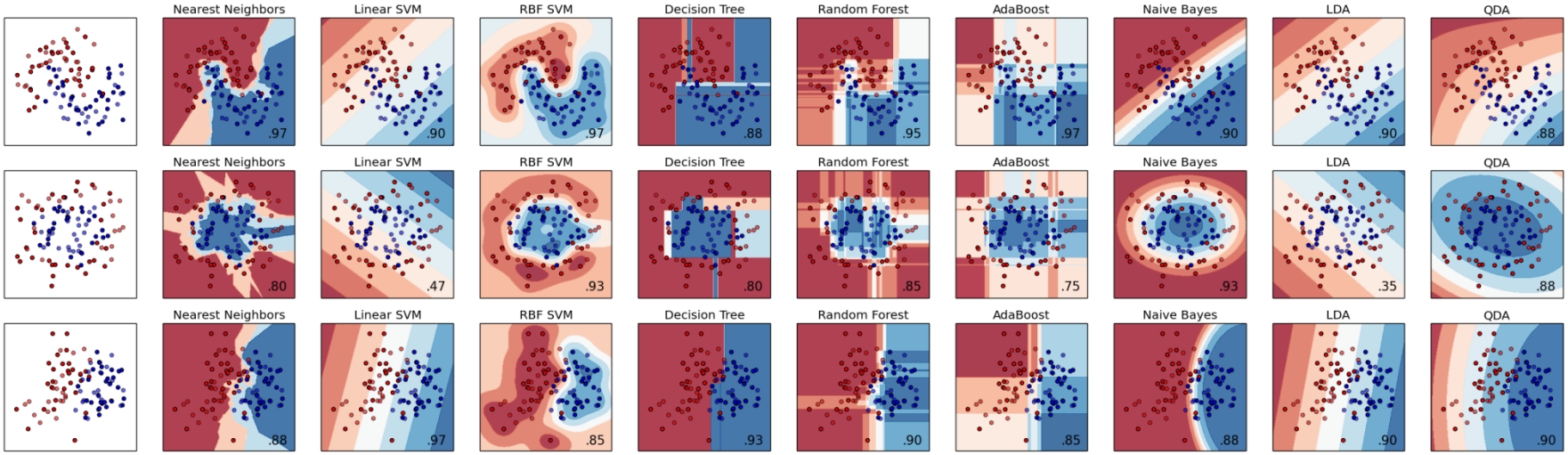
Today: Gaussian classifier aka centroid method aka naïve Bayes aka Hebb's rule revised

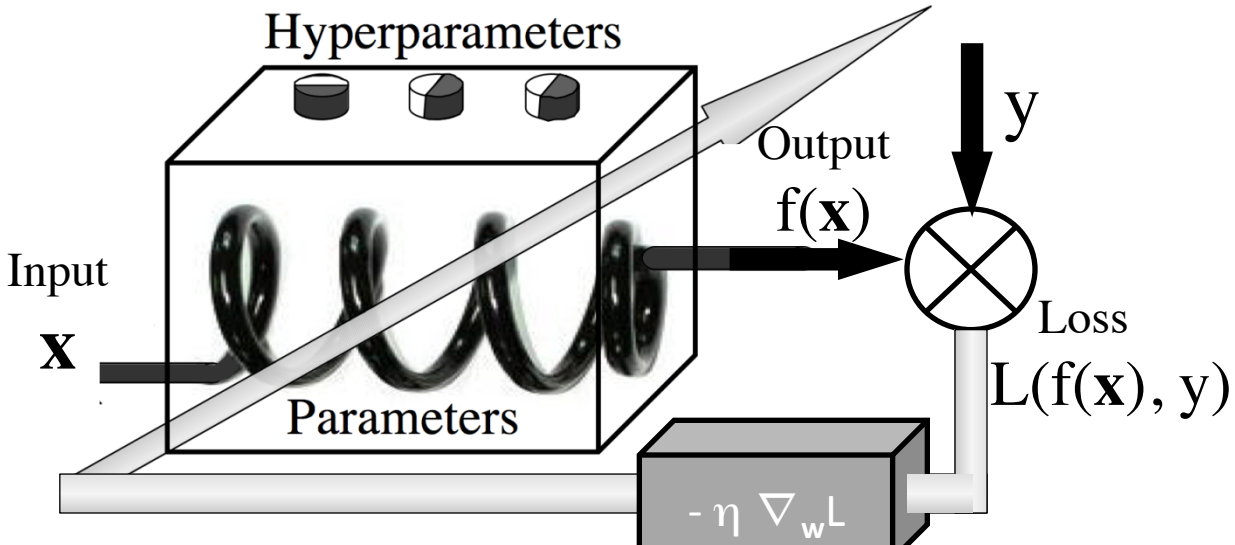


Review

State of the art ML toolkits

Source: Scikit Learn http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html





Review

Perceptrons and kernel methods

$$f(\mathbf{x}) = \left(\sum_k \alpha_k \left[\Phi(\mathbf{x}^k) \right) \right] \bullet \Phi(\mathbf{x})$$

$\mathbf{w}$ $\quad$ $k(\mathbf{x}^k, \mathbf{x}; \theta)$

Perceptrons

- α_k fitted to R_{train}
- $k(\mathbf{x}^k, \mathbf{x}) = \mathbf{x}^k \bullet \mathbf{x}$
- No θ
- $f(\mathbf{x}) = \mathbf{w} \bullet \mathbf{x}$

Hebb's method

- $\alpha_{k=} y_k$
- $k(\mathbf{x}^k, \mathbf{x}) = \mathbf{x}^k \bullet \mathbf{x}$
- No θ
- $f(\mathbf{x}) = \mathbf{w} \bullet \mathbf{x}$

Parzen windows

- $\alpha_{k=} y_k$
- Any $k(\mathbf{x}^k, \mathbf{x}; \theta)$
- θ fitted to R_{CV}
- $f(\mathbf{x})=\Sigma_k \alpha_k k(\mathbf{x}^k, \mathbf{x}; \theta)$

Kernel methods

- α_k fitted to R_{train}
- Any $k(\mathbf{x}^k, \mathbf{x}; \theta)$
- θ fitted to R_{CV}
- $f(\mathbf{x})=\Sigma_k \alpha_k k(\mathbf{x}^k, \mathbf{x}; \theta)$

Review

Training

- We do not want to just minimize:

  $R_{train}[f] = (1/N) \Sigma_{k=1:N} L(f(\mathbf{x}^k), y^k)$

  We want to get best "generalization" (test error on future example).
- So we minimize $R_{reg}[f] = R_{train}[f] + \lambda \Omega[f]$

  $\Omega[f] = \|\mathbf{w}\|^2$ is a "regularizer".
- Learning can be performed by gradient descent:

  $\Delta \mathbf{w}_{total} = - \eta \nabla_w R - \gamma \mathbf{w}$ (Total gradient) $(\gamma = 2 \eta \lambda)$

  $\Delta \mathbf{w}_{stoc} = - \eta \nabla_w L - \gamma \mathbf{w}$ (Stochastic gradient)
- Linear model: $f(\mathbf{x}) = \mathbf{w} \cdot \Phi(\mathbf{x})$; $\nabla_w f = \Phi(\mathbf{x})$
- Chain rule: $\partial L/\partial w_i = \partial L/\partial f \, \partial f/\partial w_i$ or $\nabla_w L = \partial L/\partial f \, \nabla_w f$

  $\Delta \mathbf{w}_{stoc} = - \eta \, \partial L/\partial f \, \Phi(\mathbf{x}) - \gamma \mathbf{w}$

  $\Delta \mathbf{w}_{stoc}(\mathbf{x}^k) = - \eta \, \partial L/\partial f|_{(\mathbf{x}k, yk)} \, \Phi(\mathbf{x}^k) - \gamma \mathbf{w}$ for example k
- Total gradient:

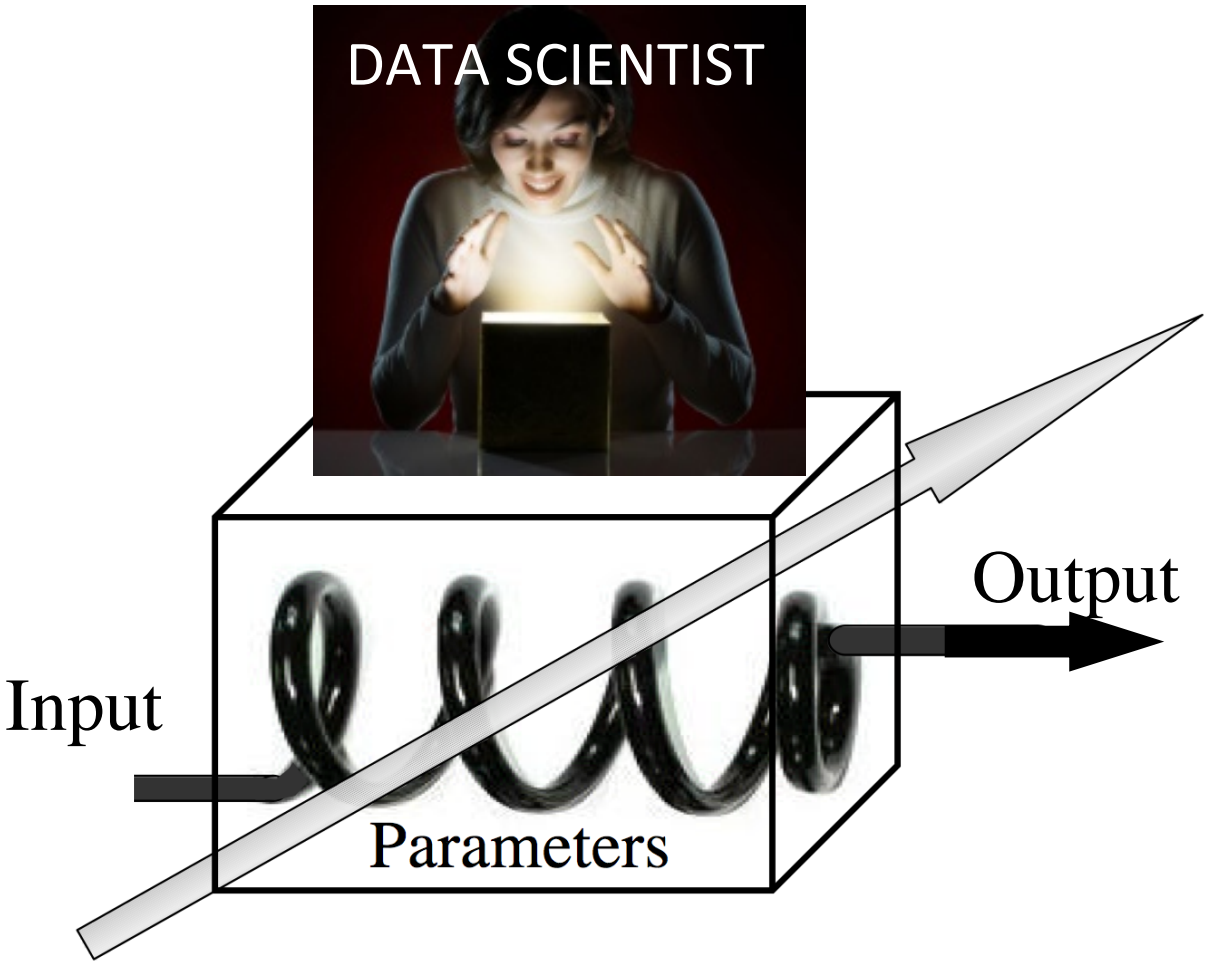
  $\Delta \mathbf{w}_{total} = (1/N) \Sigma_{k=1:N} \Delta \mathbf{w}_{stoc}(\mathbf{x}^k)$
- Kernel trick: $f(\mathbf{x}) = \Sigma_k \alpha_k k(\mathbf{x}^k, \mathbf{x}) = \Sigma_k \alpha_k \Phi(\mathbf{x}^k) \cdot \Phi(\mathbf{x})$; $\mathbf{w} = \Sigma_k \alpha_k \Phi(\mathbf{x}^k) \cdot$

  $\Delta \alpha_k = - \eta \, \partial L/\partial f|_{(\mathbf{x}k, yk)} - \gamma \alpha_k$ stochastic gradient, for example k

  $\Delta \alpha_h = - \gamma \alpha_h$ stochastic gradient, for other examples

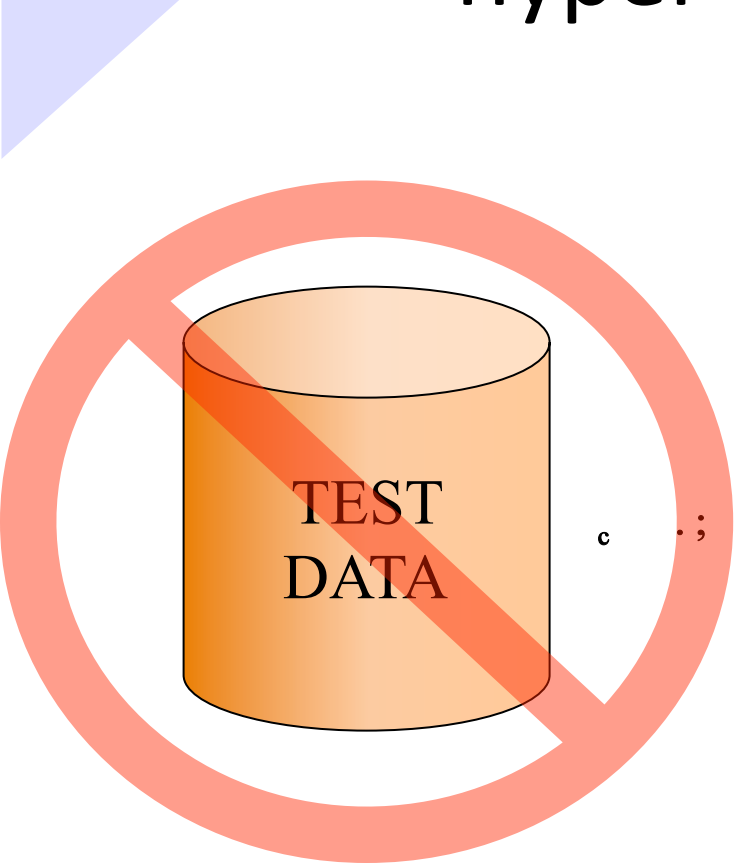
Review

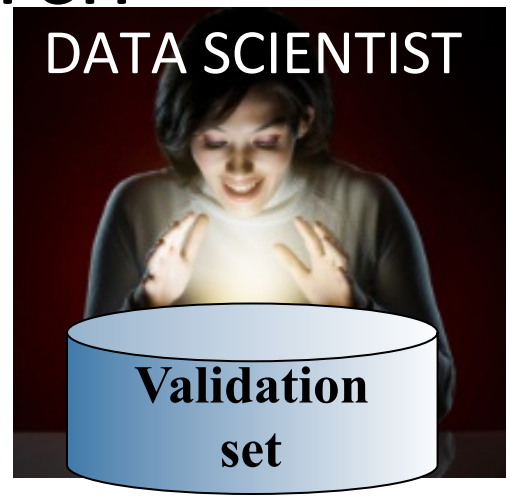
Model selection / hyper-parameter search

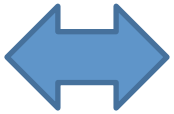


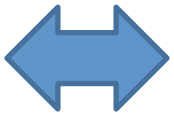
Review

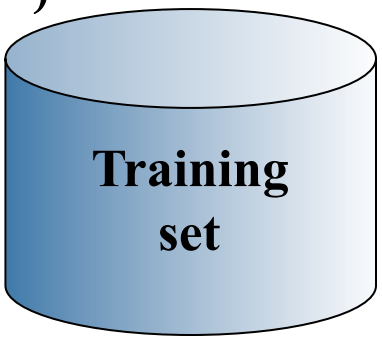
Model selection / hyper-parameter search









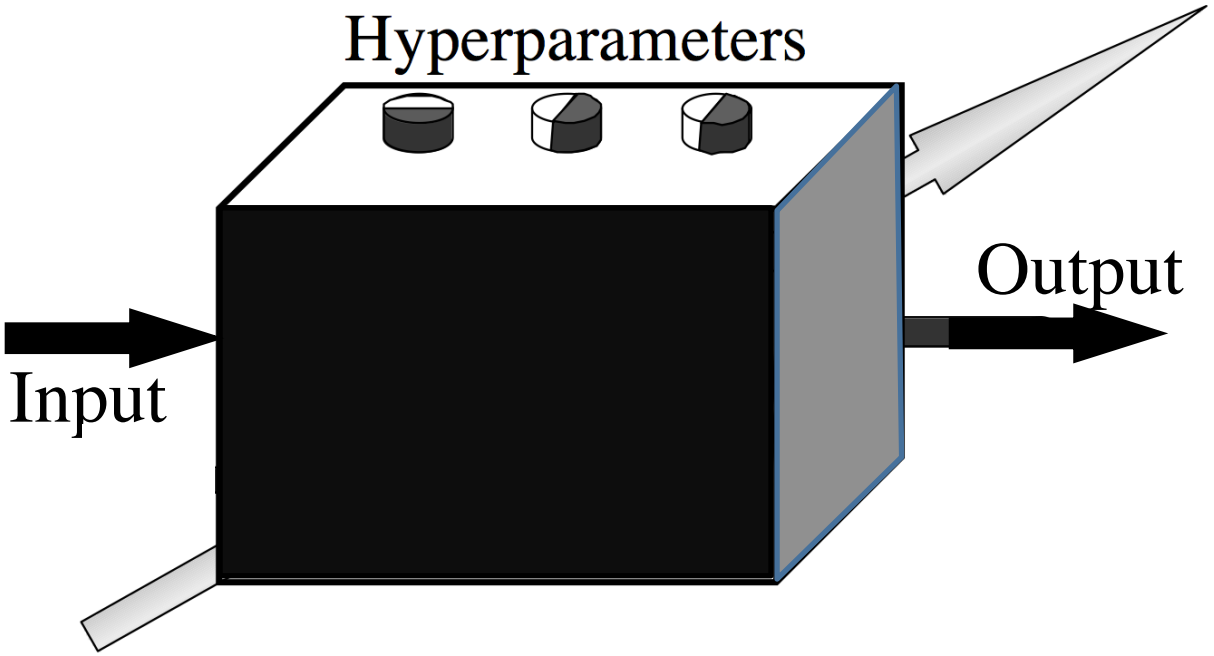


$$f^{**} = \text{argmin}_{\theta} R_2[f^*, D] \;, \quad \text{such that} \quad f^* = \text{argmin}_{\alpha} R_1[f, D]$$

Review

Wrappers don't scale well

θ



D=dim(θ), g=number of values of θ_i, K-fold CV evaluation: Kg^D boxes to train!

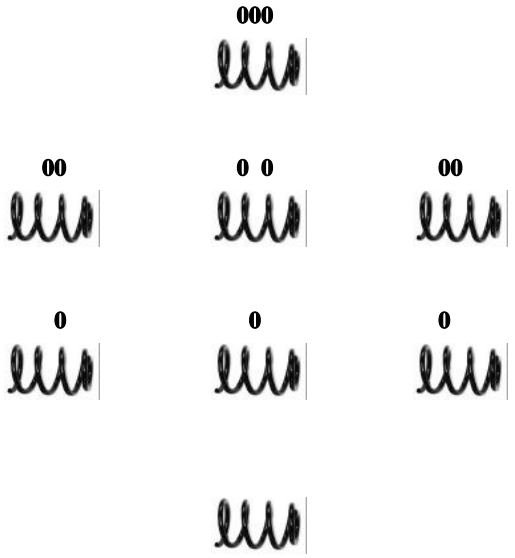
D=10, g=10, K=10: $Kg^D=10^{11}$

Review

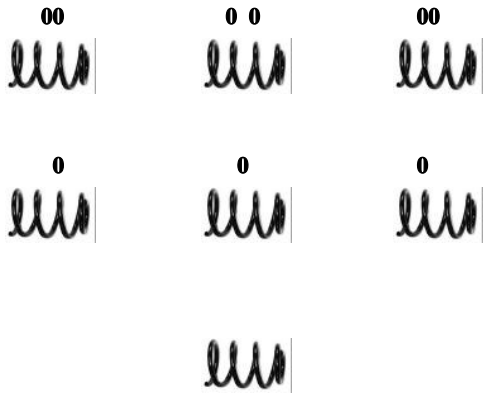
$Kg^D=10^{11}$ boxes to train!

Combine search strategies:

1) Filters



3) Wrappers



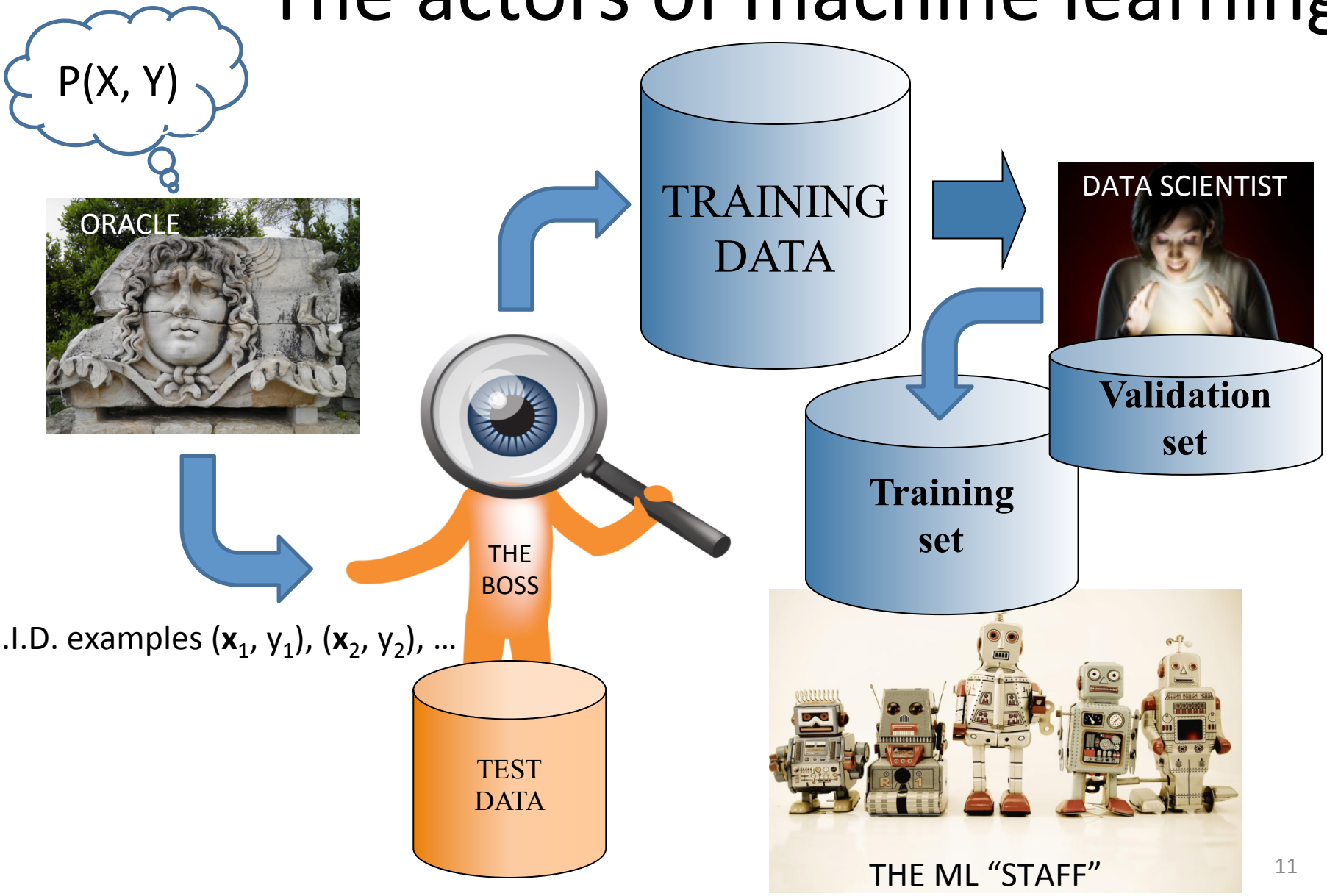
D=10 → D=4 → D=2

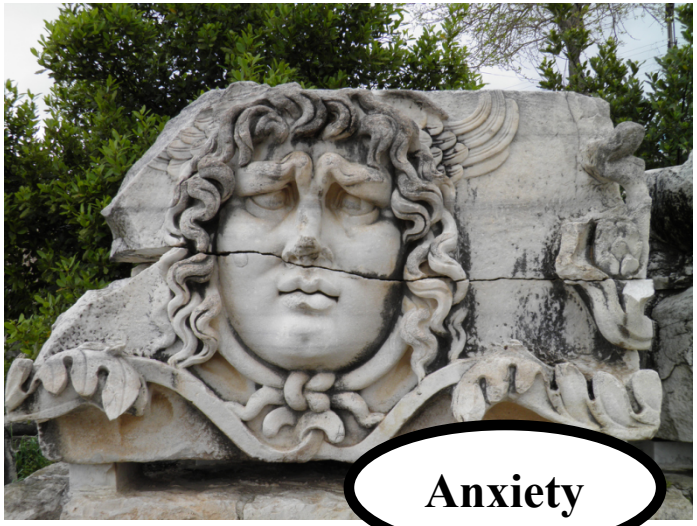
No training

1for λ x 10 for θ = 10

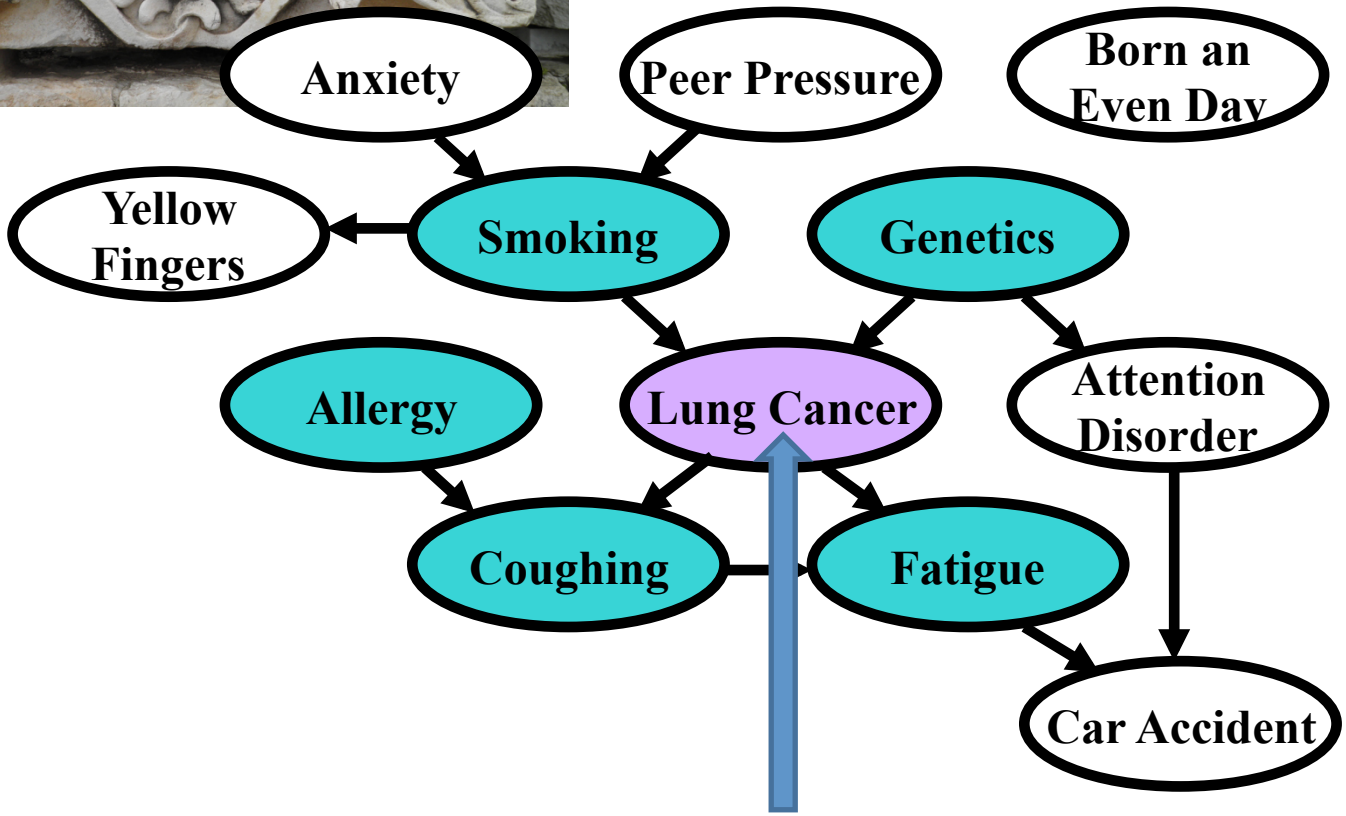
$Kg^D=10*10^2=1000$

The actors of machine learning





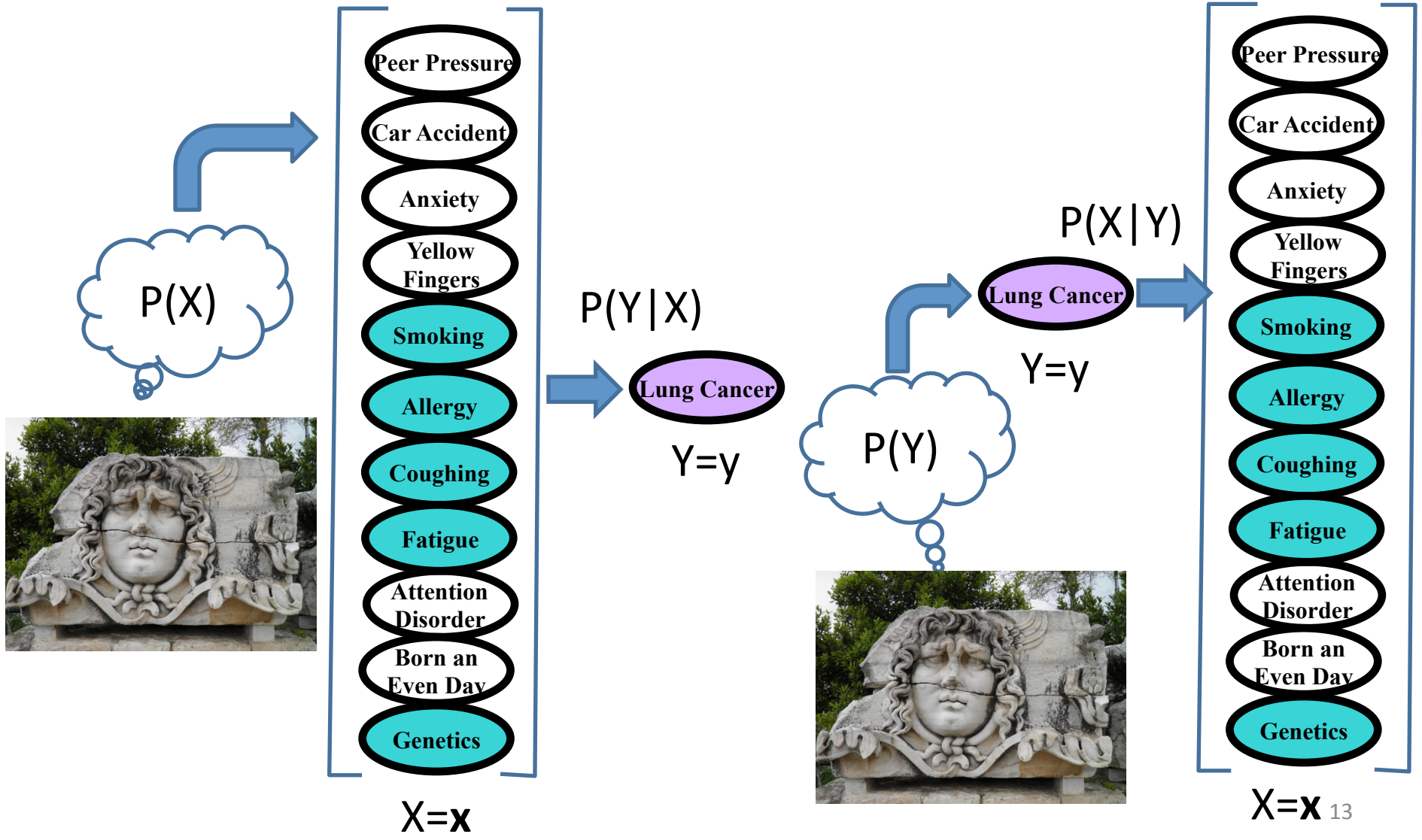
What does the Oracle have to say?



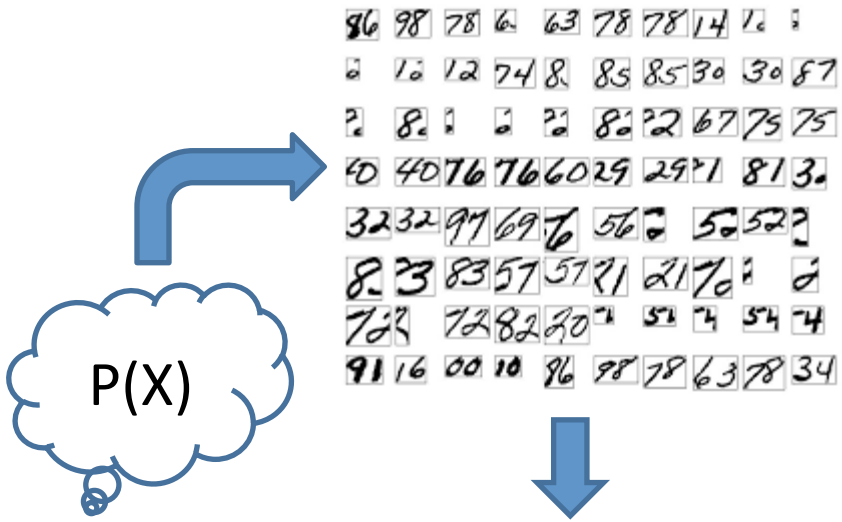
This is y (the rest are x_i)

But, here we consider only

$$P(X, Y) = P(X)P(Y|X) = P(Y)P(X|Y)$$



But, here we consider only
$P(X, Y) = P(X)P(Y|X) = P(Y)P(X|Y)$



P(X)

P(Y|X)

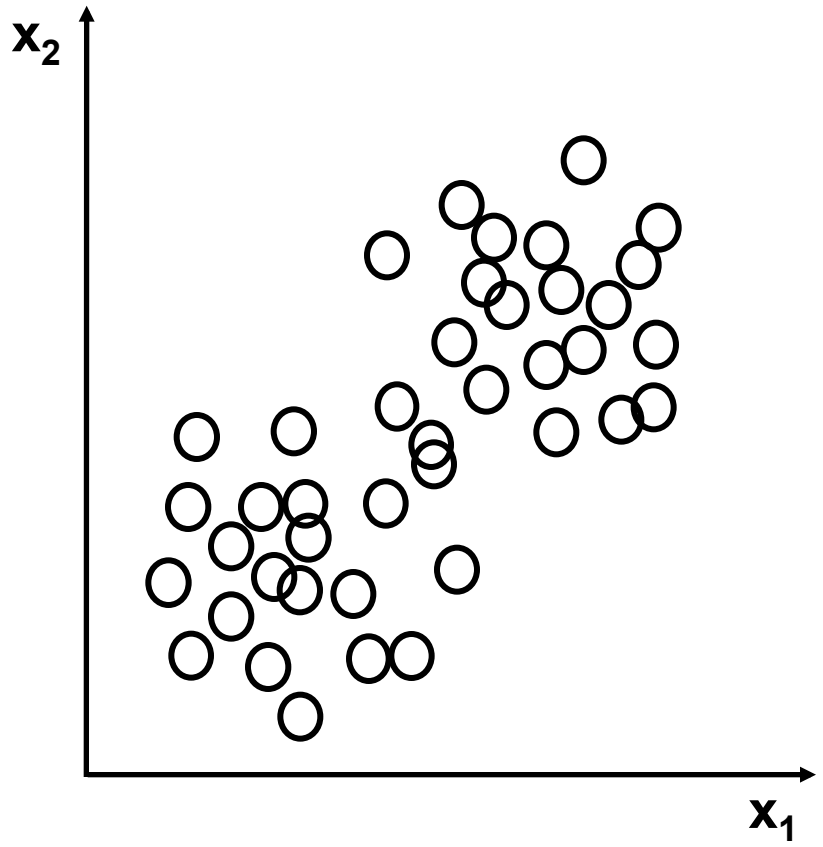
96 98 78 6 63 79…

5

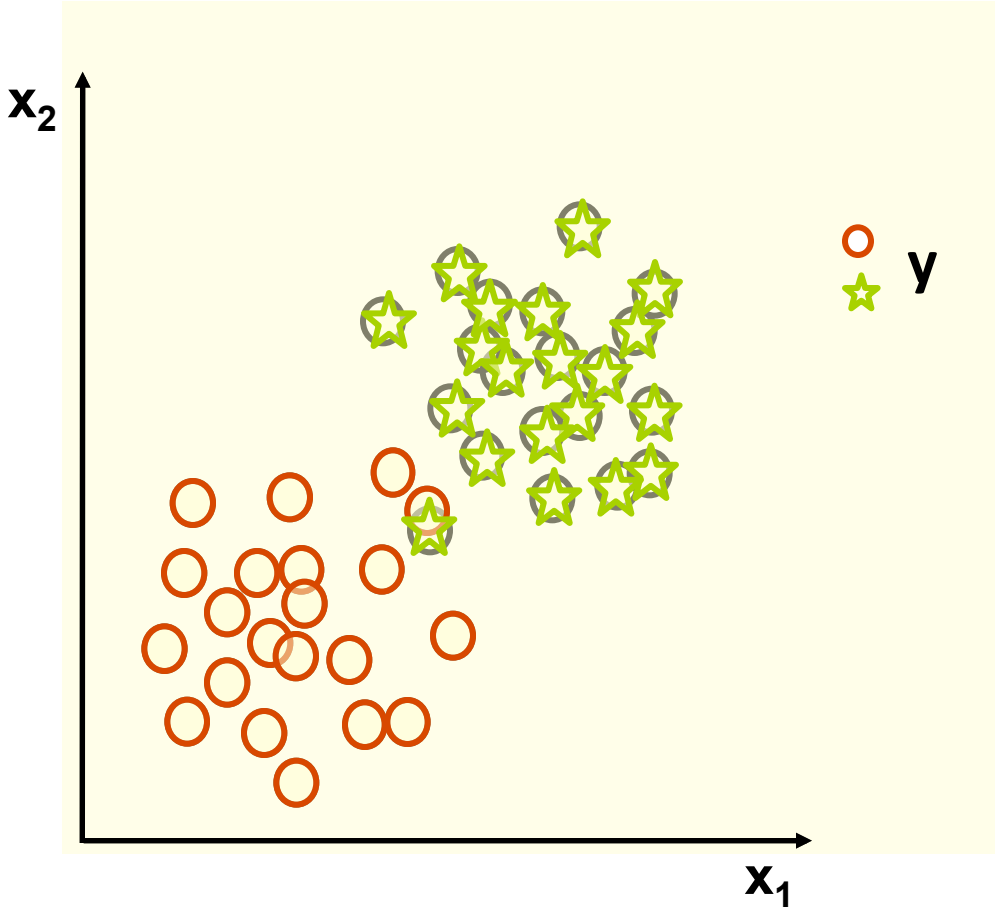
P(X|Y)

P(Y)

What is the data generating process?

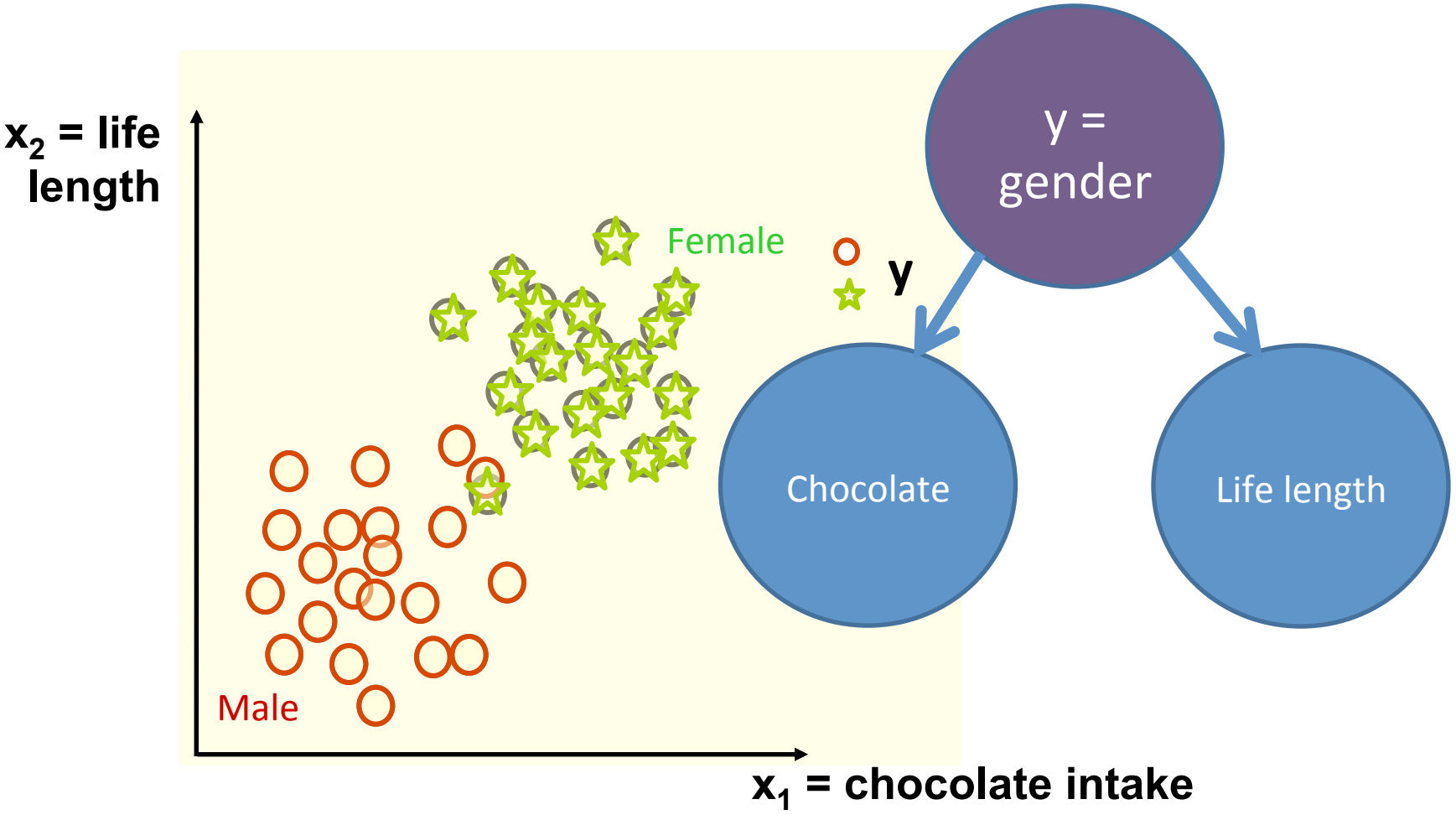


There can be a common cause y



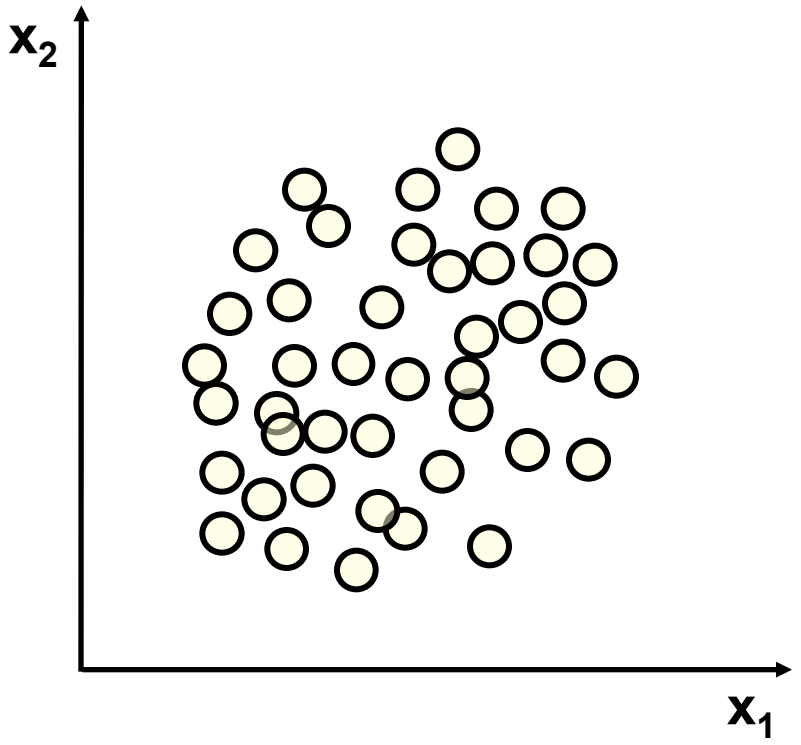
$$x_1 \perp x_2 \mid y$$

There can be a common cause y



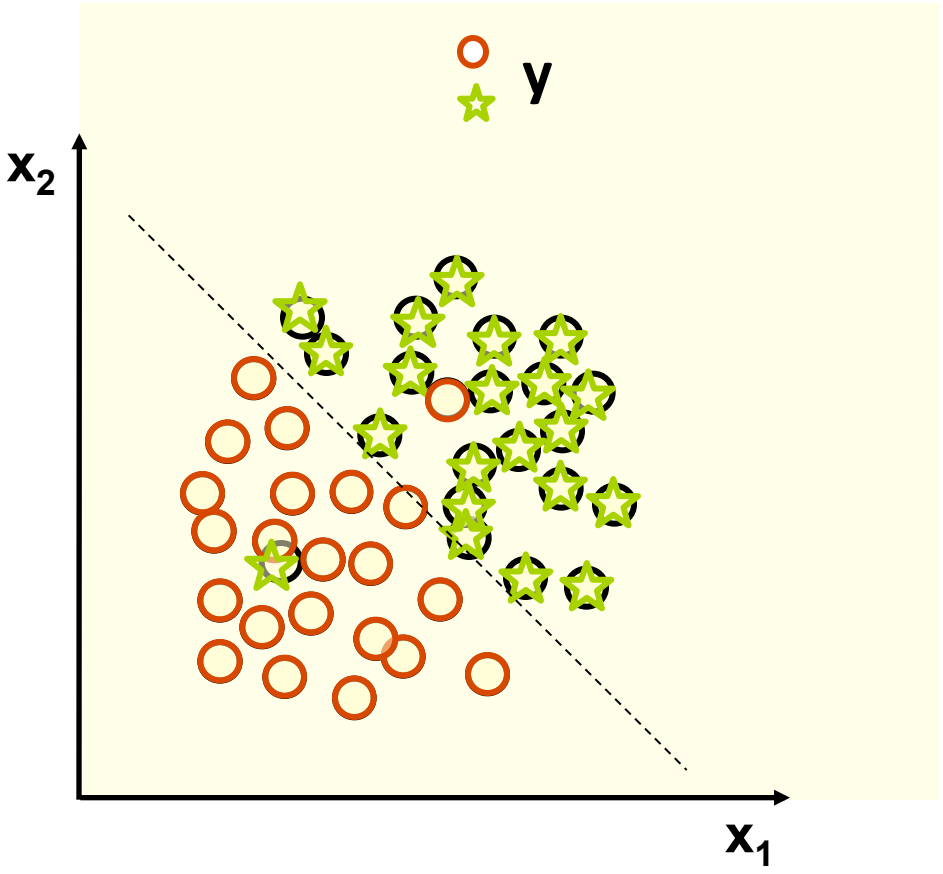
$$x_1 \perp x_2 \mid y$$

What is the data generating process?

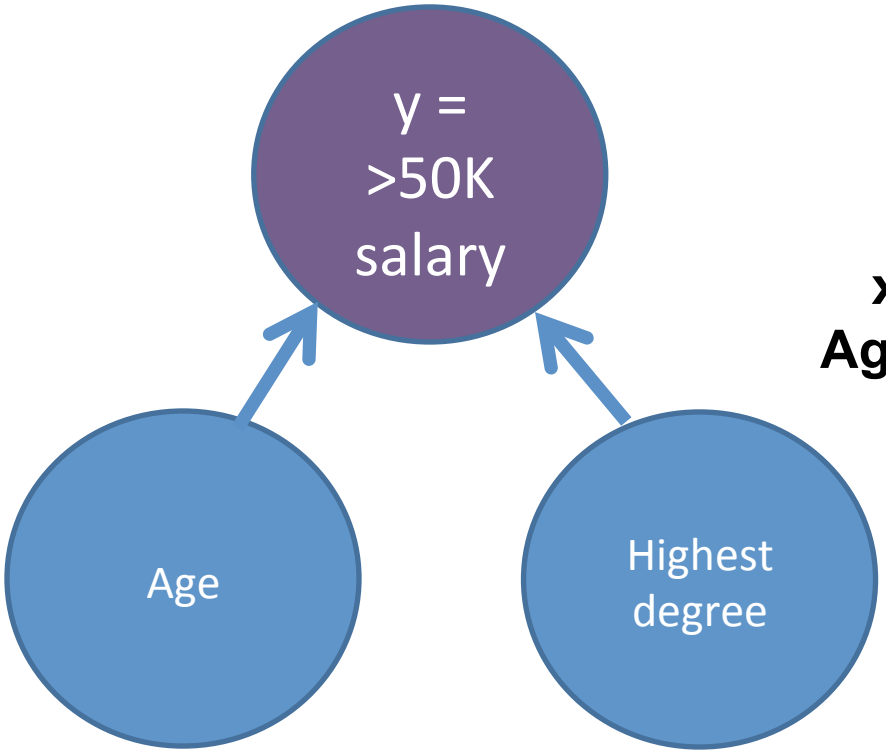


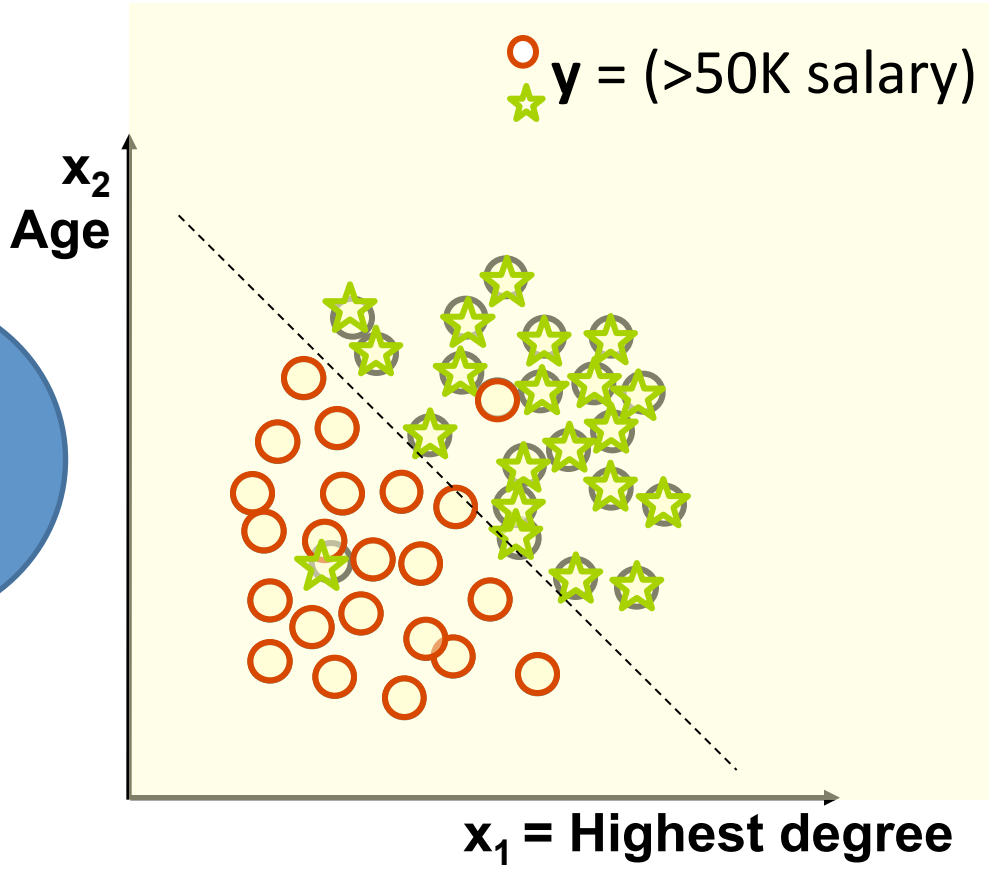
$$x_1 \perp x_2$$

y must be a consequence



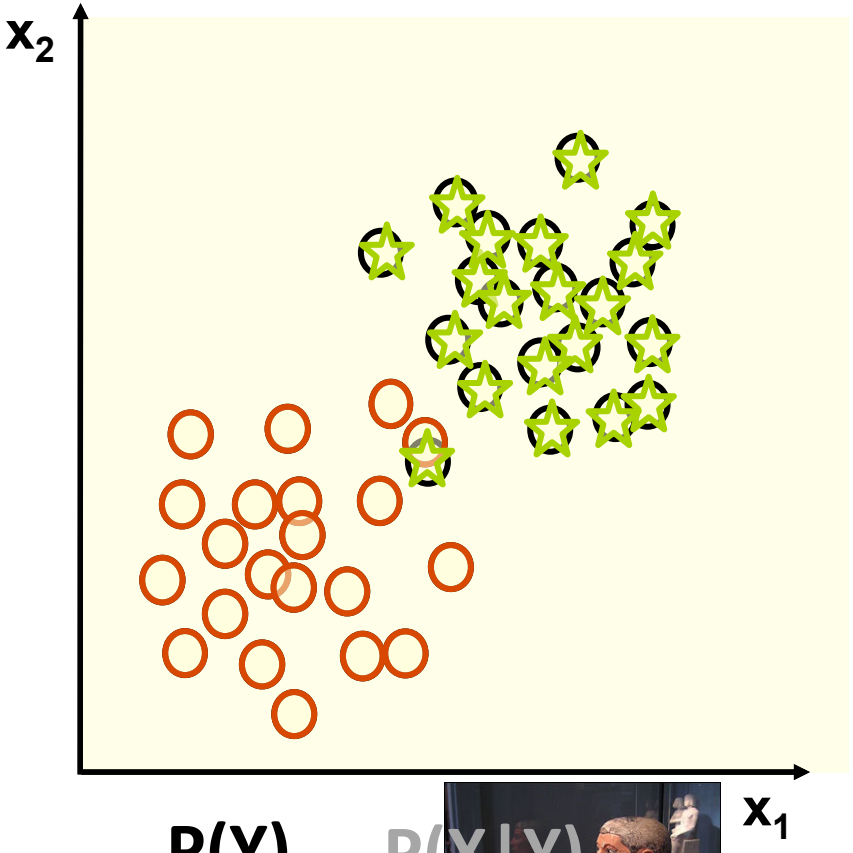
y must be a consequence





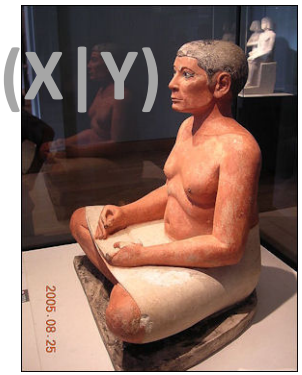
$x_1 \mid x_2$

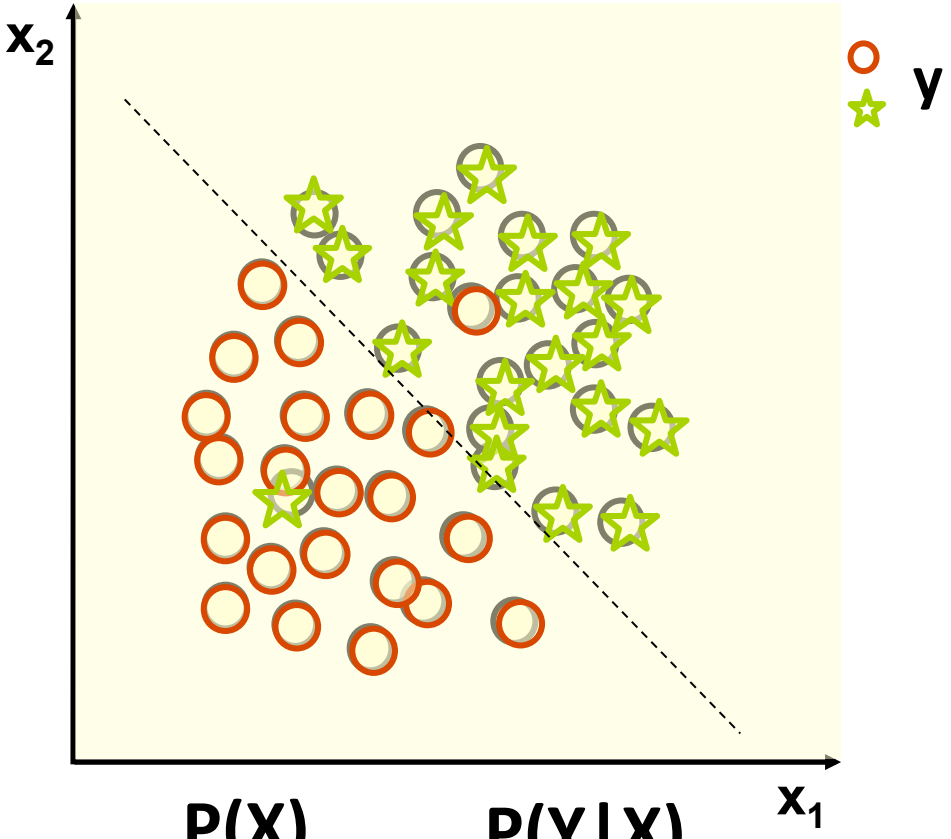
So: we have 2 cases:



P(Y)

P(X|Y)



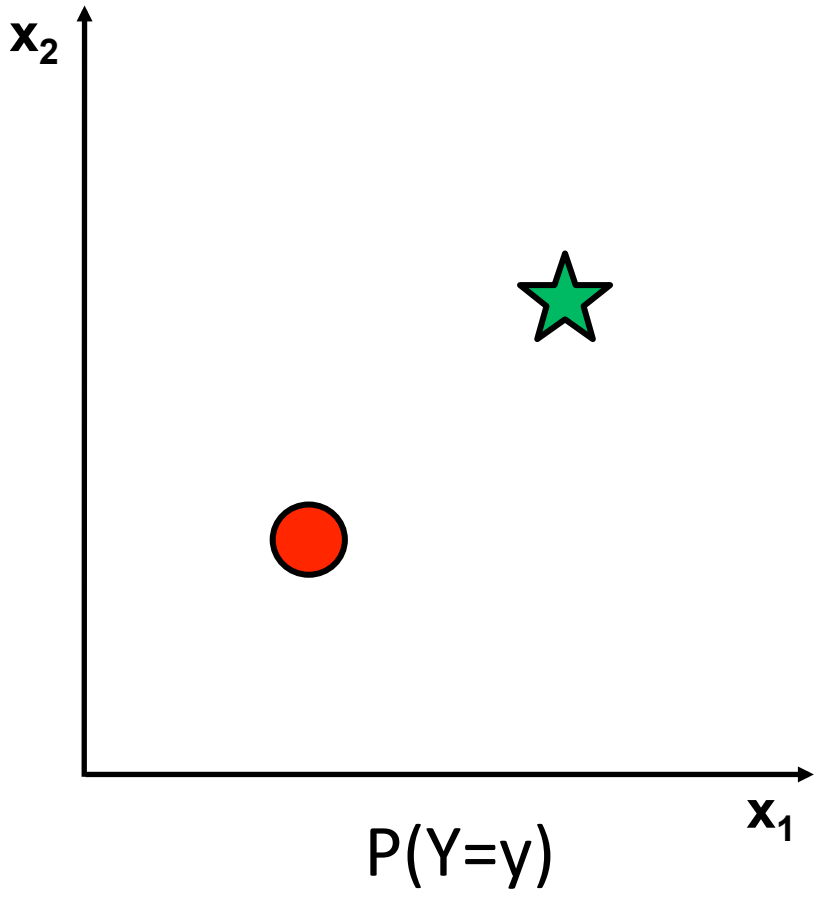


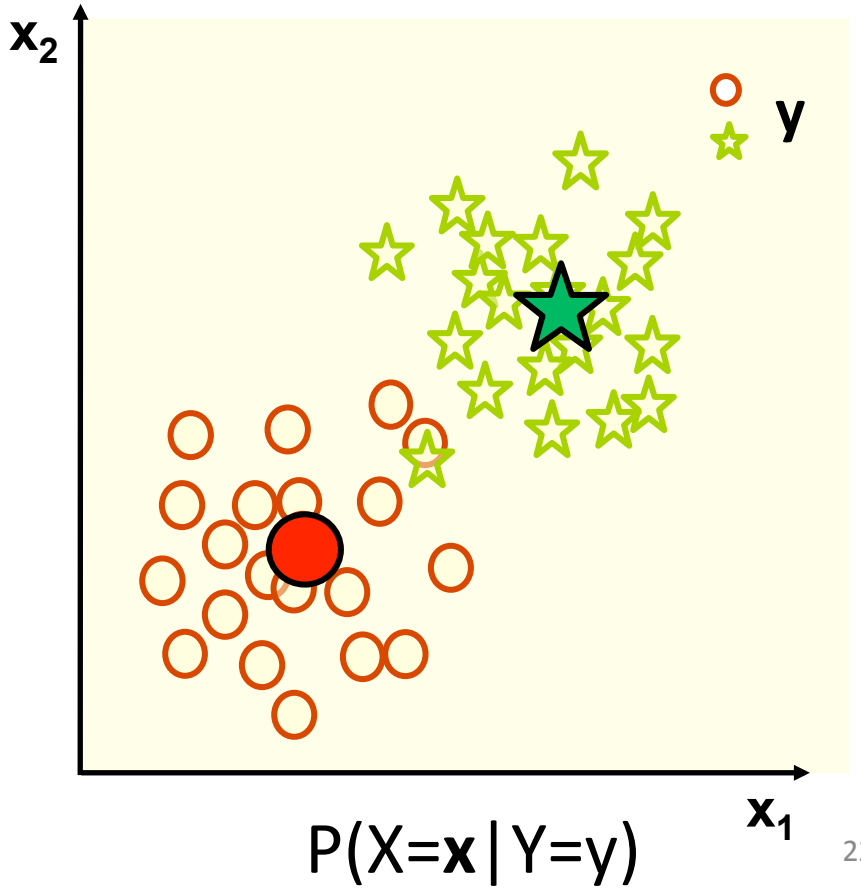
P(X)

P(Y|X)

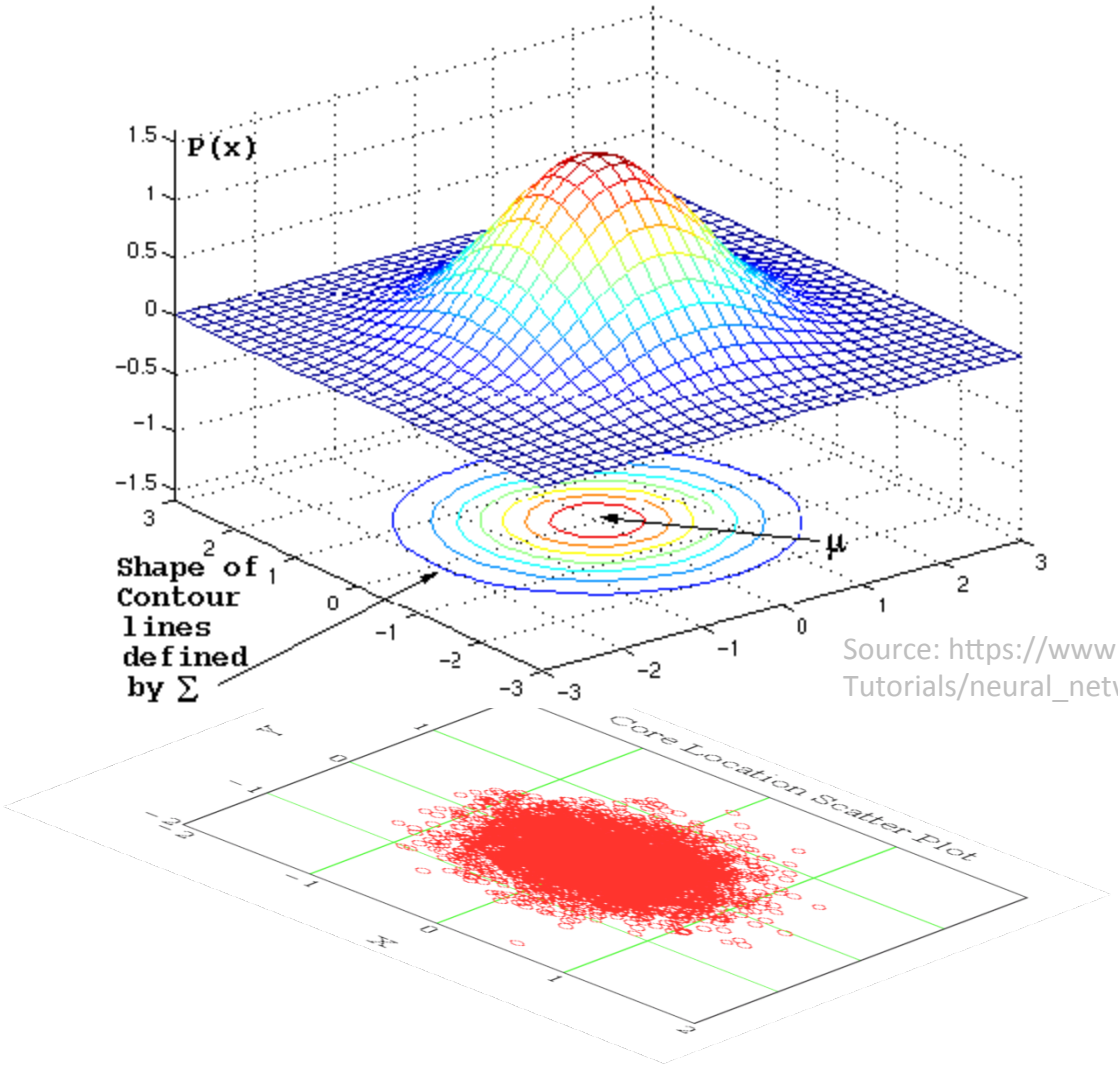
Consider the first case

1) y drawn at random determines the centroid.
2) **x** = centroid + random "noise".





The easiest is to assume isotropic Gaussian noise ~exp(-$\|\mathbf{x}-\boldsymbol{\mu}\|^2/2\sigma^2$)



Source: https://www.byclb.com/TR/Tutorials/neural_networks/ch4_1.htm

So we get the Gaussian classifier

- Bayes optimal classifier: Classify **x** in class y according to the largest $P(Y=y|X=\mathbf{x})$.
- Bayes rule: $P(X, Y) = P(X)P(Y|X) = P(Y)P(X|Y)$.

  So $\underbrace{P(Y=y|X=\mathbf{x})}_{\text{Posterior}} \sim \underbrace{P(Y=y)}_{\text{Prior}} \underbrace{P(X=\mathbf{x}|Y=y)}_{\text{Likelihood}}$.

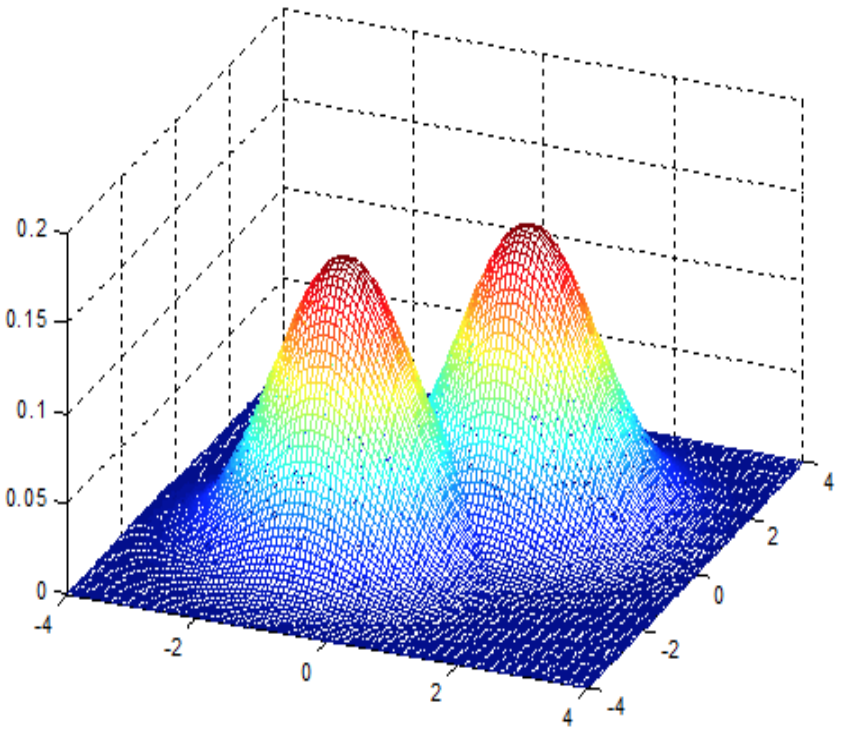
- Prior = class relative abundance:

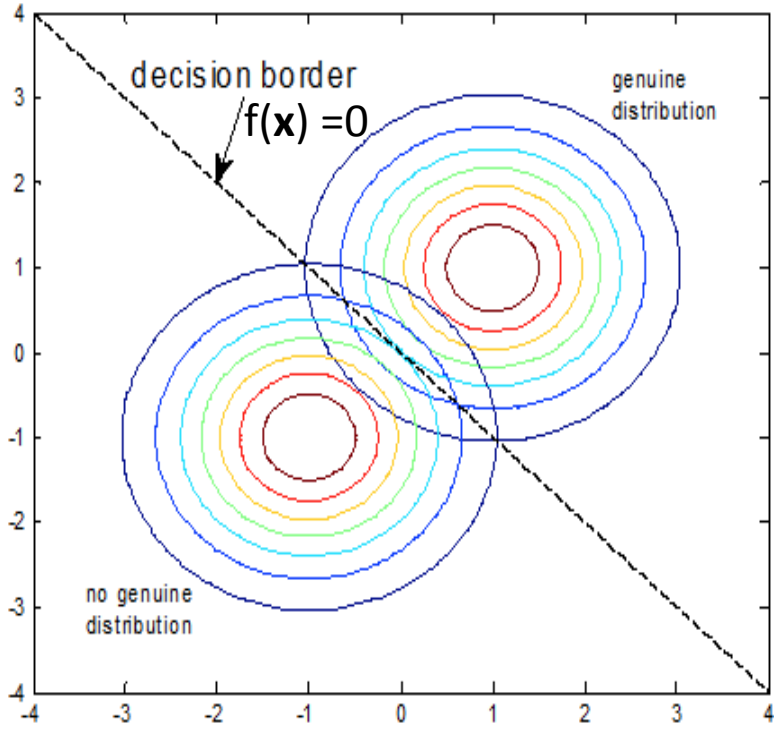
  $P(Y=1) \approx N_1/N$. $P(Y=-1) \approx N_0/N$.

- Likelihood = Proba **x** belongs to class y, proportional to $\exp(-\|\mathbf{x}-\boldsymbol{\mu}^{[y]}\|^2/2\sigma^2)$.

No sweat: this is a linear classifier!

Source: http://www.intechopen.com/source/html/17742/media/image25.png



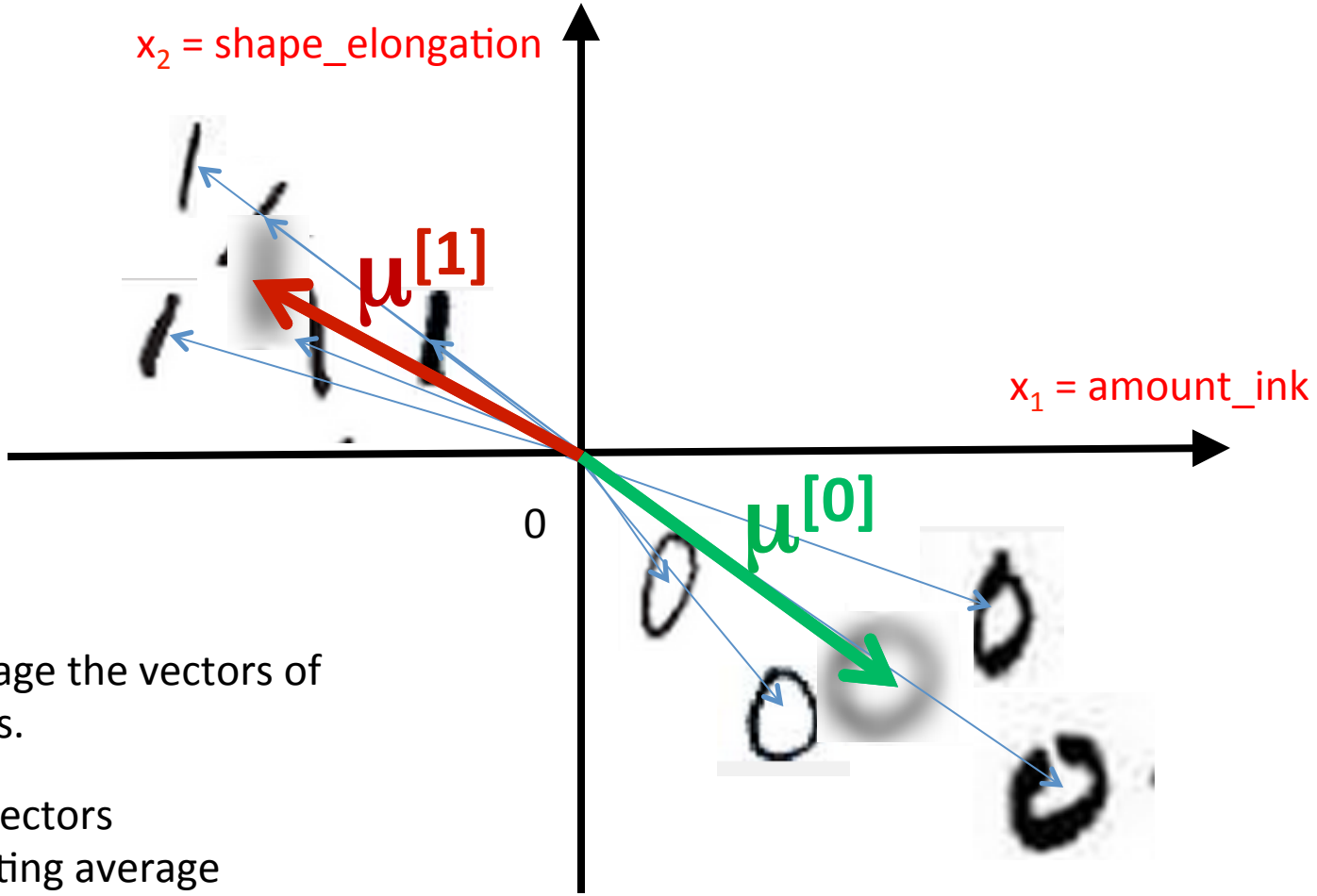


$$f(\mathbf{x}) = \underbrace{(\mu^{[1]} - \mu^{[0]})}_{\mathbf{w}} \bullet x + b$$

$$b = (\mu^{[0]2} - \mu^{[1]2})/2 + \log(N_1/N_0)$$

Does this remind you of something?

The "centroids" method (also called "template matching")!

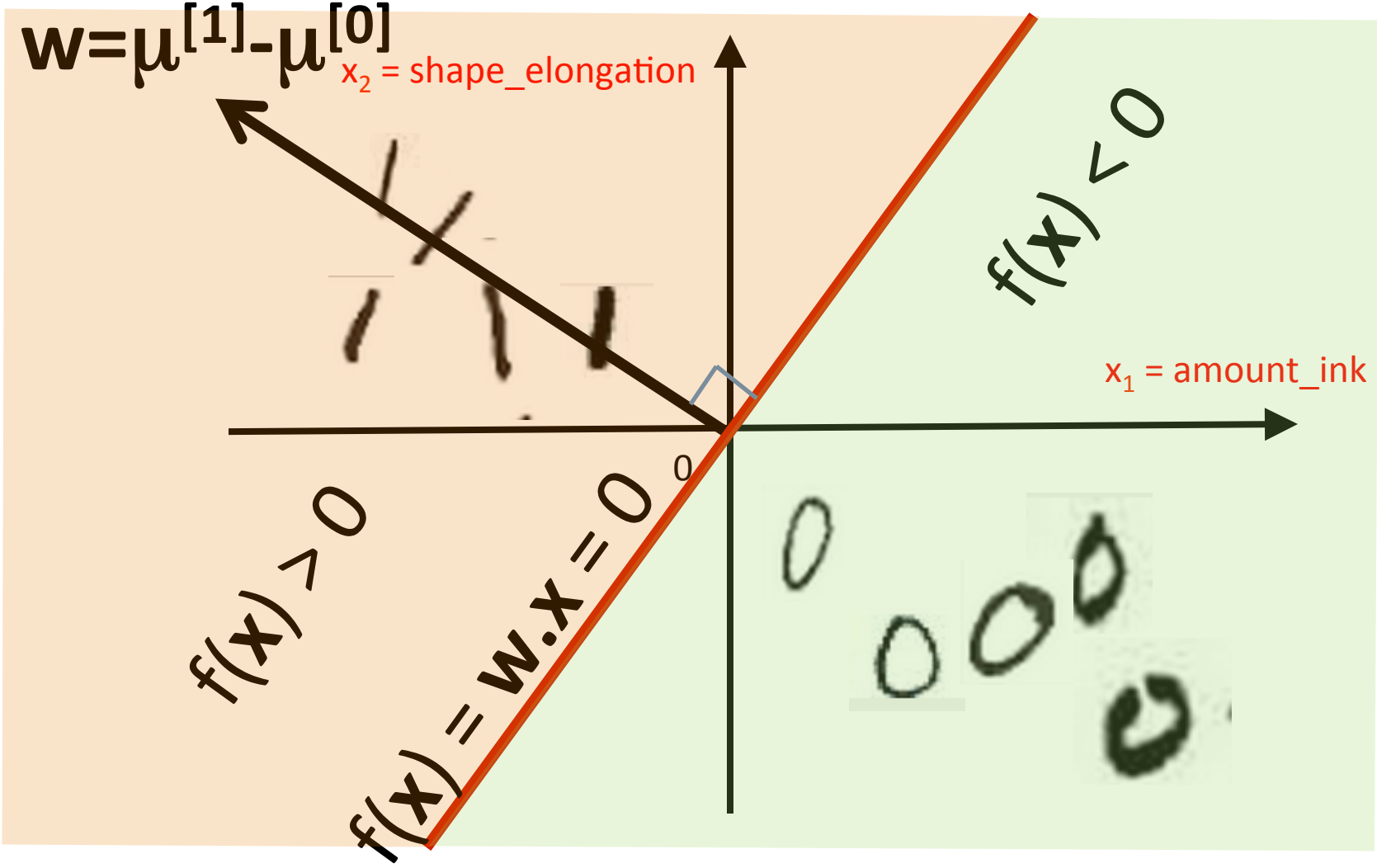


Just average the vectors of each class.

You get vectors representing average shapes.

Lecture #2: separate “0” and “1”:

get the separaring “hyperplane”

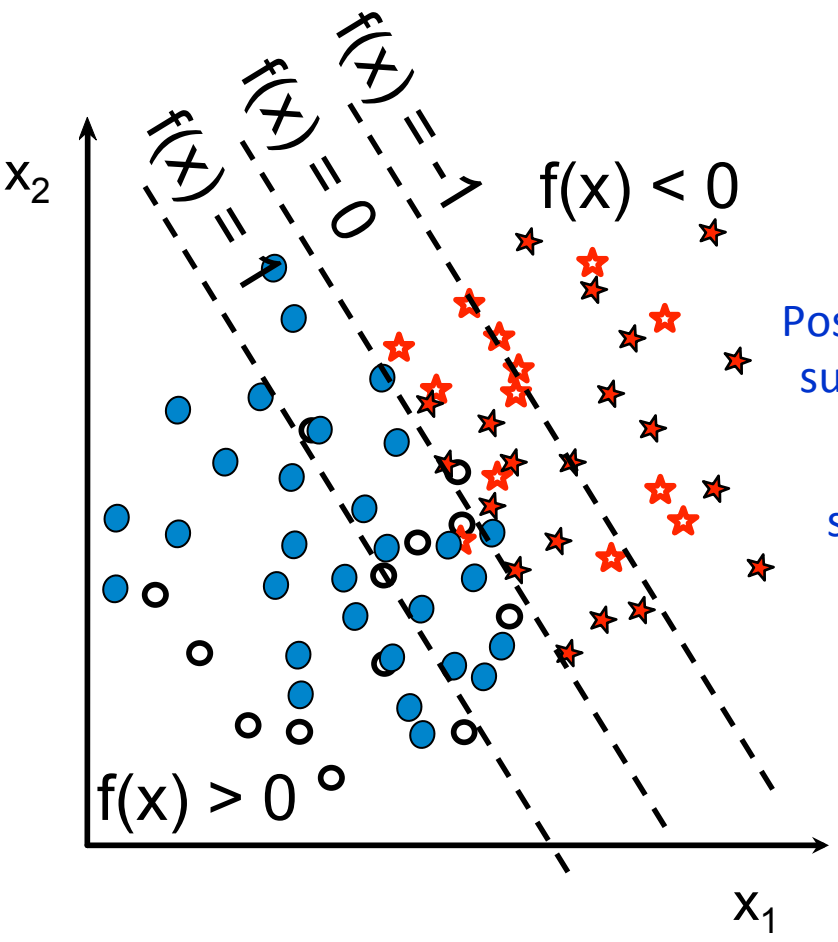


So: Gaussian classifier = centroid method

- This is a VERY useful method.
- The assumptions are:
  - Generating model: draw y first, then draw **x** given y.
  - Gaussian "noise".
  - Independence of the features in a given class (no covariance).
  - Same σ for all classes.
- This NOT the optimum Bayes classifier (even though it uses Bayes rule), because:
  - The assumptions are almost always violated.
  - We estimate $\boldsymbol{\mu}^{[1]}$ and $\boldsymbol{\mu}^{[0]}$ from a finite number of examples.
- Now we know how to compute the bias
  $b = (\boldsymbol{\mu}^{[0]2} - \boldsymbol{\mu}^{[1]2})/2 + \log(N_1/N_0)$, but who cares?

We do not care about "b" because we "**post fit it**"

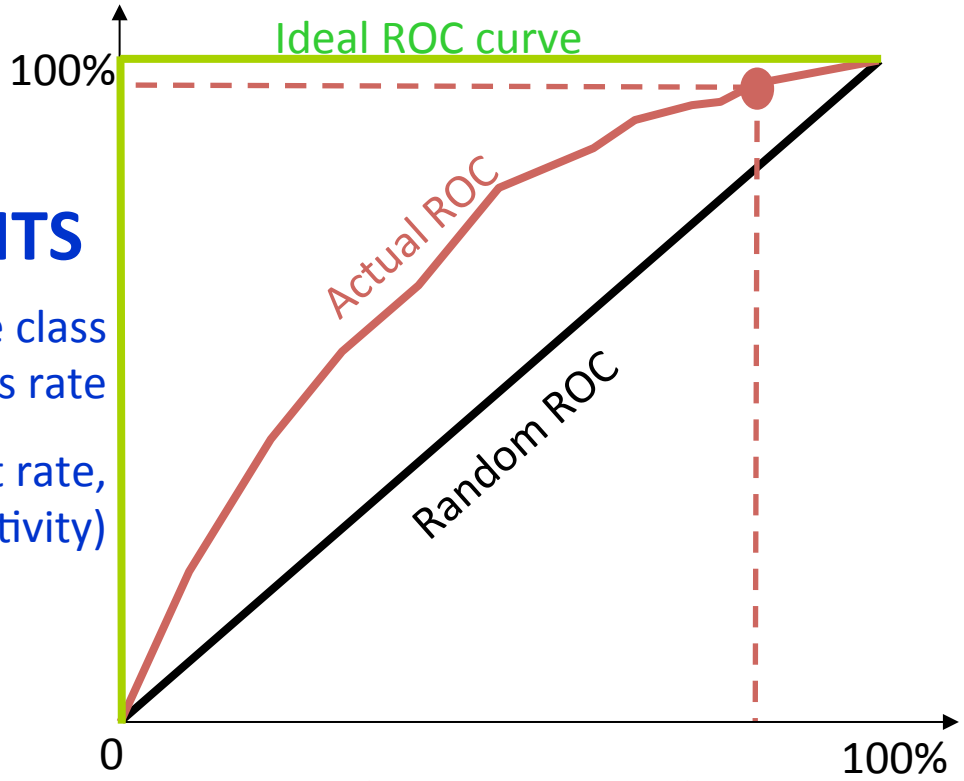
For a given threshold on f(x), you get a point on the ROC curve.



HITS

Positive class success rate

(hit rate, sensitivity)



FALSE ALARMS

1 - negative class success rate
(false alarm rate, 1-specificity)

Link with Hebb's rule…

- Class centroids:

$$\boldsymbol{\mu}^{[0]} = (1/N_0) \sum_{\{y=-1\}} \mathbf{x}^k$$

$$\boldsymbol{\mu}^{[1]} = (1/N_1) \sum_{\{y=+1\}} \mathbf{x}^k$$

- Weight vector:

$$\mathbf{w} = \boldsymbol{\mu}^{[1]} - \boldsymbol{\mu}^{[0]} = \sum_k y^k \mathbf{x}^k$$

$y^k = 1/N_1$ if $\mathbf{x}^k$ belongs class 1,

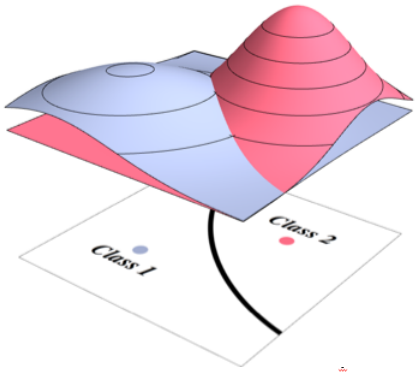
$y^k = -1/N_0$ otherwise.

So this is Hebb's rule with "rebalanced" target values.

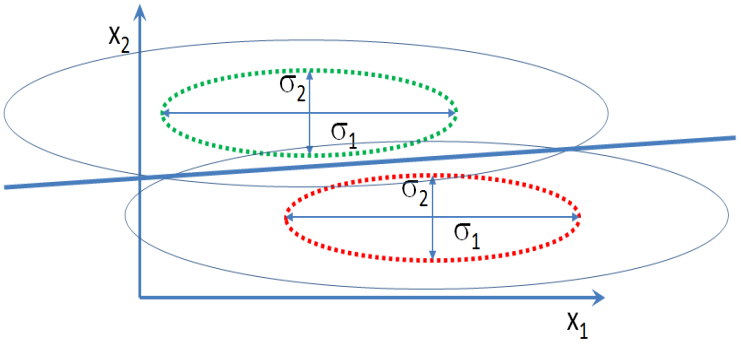
It can trivially be "kernelized" with $\alpha^k = y^k$ using the linear kernel $k(\mathbf{x}, \mathbf{x}') = \mathbf{x} \bullet \mathbf{x}'$

Extensions

1) Different σ for the 2 classes: $\sigma^{[0]}$ and $\sigma^{[1]}$.

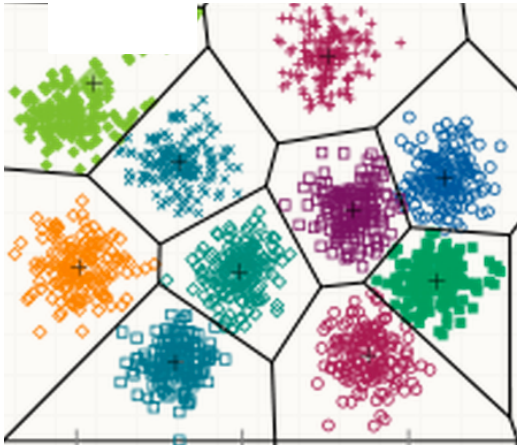


2) Different σ_i for all coordinates (but same for both classes).
$f(\mathbf{x}) = \sum_i [(\mu_i^{[1]} - \mu_i^{[0]})/\sigma_i^2] x_i + b$

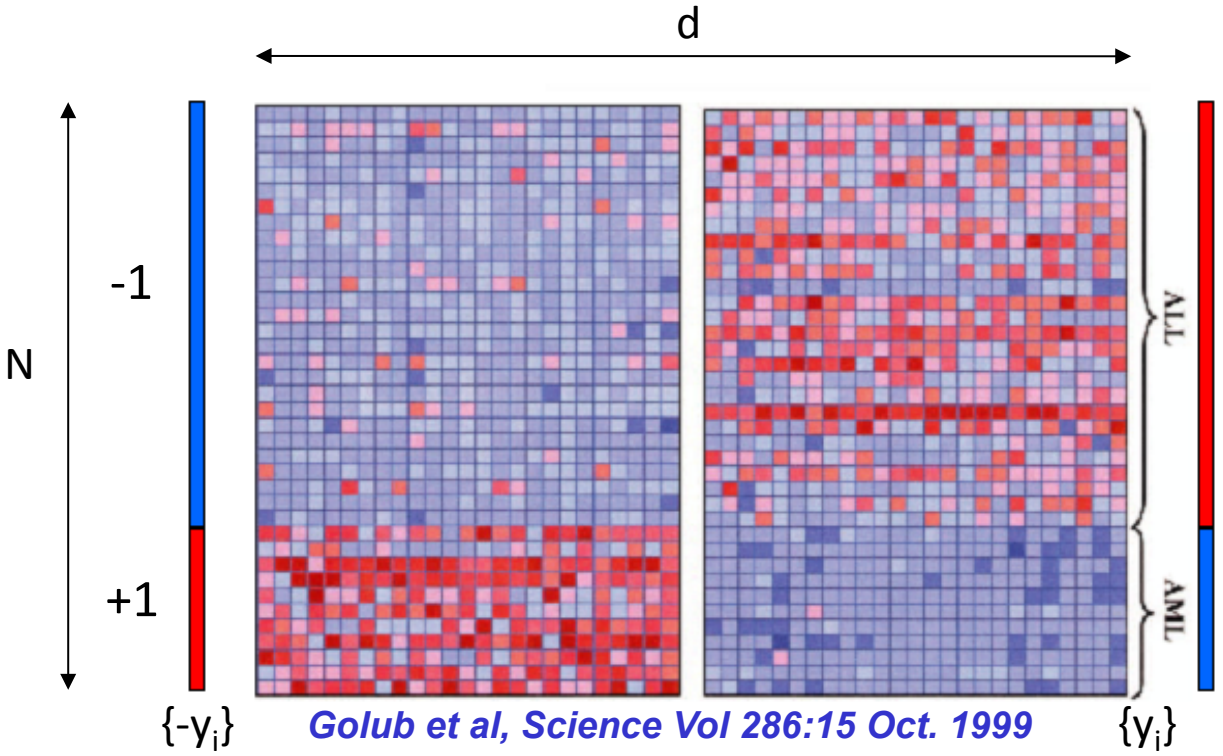


3) Different $\sigma_i^{[0]}$ and $\sigma_i^{[1]}$.

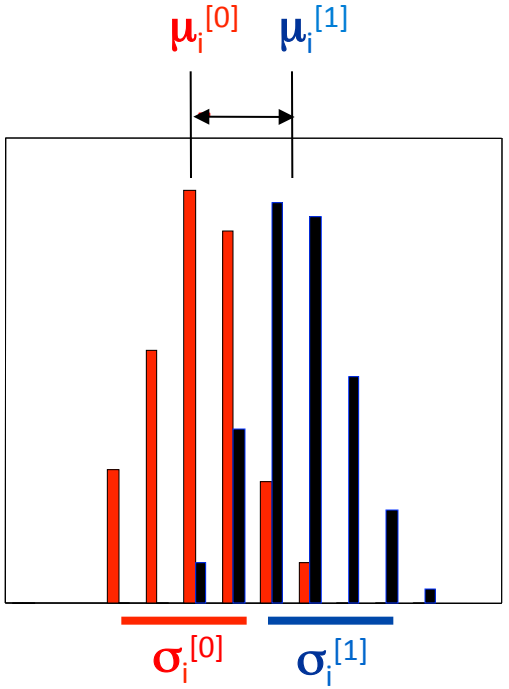
4) Multiclass.



Gaussian classifier, T-test, and univariate feature selection



Golub et al, Science Vol 286:15 Oct. 1999



$$w_i = \frac{\mu_i^{[1]} - \mu_i^{[0]}}{\sigma_i^2} \qquad \sigma_i \cong \tfrac{1}{2}\left(\sigma_i^{[0]} + \sigma_i^{[1]}\right)$$

Rank features according to "signal-to-noise" ratio:

$$S2N = |\, \sigma_i w_i \,| \cong |\, \text{corrcoef} \,| \sim |\, \mathbf{x} \bullet \mathbf{y} \,|$$

For "balanced classes and after "standardization" $\mathbf{x} \leftarrow (\mathbf{x}-\mu_x)/\sigma_x$

Link with "Naïve Bayes"

- Naïve Bayes classifies according to $\text{argmax}_y\ P(Y=y|X=\mathbf{x})$ making **independence assumptions** between variables.
- Independence between variables given Y means:

$$P(X_1, X_2, \ldots X_N|Y) = \Pi_{i=1:N}\ P(X_i|Y)$$

So Naïve Bayes means classify according to:

$$\text{argmax}_y\ \Pi_{i=1:N}\ P(X_i=x_i|Y=y)\ P(Y=y)$$

- The Gaussian classifier also makes independence assumptions:

$$P(X|Y) \sim \exp(-\|\mathbf{x}-\boldsymbol{\mu}\|^2/2\sigma^2) = \exp(-\textstyle\sum_{i=1:N}(x_i-\mu_i)^2/2\sigma^2)$$
$$= \Pi_{i=1:N}\exp(-(x_i-\mu)^2/2\sigma^2) \sim \Pi_{i=1:N}\ P(X_i|Y)$$

"Naïve Bayes" for binary inputs

$$\hat{y} = \text{argmax}_y \prod_{i=1:N} P(X_i=x_i|Y=y)\, P(Y=y)$$

- Binary variables $x_i \in \{0, 1\}$, $y \in \{-1, 1\}$
- Each factor $P(X_i=x_i|Y=y)$ and $P(Y=y)$ can be estimated from frequency counts:

$P(X_i=1|Y=1) = f_{11i}$ $\quad P(X_i=0|Y=1) = f_{01i}$ $\quad P(Y=1) = f_1$

$P(X_i=1|Y=-1) = f_{10i}$ $\quad P(X_i=0|Y=-1) = f_{00i}$ $\quad P(Y=-1) = f_0$

$$f(\mathbf{x}) = \log(f_1/f_0) + \sum_{i=1:N} x_i \log P(X_i=1|Y=1) + (1-x_i) \log P(X_i=0|Y=1) - x_i \log P(X_i=1|Y=-1) - (1-x_i) \log P(X_i=0|Y=-1)$$

$$f(\mathbf{x}) = \sum_{i=1:N} w_i x_i + b = \mathbf{w} \bullet \mathbf{x} + b$$

$$w_i = \log(f_{11i}/f_{10i}) - \log(f_{01i}/f_{00i})$$

$$b = \sum_{i=1:N} \log(f_{01i}/f_{00i}) + \log(f_1/f_0)$$

- Extends to categorical variables (instead of binary).

Generative models

You are here

LDA

Naïve Bayes

Mixture models

Gaussian processes

Summary

- The Gaussian classifier is a "generative model" modeling $P(X=\mathbf{x}|Y=y) \sim \exp(-\|\mathbf{x}-\boldsymbol{\mu}^{[y]}\|^2/2\sigma^2)$.
- Using Baye's rule,

$$P(Y=y|X=\mathbf{x}) \sim P(Y=y)\, P(X=\mathbf{x}|Y=y)$$

one can make "Bayes optimal"-style decisions according to

$$\max_y P(Y=y|X=\mathbf{x}).$$

- The main assumption is the "naïve Bayes" assumption of **statistical independence** of the input variables:

$$P(X=\mathbf{x}|Y=y) = \Pi_{i=1:N} P(X_i=x_i|Y=y)$$

- If the 2 classes have the same σ, the Gaussian classifier is a linear discriminant, aka the centroid method, aka Hebb's rule with target values $1/N_1$ and $-1/N_0$.
- "Naïve Bayes" is NOT "Bayes optimal" (because of independence assumptions and the estimation of parameters from a finite training set, but it is **very useful**).

Come to my office hours…
Wed 2:30-4:30 Soda 329

Next time: LDA