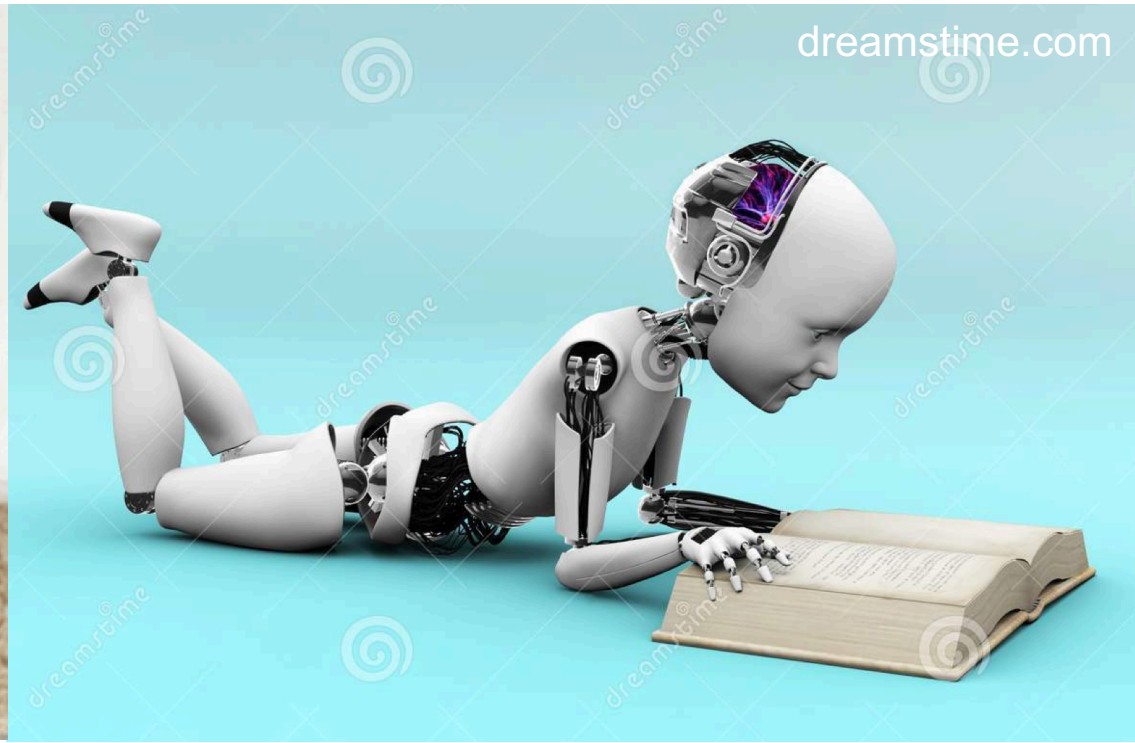
CMSC 510 – L02
Regularization Methods for Machine Learning

Instructor:
Dr. Tom Arodz

Machine Learning

- Learning from experience
  vs absorbing knowledge



Machine Learning

- Machine learning:
  - Study of algorithms that
    - improve their performance P
    - at some task T
    - with experience E
  - We have a well-defined learning task: <P,T,E>

Machine Learning

- Machine learning: study of algorithms that
  - improve their performance P
    - HOW TO MEASURE THE PERFORMANCE?
  - at some task T
    - WHAT IS THE TASK? HOW TO ABSTRACT IT?
  - with experience E
    - HOW IS EXPERIENCE GAINED?

Supervised Machine Learning

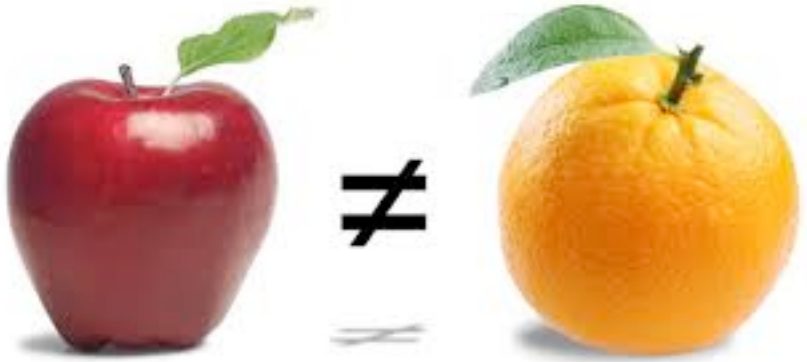
- **Algorithms** for solving the learning problem:

  <Performance, Task, Experience>

- Supervised Machine Learning/Classification:
  - Task:
    - Learn the ability to categorize objects:
      to predict the class of object
      based on some attributes/features of the object
  - Performance:
    - Measured e.g. as the accuracy of predictions
      for previously unseen objects
  - Experience:
    - A collection of objects, each described by its attributes, and labeled with its classes

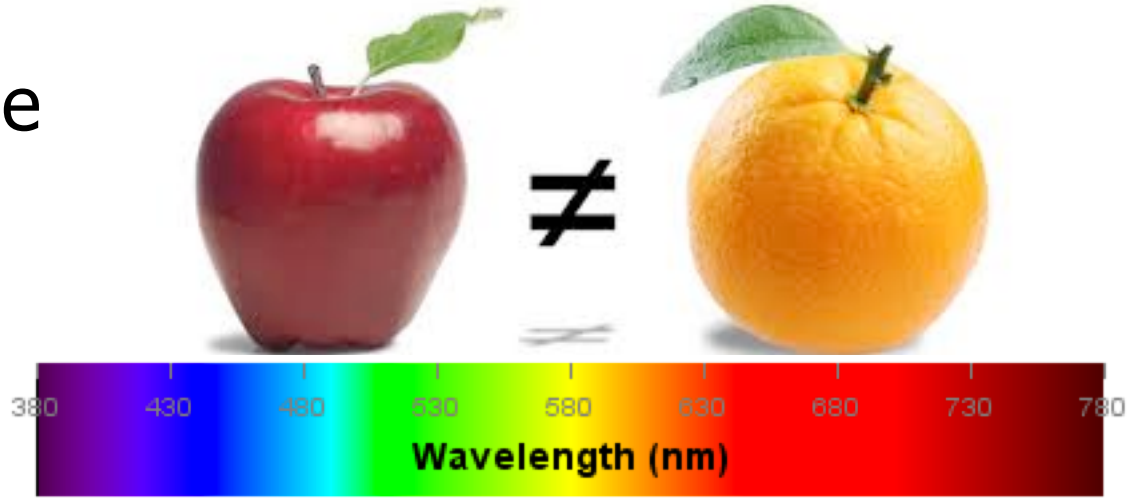
Supervised ML / Classification

- Apple vs Orange

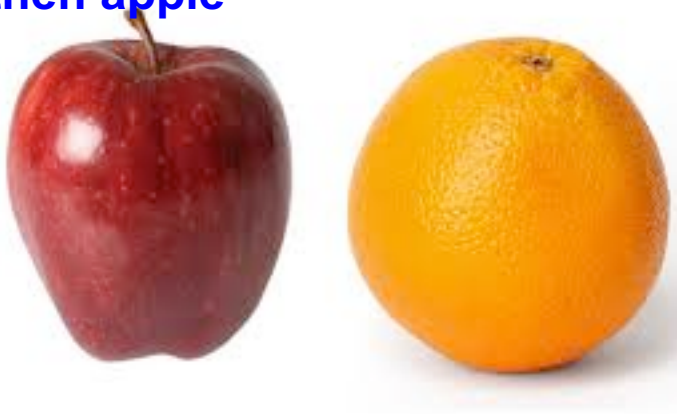


Supervised ML / Classification

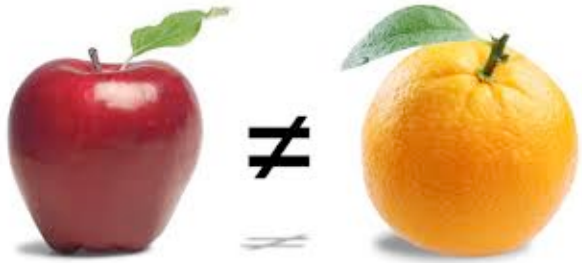
- Apple vs Orange



- **Feature/attribute: wavelength of light reflected from the object**
- **If wavelength>600 then apple otherwise orange**



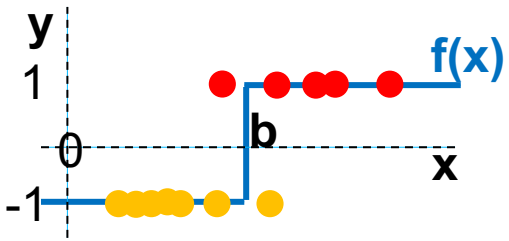
Supervised ML / Classification

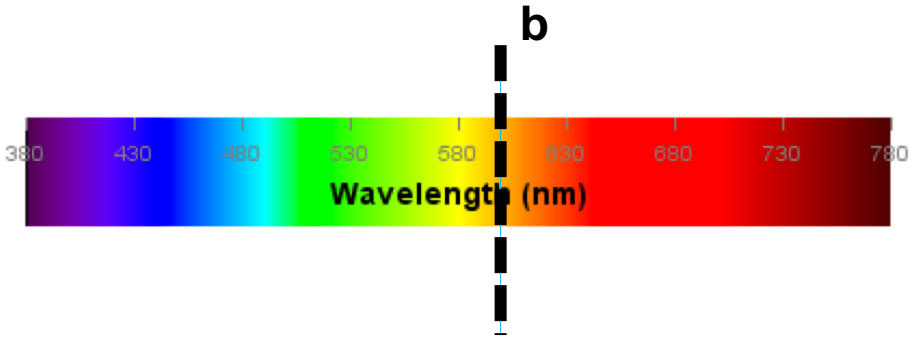


- Apple vs Orange

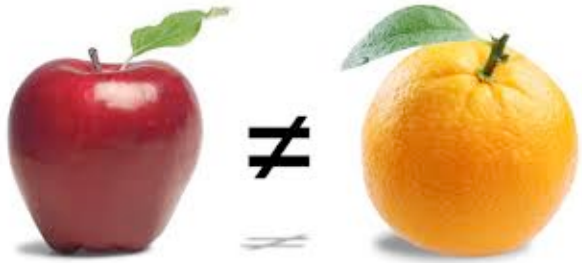
- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)

- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
  - **b is unknown**, need to be learned from examples





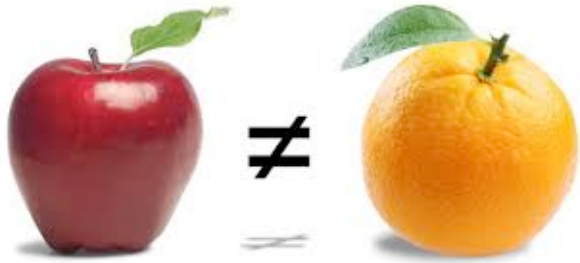
Supervised ML / Classification



- Apple vs Orange

- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)
- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
- Example of an algorithm:
  - `Set initial values of b (0, or random, or a guess)`
  - `Loop:`
    - `Take a sample` x_i `and predict` $f(x_i)=sign(x_i - b)$
    - `Compare true class` y_i `with predicted class` $f(x_i)$
    - `If prediction is right, go to next sample (i=i+1)`
    - `If prediction is wrong, update b`
      - `In a way that would make it more likely to get correct prediction for the current sample`
      - **if $y_i = 1$, $f(x_i)=-1$, what should we do?**
      - **if $y_i = -1$, $f(x_i)=1$, what should we do?**

Supervised ML / Classification

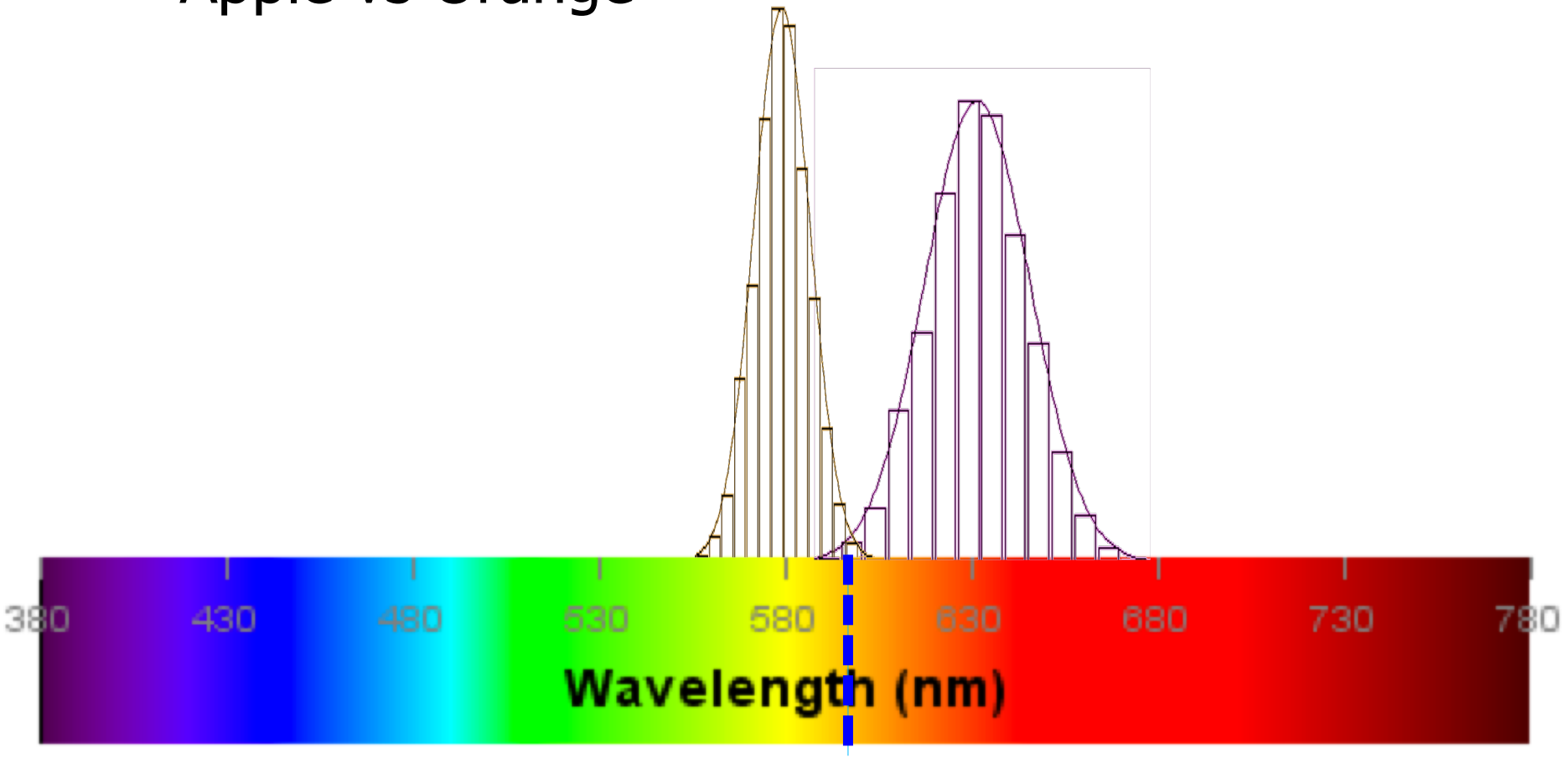


- Apple vs Orange

- Let:
  - **x** denote object's **color** (wavelength in nm)
  - **y** denote object's **class** (-1 = orange, 1 = apple)
- Prediction will be made by evaluating a function:
  - **f(x)=sign(x – b)** - it returns either -1 or 1 (or 0: tough to predict)
- Example of an algorithm:
  - `Set initial values of b (0, or random, or a guess)`
  - `Loop:`
    - `Present a sample` x_i `and predict` $f(x_i)=sign(x_i - b)$
    - `Compare true class` y_i `with predicted class` $f(x_i)$
    - `If prediction is right, go to next sample (i=i+1)`
    - `If prediction is wrong, update b`
      - `In a way that would make it more likely to get correct prediction for the current sample`
      - **if $y_i = 1$, $f(x_i)=-1$, what should we do? Decrease b to increase f**
      - **if $y_i = -1$, $f(x_i)=1$, what should we do? Increase b to decrease f**

Supervised ML / Classification

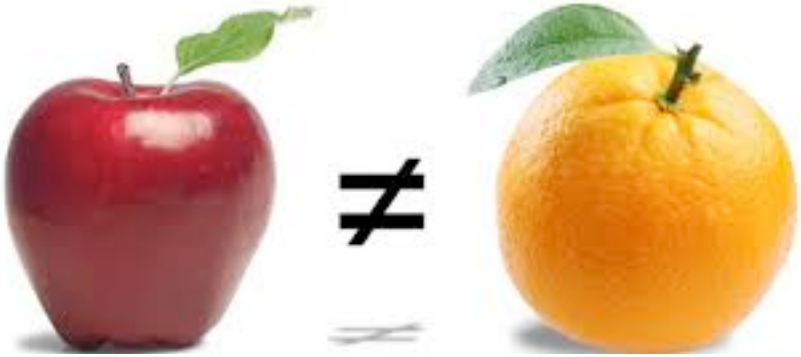
- Apple vs Orange



- **Feature A: object wavelength**
- **If A>600 apple**
  otherwise orange

Supervised ML / Classification

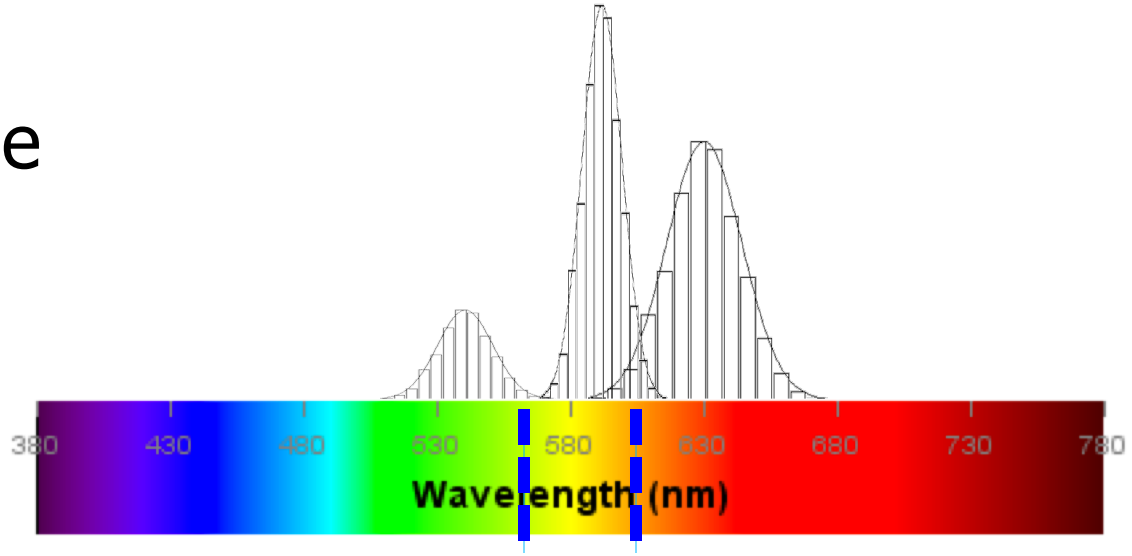
- Apple vs Orange



- **Feature A: object wavelength**
- **If A>600 apple**
  otherwise: if A<560 apple
  otherwise orange

Supervised ML / Classification

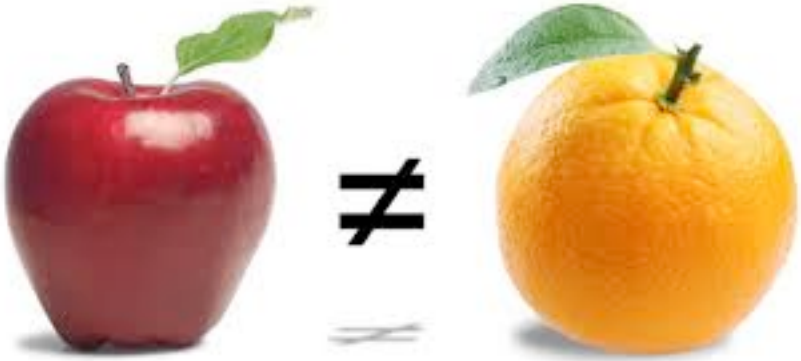
- Apple vs Orange



- **Feature A: object wavelength**
- **If A>600 apple**
  otherwise: if A<560 apple
  otherwise orange

Supervised ML / Classification

- Apple vs Orange



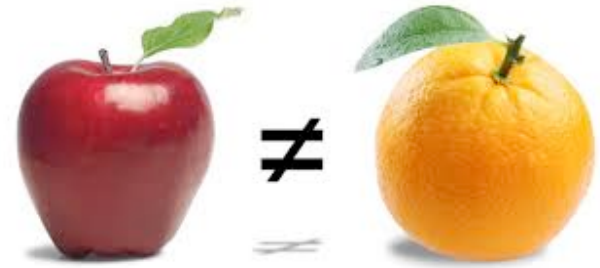
- **Feature A: object wavelength**
- **If A>600 apple**
  otherwise: if A<560 apple
  otherwise ???

WHAT SHOULD WE DO?

Wavelength alone is not enough

Any other features?

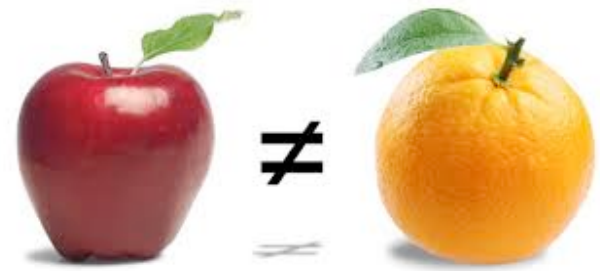
Supervised ML / Classification



- **Feature A: color**
  - **Apple: green, red, orange**
  - **Orange: orange**
- **Feature B: color variability**
  - **Apple: uniform or not**
  - **Orange: uniform**
- **Feature C: texture**
  - **Apple: smooth**
  - **Orange: rough**
- **Feature D: reflectance**
  - **Apple: reflects more light**
  - **Orange: reflects less light**
- **Feature E: shape**
  - **Apple: non-convex**
  - **Orange: convex (almost)**

- **We always arrange features as a vector,**
  e.g. for each fruit it's [A,B,C,D,E]

Supervised ML / Classification



- **Forget these:**
  - color
  - color variability
  - texture
  - reflectance
  - shape

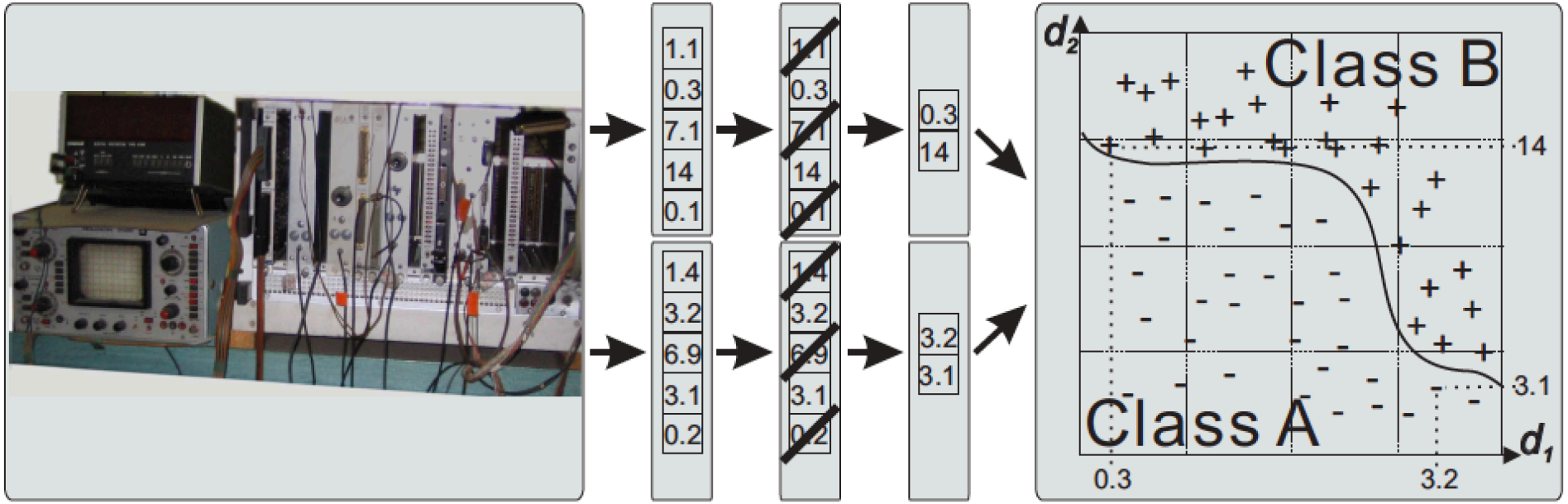
A fully accurate feature eliminates the need for machine learning

- **Do genome sequencing of the fruit!**

- **If you know the attribute that truly differentiates the classes**
- **And you can measure it**
- **Then there's nothing to learn!**
- **Classification methods are useful if the differences between classes are not known or not easily quantified**

Supervised ML / Classification

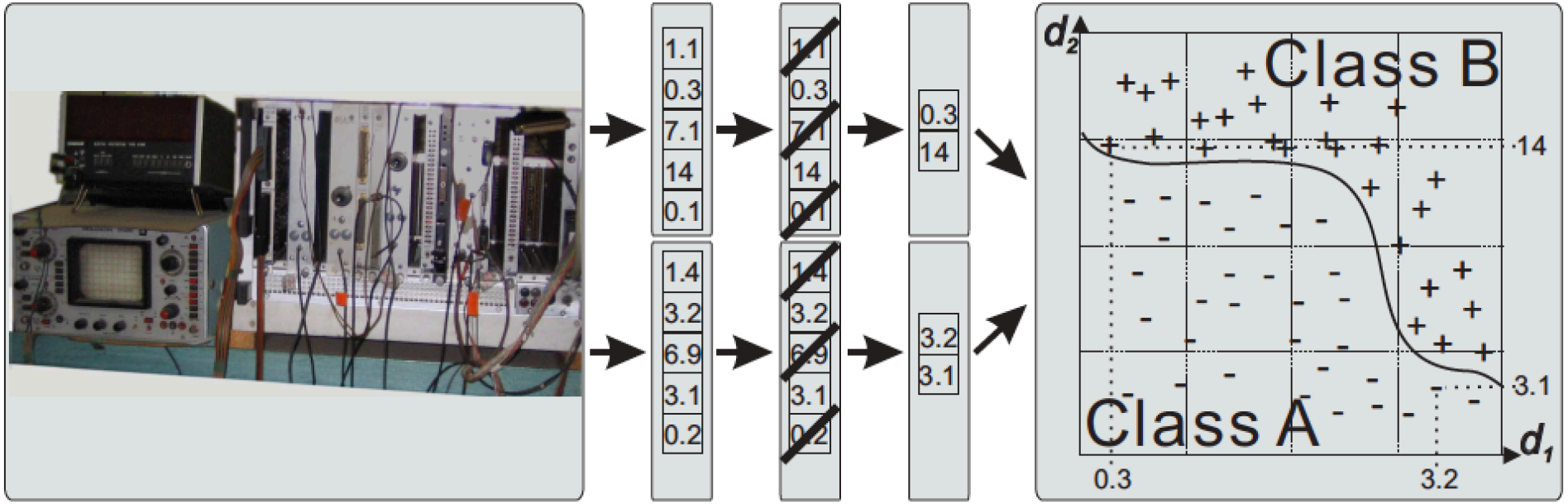
- Building a ***Classifier***



- Raw feature generation: measurable attributes
- Feature extraction: extract potentially relevant attributes
- Feature selection: select truly relevant features
- Classification: build a predictive model
- Validation: test the model

Supervised ML / Classification

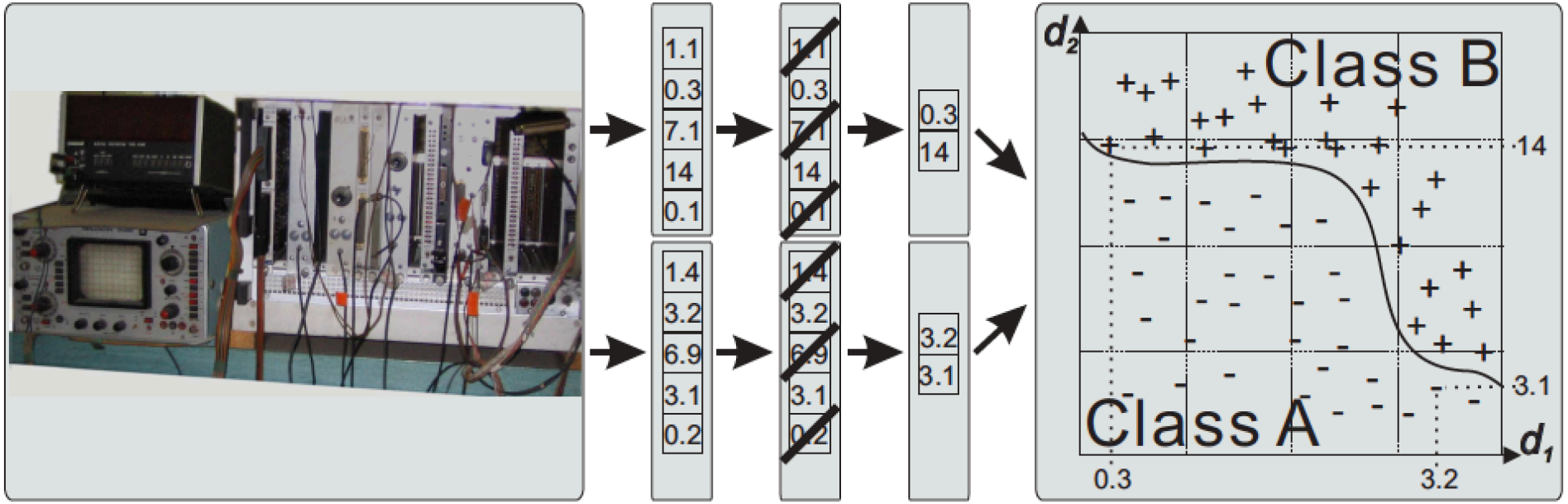
- Building a ***Classifier***



- Features are typically numerical (real numbers, or binary)
  - If not, some numerical encoding is typically used
- Feature space:
  - if we have 2 features for each object
  - then each object is a point in a 2D space
  - 1000 features => 1000 dimensions

Supervised ML / Classification

- Building a ***Classifier***



- F-dimensional feature space: R^F
- Often, the basics theory is done for problems with just 2 classes
- Then, building a classifier translates to:
  - assigning regions of feature space to class A and the rest to class B
  - finding a ***decision boundary*** in feature space between class A and class B

Linear Models

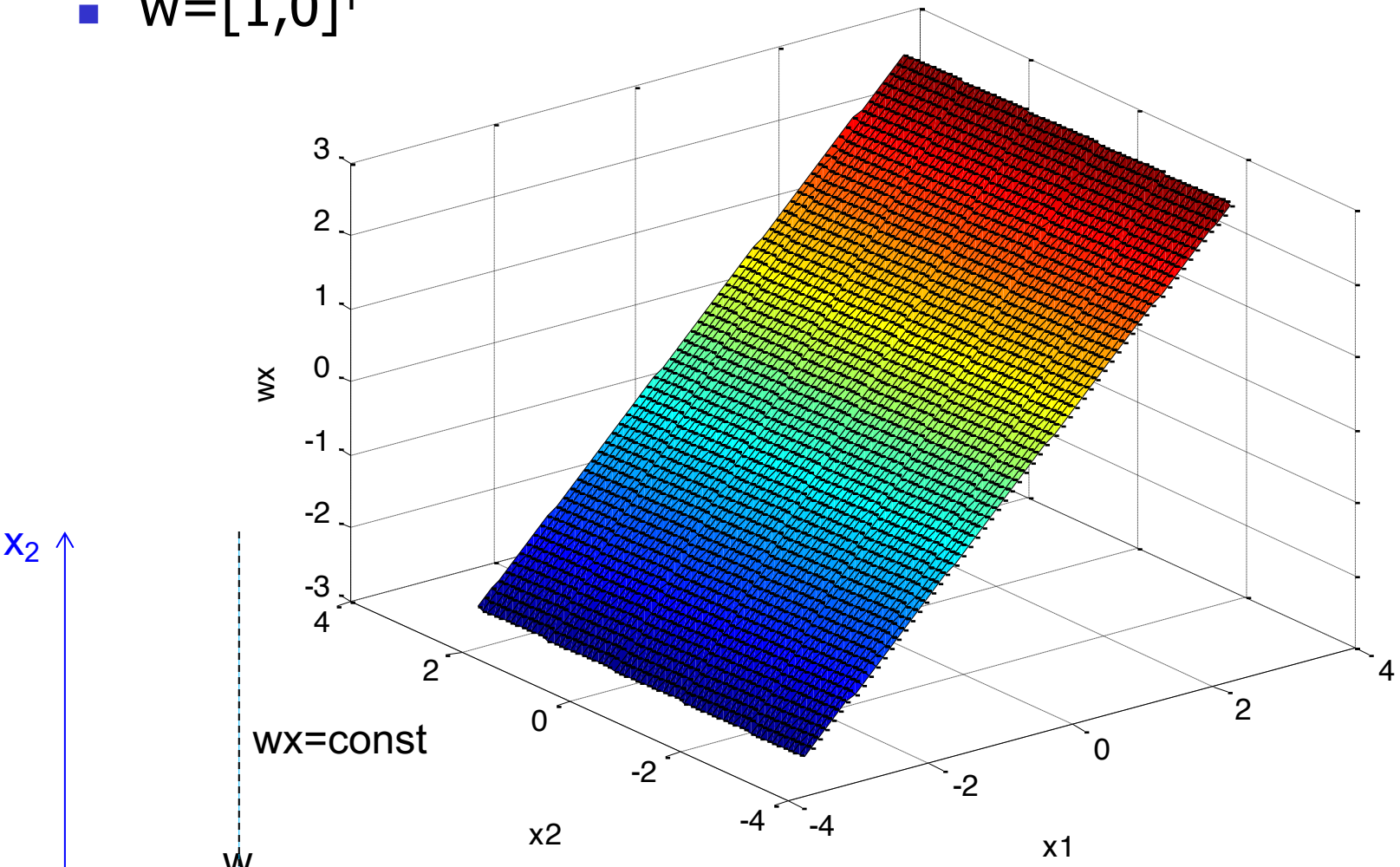
- A very simple but often powerful family of machine learning models (classifiers or regressors)

- "predicted y" $= \langle w,x \rangle + b$
  $= w^T x + b$
  $= w_1 x_1 + w_2 x_2 + b$

Linear Models

- $y=w^Tx=w_1x_1+w_2x_2$
- $w=[1,0]^T$

For this w, feature x_2 is irrelevant for predictions

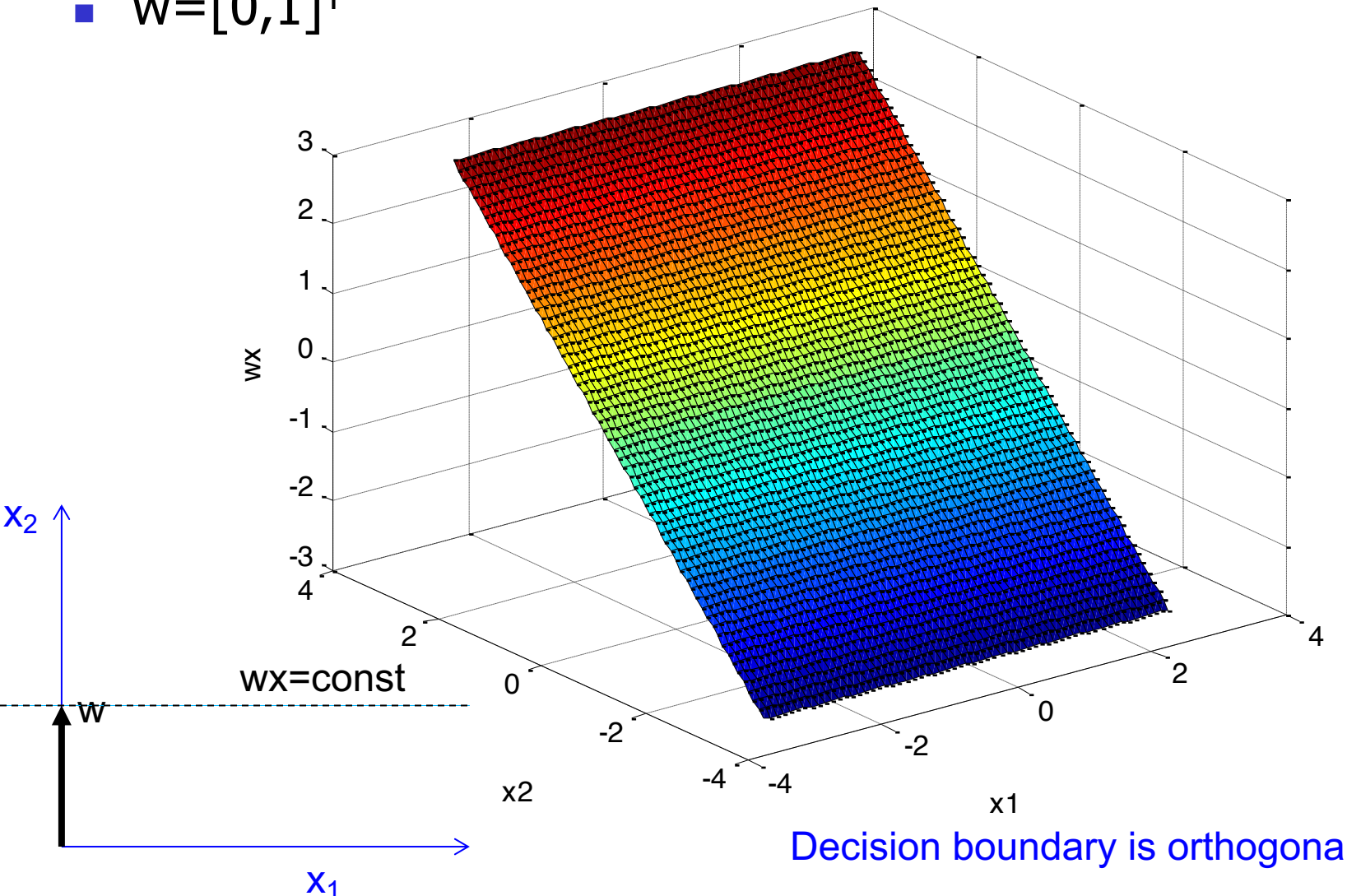


Decision boundary is orthogonal to w

Linear Models

- $y=w^Tx=w_1x_1+w_2x_2$
- $w=[0,1]^T$

Which features are important here?

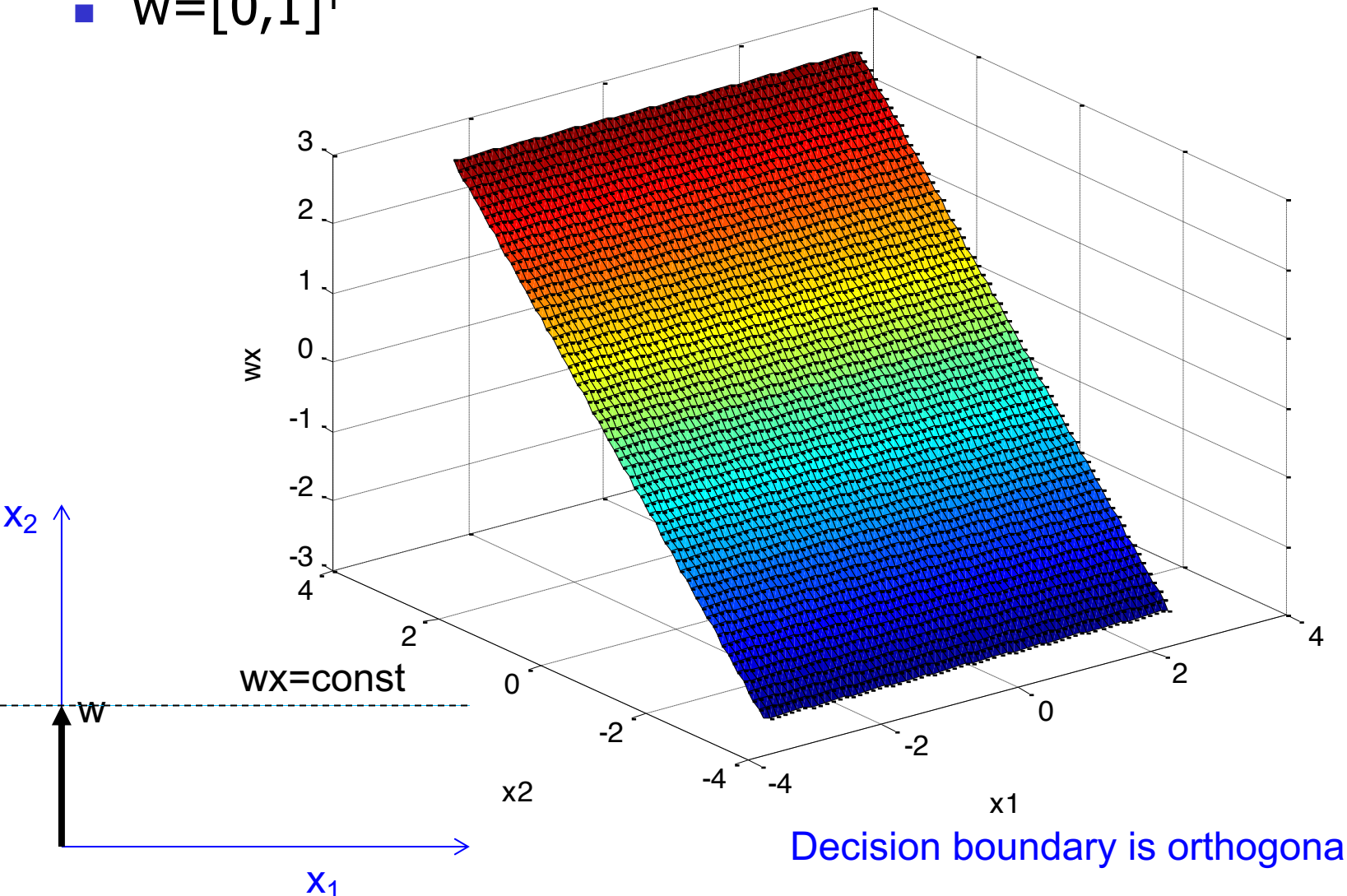


Decision boundary is orthogonal to w

Linear Models

- $y=w^Tx=w_1x_1+w_2x_2$
- $w=[0,1]^T$

For this w, feature x_1 is irrelevant for predictions

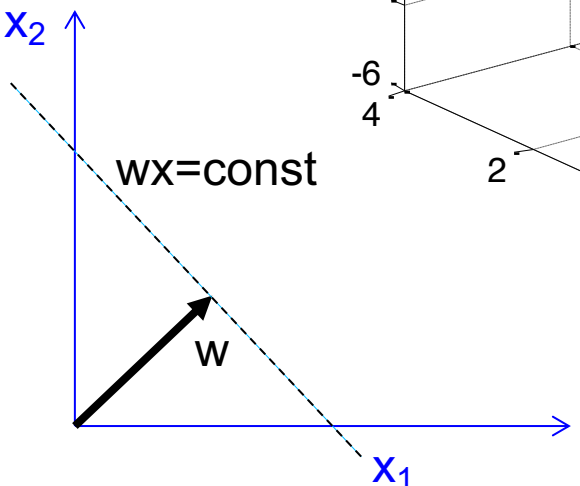


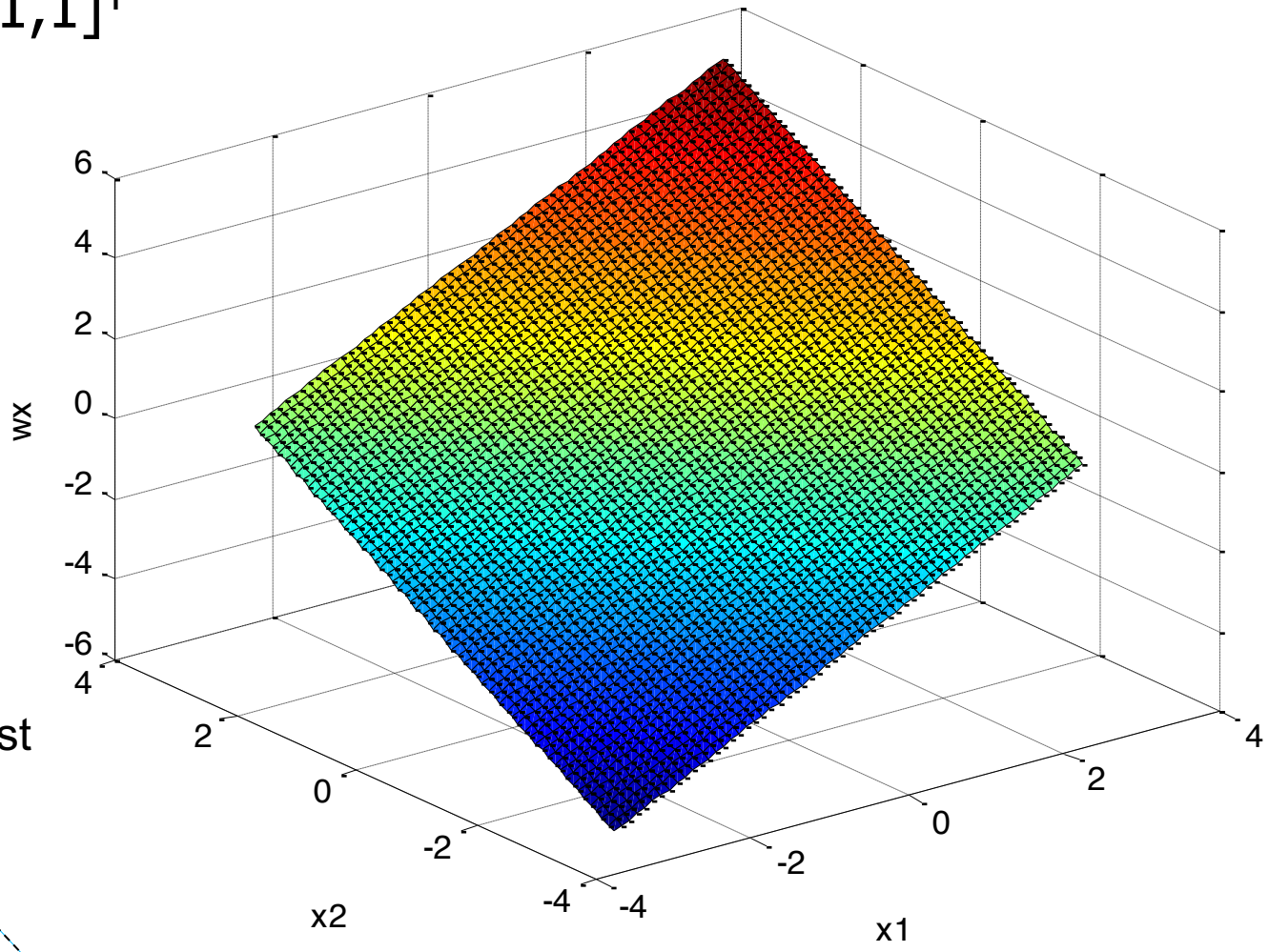
Decision boundary is orthogonal to w

Linear Models

- $y=w^Tx=w_1x_1+w_2x_2$
- $w=[1,1]^T$

For this w, both features are equally relevant for predictions





Decision boundary is orthogonal to w

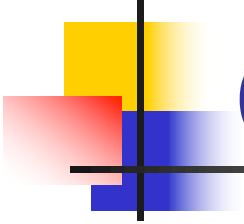
Linear Models - Regression

- 1D Example:
  - three samples z=(x,y):
    - $z_1=(0,1)$, $z_2=(2,3)$, $z_3=(-1,0)$
- MSE = $[(y_1 - (wx_1 + b))^2$
  $+ (y_2 - (wx_2 + b))^2$
  $+(y_3 - (wx_3 + b))^2] / 3$
- MSE = $[(1 - (w*0 + b))^2$
  $+ (3 - (w*2 + b))^2$
  $+(0 - (w*(-1) + b))^2] / 3$

- How to find w, b that leads to lowest MSE?
  - We need to talk about optimization methods
    - Plug in known values of features (x) and targets (y)
    - Treat MSE as a function MSE(w,b) - that is, not a function of x anymore!
    - Find w,b that leads to minimum of the function MSE(w,b)

Optimization

- Supervised machine learning
  - We have training data (x_i,y_i), *i from 1 to m*
  - We want to find function h(x) such that $h(x_i)$ is close to y_i
  - A machine learning solution:
    1. Pick an architecture of the function h(x,w), with input x, trainable parameters w
       - e.g. $h(x)= \sum_j w_j x_j +b$
       - e.g. h(x)="big neural network"
    2. Pick an objective function L(h(x,w), y) that measures how good h() is, for given (x,y), given w
       - e.g. L is mean squared error
    3. Pick a method for finding "good" values of parameters w
       - How to solve: $\min_w \sum_i L(h(x_i,w),y_i)$
       - In general, how to find $w_{opt} = \text{argmin}_w f(w)$



Optimization

How fast is finding a global minimum x* of a function f(x)?

The "x" here is e.g. [w, b].
It's not feature values!

We have a minimum-finding method M

We have a class of optimization problems F

E.g. $\min_x$ *f*(x) s.t. *x in* $[0,1]^n$ *where f is a polynomial*

We define the performance of M on class F by measuring the performance of M on the worst (hardest) problem from F

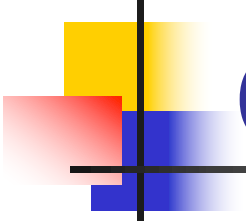
Which problem from F is worst? It may depend on M

Method M operates using an oracle O

For a possible solution vector x, oracle returns one number: f(x)

Method M goes through a sequence of solutions {x},

for each x we measure error ε (e.g. ε=|f(x) - f(minimum)|)



Optimization

Two approaches to finding minimum

1. **Analytical - Paper and pencil**
    1. **If our function is simple enough, we may be able to find the formula for the minimum**

2. **Numerical**
    1. **We try many x (chosen in some smart way, possibly)**
    2. **For each x, we have a way to obtain f(x)**
        - We have an ***oracle*** (a coded procedure) that gives us f(x) for any x
        - We may have an oracle that gives us also some other information, such as derivative of f(x) at x
    3. **At some point we decide to stop,**
    4. **We return smallest f(x) we encountered**

Optimization

Any method based on an oracle operates by

performing a series of queries to oracle O,

acting based on information I obtained from the queries

Any oracle-based optimization method M is an iterative scheme:

0. Create initial solution x_0, set k=0, I_{-1}=empty
1. Call oracle O at x_k
2. Update accumulated information $I_k=I_{k-1}$ + $(x_k,O(x_k))$
3. Apply internal rules of method M to I_k, to generate x_{k+1}
4. Calculate error ε, check stopping criterion based on ε

   If criterion met, exit with the solution,

   if no, k=k+1, go back to step 1.

Two measures of computational complexity of M on problem P:

Analytical complexity A(M,ε):
number of calls to oracle O required to get error at most ε

Arithmetic complexity:
number of all operations (typically proportional to analytical complexity)

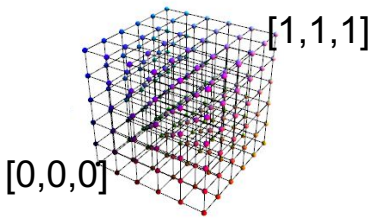
Optimization

Let's try a simple method for solving any problem of the form

Minimize $f(x)$
subject to: $x \in B_n$

$f: R^n \rightarrow R$ that is, x: n-dimensional vector $[x^{(1)}, x^{(2)}, x^{(3)}, \ldots x^{(n)}]$

$B_n = \{x \in R^n: 0 <= x^{(i)} <= 1, i = 1 \ldots n\}$, that is, n-dimensional box [0,0,0,..,0] – [1,1,1,…,1]



Optimization

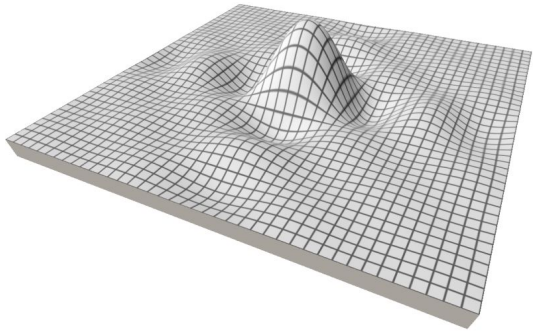
Simple method M_{grid}:

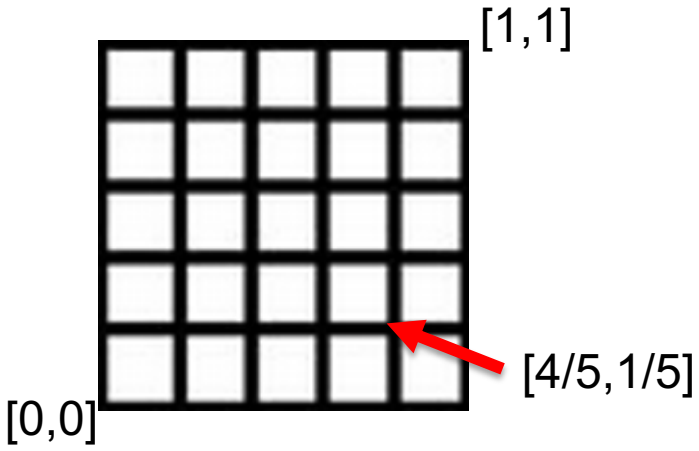
Generate $(p+1)^n$ points on a n-dimensional grid in [0,1]

$x(i_1,i_2,\ldots,i_n)=(i_1/p, i_2/p, \ldots, i_n/p), \quad i_k \in \{0,1,\ldots,p\}$

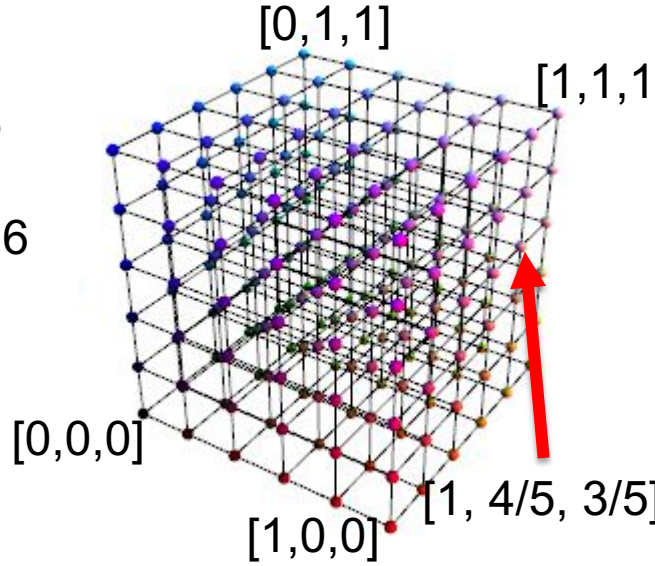
Call oracle $(p+1)^n$ times,
get values of f(x) for each grid point x

Return the grid point x' with lowest f





- p=5,
- If n=2, we need $6^2=36$ points in the grid
- If n=3, we need $6^3=216$ points in the grid
- each cell has side length of 1/5



Optimization

Problem: Find minimum f(x) s.t. $x \in B_n$, assuming f is Lipschitz contin.

$| f(x) - f(y) | <= L || x-y ||_\infty$ for all $x,y \in B_n$, where: $|| x ||_\infty = \max_{1<=i<=n} |x^{(i)}|$

Simple method M_{grid}:

Generate $(p+1)^n$ point on a n-dimensional grid in [0,1]

$x(i_1,i_2,\ldots,i_n)=(i_1/p, i_2/p, \ldots, i_n/p)$, $i_k \in \{0,1,\ldots,p\}$

Call oracle $(p+1)^n$ times, get values of f for each grid point

Return the point x' with lowest f

What is the highest error we can get with method M_{grid}?

Let $f^*=f(x^*)$ be the value of f in the real global minimum x^*

$e(x') = | f^* - f(x') |$

Can we say something about error e(x')?

No, the function can have any value outside of grid points.

We could quantify e(x') if we added some constraint on the variability of f()

Optimization

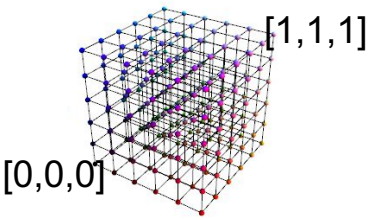
Let's try a simple method for solving any problem of the form

Minimize f(x)
subject to: $x \in B_n$

f: $R^n \rightarrow R$ that is, x: n-dimensional vector $[x^{(1)}, x^{(2)}, x^{(3)}, \ldots x^{(n)}]$

$B_n = \{x \in R^n: 0 <= x^{(i)} <= 1, i=1 \ldots n\}$, that is, n-dimensional box [0,0,0,..,0] – [1,1,1,…,1]

Let's define a norm (vector length) in R^n:



$|| x ||_\infty = \max_{1<=i<=n} |x^{(i)}|$

$x^{(i)}$ is the i-th coordinate of vector x

In this L_∞ norm, length of vector x is max. magnitude of its coordinates

E.g. if x=[0.1, 0.3, -0.7, 0.5] then $|| x ||_\infty = 0.7$

Function p(x) is a norm if for any vectors x, y and any real number c:

1. $p(cx) = |c| \, p(x)$
2. $p(x + y) \leq p(x) + p(y)$
3. If p(x) = 0 then x is the zero vector [0,0,…0]

$p(x) = \max_{1<=i<=n} |x^{(i)}|$ meets these conditions: it's a proper norm

Optimization

Let's try a simple method for solving any problem of the form

Minimize f(x)
subject to: $x \in B_n$

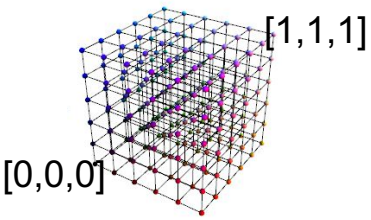
f: $R^n \rightarrow R$ that is, x: n-dimensional vector $[x^{(1)}, x^{(2)}, x^{(3)}, \ldots x^{(n)}]$

$B_n = \{x \in R^n: 0 <= x^{(i)} <= 1, i=1 \ldots n\}$, that is, n-dimensional box [0,0,0,..,0] – [1,1,1,…,1]

Let's define a norm (vector length) in R^n:

$|| x ||_\infty = \max_{1<=i<=n} |x^{(i)}|$

$x^{(i)}$ is the i-th coordinate of vector x



Let's put a mild condition on the variability of function f:

f(x) is *Lipschitz continuous* on B_n with Lipschitz constant L if:

$$| f(x) - f(y) | <= L || x-y ||_\infty \text{ for all } x,y \in B_n$$

This makes optimization problem slightly easier:
f(x) doesn't change drastically if we change x a bit

**When x changes by δ,
then value of the function f changes by
at most Lδ**

Optimization

Problem: Find minimum f(x) s.t. $x \in B_n$, assuming f is Lipschitz contin.

$| f(x) - f(y) | <= L || x-y ||_\infty$ for all $x,y \in B_n$, where: $|| x ||_\infty = \max_{1<=i<=n} |x^{(i)}|$

What is the highest error we can get with method M_{grid}?

Let $f^*=f(x^*)$ be the value of f in the real global minimum x^*

$e(x') = | f^* - f(x') |$

We can prove that $e(x')<=L/2p$ (next slide)

For given ε, if we want certainty we get x' with error below ε,
$e(x') <= \varepsilon$, we need p that gives $\varepsilon=L/2p$,
i.e., $p>=\text{floor}(L/2\varepsilon)+1$

A(M,ε): number of calls to O required to get error at most ε

$A(M_{grid},\varepsilon) = (p+1)^n >= (\text{floor}(L/2\varepsilon) + 2)^n >= (L/\varepsilon)^n$