Supervised Learning: Linear Models

P J Narayanan

Outline

- Model fitting: Prediction of *y* from *x*
  - Linear regression: Model is linear
  - Gradient descent method to find best fitting line
- Logistic regression: when *y* is binary
  - Function model is different
  - Gradient descent works
  - Used for classification. Gives probabilities
- Linear classifiers: Find line to separate classes
  - Linear separability

The machine learning framework

$$y = f(\mathbf{x})$$

output — prediction function — feature or representation

- **Training:** given a *training set* of labeled examples $\{(\mathbf{x}_1,y_1), \ldots, (\mathbf{x}_N,y_N)\}$, estimate the prediction function f by minimizing the prediction error.
- **Testing:** apply f to a never before seen *test example* $\mathbf{x}$ and output the predicted value $y = f(\mathbf{x})$

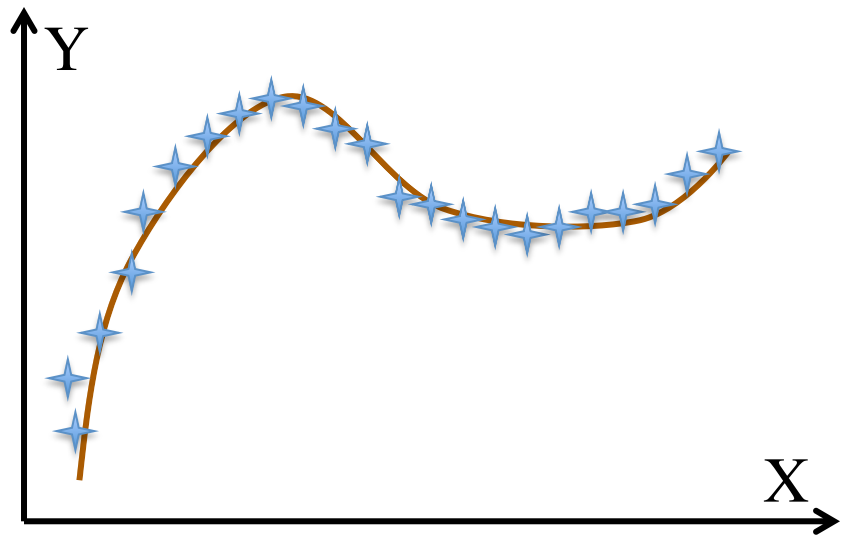
Slide credit: L. Lazebnik

Fitting Functions to Data

Why?

- Discover (hidden) structure in the data, given samples
- A functional form is a compact representation usable for interpolation and extrapolation
- Forms: Lines, Polynomials, Gaussian, etc.

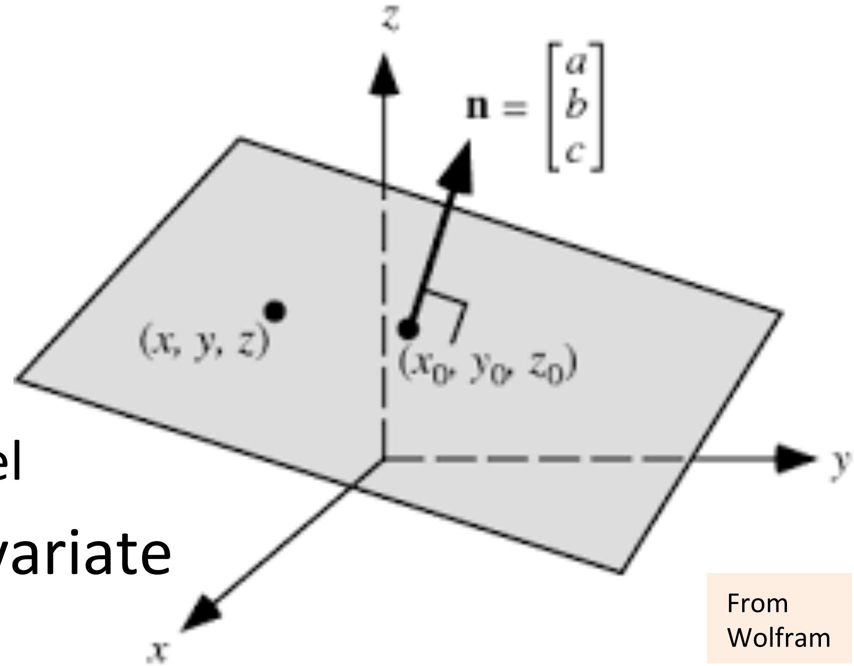
Called ***regression*** in general



$$y = f(x)$$

Scalar or vector valued x, y
Multivariate when x is a vector $\boldsymbol{x}$

Linear Model



From Wolfram

- Linear when f is a line.
  In general, a hyperplane
  - $y = a\,x + b$
    - a, b: parameters of the model
  - $y = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} = \boldsymbol{w} \bullet \boldsymbol{x}$ when multivariate
    - $\boldsymbol{x} = [1\ \ x_1\ \ x_2\ \ \ldots.\ \ x_d]^{\mathrm{T}}$
    - Vector $\boldsymbol{w}$ represents the parameters of the model
  - Line: Only 2 parameters in 2D.
    d parameters in a *d*-dimensional space

$$y = f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$$

Notations

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ .. \\ x_d \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ .. \\ w_d \end{bmatrix}$$

$$\mathbf{w} \cdot \mathbf{x} = \mathbf{w}^T \mathbf{x} = w_0.1 + w_1 x_1 + w_2 x_1 + \cdots + w_d x_d$$

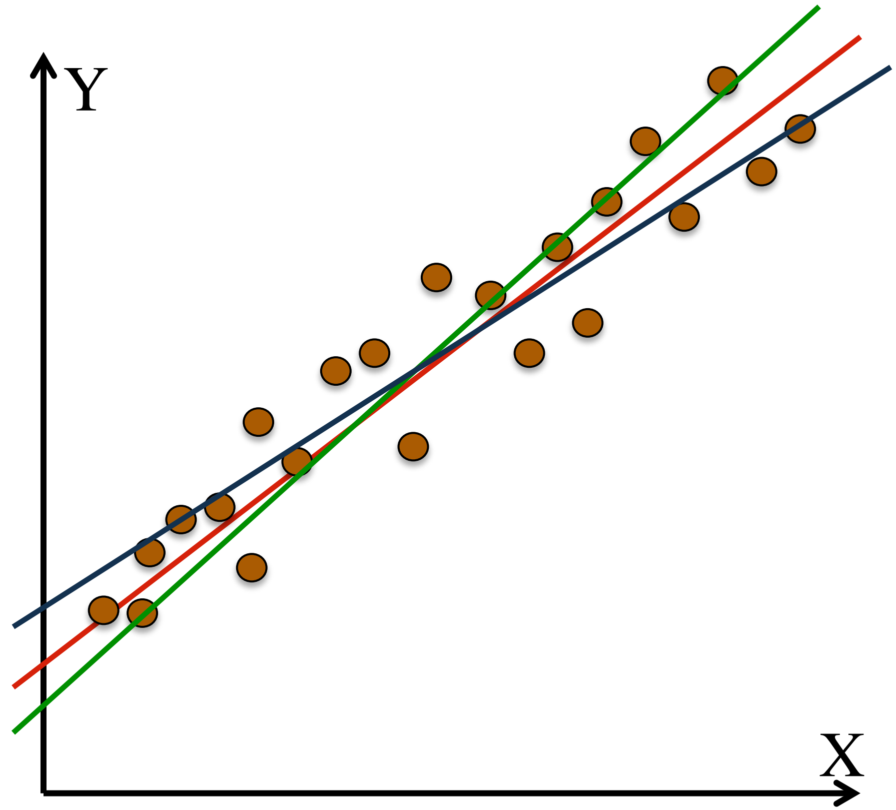
IIIT Hyderabad

The Problem

- Find $\boldsymbol{w}$ given examples $(\boldsymbol{x}_i, y_i), i = 1, 2, \ldots, m$
- Supervised situation: output label is available for a number m of training input samples
- Objective: predict y values for input values $\boldsymbol{x}$ that are not seen before
  - Called *generalization* in Machine Learning
- How do we find $\boldsymbol{w}$? Gradient descent!

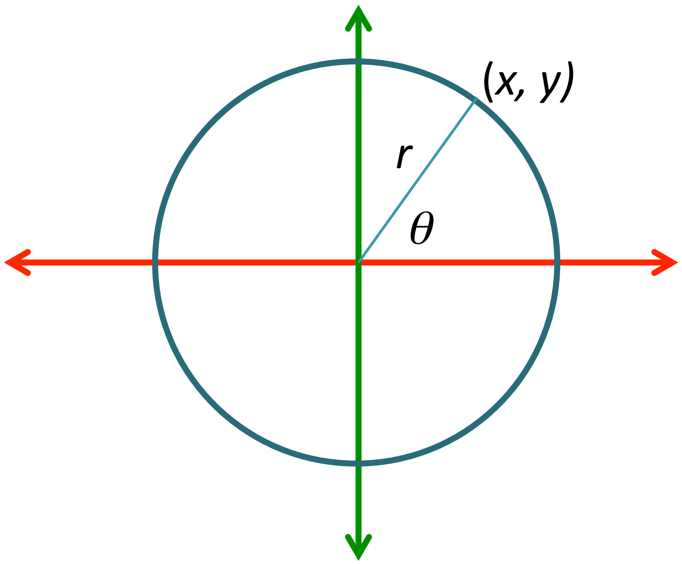
Typical Scenario

- Data points of x against y appear distributed
- Ideal: All points on the line if model is truly linear
  - Measurement errors and "noise" create deviations
- Several ways to find the best line
  - Analytical, Least Squared Error, Gradient Descent



Larger Issues

- What's great about linear?
  - Simple
- But the world is not linear
- But many can be converted to!
  - Circular to linear
  - ExOR to linear
  - Pendulum: T^2 to L is linear



$$x = r\cos\theta, y = r\sin\theta$$

Circle is $r = k$ in the r-θ space

$a\,r + b\,\theta + c = 0$ is a weird shape!

Questions?

Iterative Procedure

- Start with a guess for model parameters ***w***
  - Adjust till it fits well

$$y = f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$$

- Prediction for a given ***x***: $f_w(x) = w^T x$

$$f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$$

- Consider the j^{th} training sample (x^j, y^j)

$$f_{\mathbf{w}}(\mathbf{x}^j) = \mathbf{w}^T\mathbf{x}^j$$

- Predicted value:
  Observed value: y^j

$$y^j \neq f_{\mathbf{w}}(\mathbf{x}^j)$$

- Typically **not** equal! The difference guides change in ***w***

IIIT Hyderabad

Loss Function

- **Error** or **loss**: *D().* How far is the prediction from the observed value?
  - Different loss functions used

$$D(f_{\mathbf{w}}(\mathbf{x}^j), y^j)$$

- Strategy: Bring the *predicted* value closer to the *observed* value by adjusting ***w***

$$\sum_j D(f_{\mathbf{w}}(\mathbf{x}^j), y^j)$$

- How do we adjust ***w***? Gradient descent

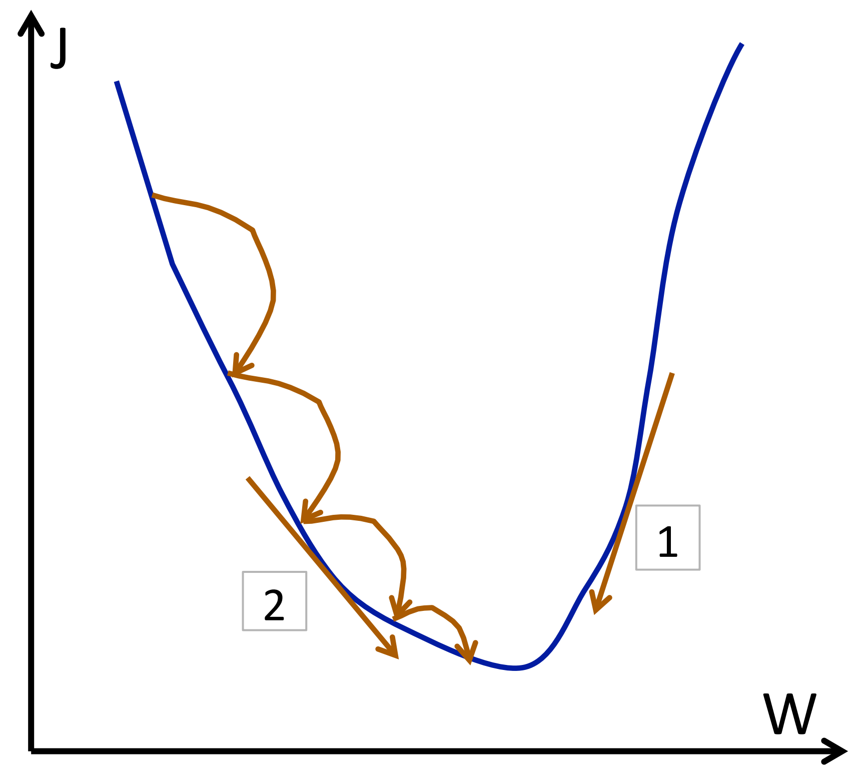
Gradient Descent

- Minimum of the function lies in the ***opposite*** direction of the gradient

  1. Positive gradient: function will increase if we go forward
  2. Negative gradient: minimum lies ahead

- Take a step against gradient:

$$\mathbf{w}' = \mathbf{w} - \eta \nabla J(\mathbf{w})$$



Fine Points

When do we stop the iterations?

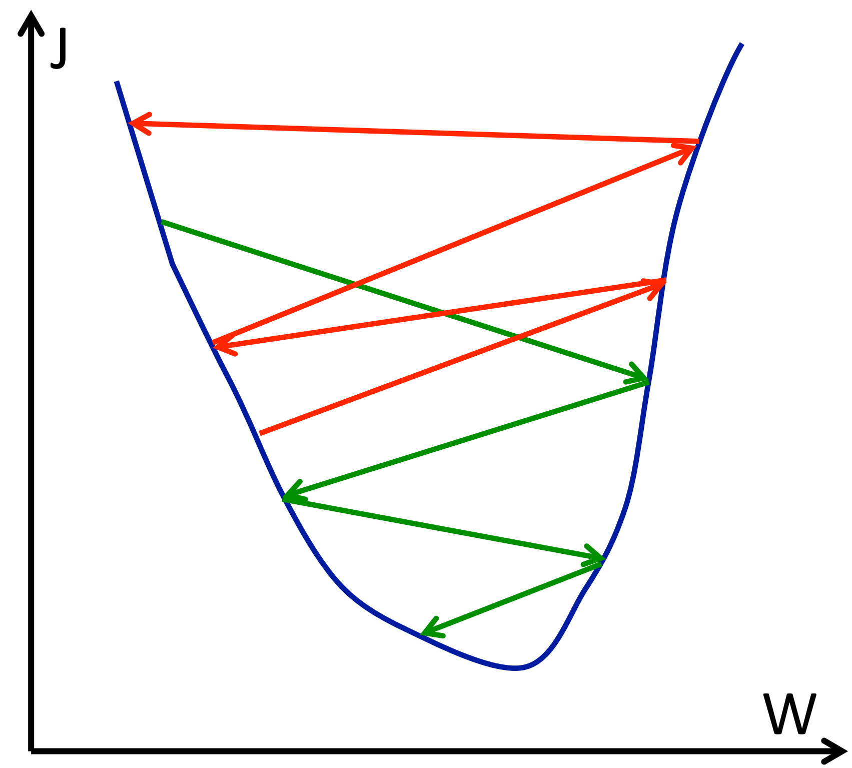
- When the gradient value is too low ($< \varepsilon$)
  - Future changes will be low!
- When the change in objective function is too small
  - We are close already

How about the step size?

- Ensure smooth convergence

Gradient Descent

- Converges to the minimum when function is convex
  - Converges to a local minimum otherwise
- Learning rate is critical
  - Start high to make rapid strides
  - Reduce with time for smoother convergence



IIIT Hyderabad

Questions?

Minimize Loss Function

- A loss function: L2 or Euclidean distance

$$J(\mathbf{w}) = \frac{1}{2}||(f_{\mathbf{w}}(\mathbf{x}^j) - y^j)||_2$$

$$= \frac{1}{2}(f_{\mathbf{w}}(\mathbf{x}^j) - y^j)^2$$

- *J(**w**)* is the function to be minimized with respect to ***w***

- Cost due to a sample & Cost for all of them

$$= \frac{1}{2}\sum_{j=1}^{m}(f_{\mathbf{w}}(\mathbf{x}^j) - y^j)^2$$

LMS Update Rule

- Gradient descent follows this equation

$$\mathbf{w}' = \mathbf{w} - \eta \nabla J(\mathbf{w})$$

- What is the gradient of *J(**w**)*?

$$\frac{1}{2}\nabla_w (f_{\mathbf{w}}(\mathbf{x}^j) - y^j)^2 = (f_{\mathbf{w}}(\mathbf{x}^j) - y^j)\mathbf{x}^j$$

- This is a vector of *d* dimensions, like ***x, w***

Batch and Stochastic GD

- **Batch** GD: Go through all input samples and update at end
  - Uses “true” gradient
  - Expensive computationally

$$\mathbf{w}' = \mathbf{w} - \eta \sum_j (f_{\mathbf{w}}(\mathbf{x}^j) - y^j)\mathbf{x}^j$$

- **Stochastic** GD: Update weights after each sample
  - More “noisy”, but faster
  - Noise may actually help!

$$\mathbf{w}' = \mathbf{w} - \eta (f_{\mathbf{w}}(\mathbf{x}^j) - y^j)\mathbf{x}^j$$

- Mini-batch: Update after a small number of samples

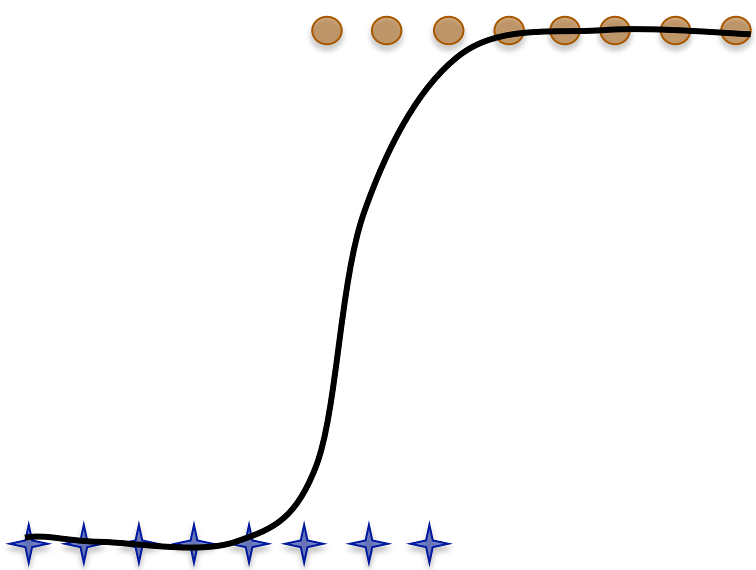
Summary

- Linear regression fits a line or a hyperplane to a set of points
  - Models the behaviour of the (continuous) dependent variable against the independent one
- Several methods exist to perform line fitting
  - Iterative methods work well in several situations
  - Machine learning uses lots of data. Incremental methods are more suitable
- Gradient Descent is a versatile method!

Questions?

Categorical/Binary Functions

- When *y* value is a category label (with no clear relative ordering)
- Easy case: *y* is binary. **Yes**/**No**. **True**/**False**. **1**/**0**.
- Can be interpreted as the probability of an outcome being true, given an event

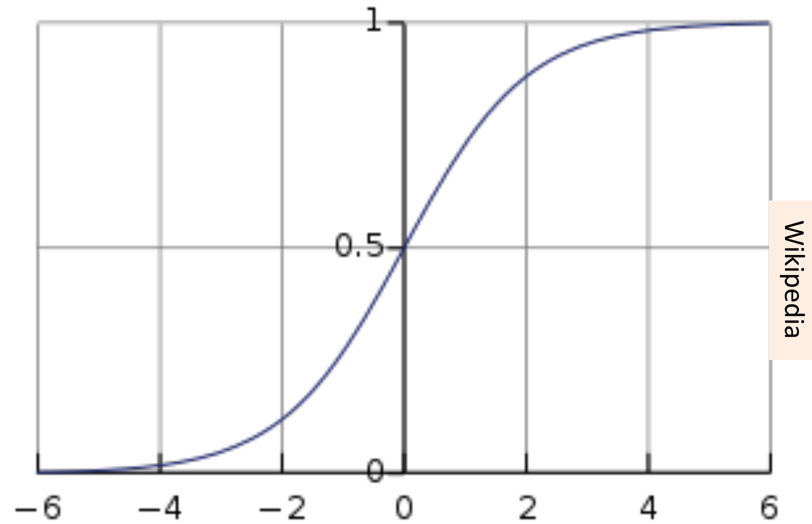


Logistic Function

- The **logistic** function comes handy

$$f(z) = \frac{1}{1 + e^{-z}}$$

- Goes from *0* to *1* as *z* goes from $-\infty$ to $+\infty$



- Can be interpreted as a probability

P(success) = f(effort)

Use *kz* to make it steeper.

$(z - z_0)$ to shift transition point

Logistic Function

- The **logistic** function
  - Probability of an outcome

$$f(z) = \frac{1}{1+e^{-z}}$$

- Has an interesting derivative form

$$f'(z) = f(z)(1 - f(z))$$

- Connect with linear: $z = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$

$$f_{\mathbf{w}}(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$$

- Generalized Linear Model with parameters ***w***

IIIT Hyderabad

Logistic Regression

- Fit a logistic function to the outcome *y* with respect to *x*.

$$\mathbf{w}' = \mathbf{w} - \eta(f_{\mathbf{w}}(\mathbf{x}^j) - y^j)\mathbf{x}^j$$

- Gradient Descent can be used to minimize the loss function

The Maths is involved and uses Maximum Likelihood estimate, etc.

- Results in the exact same update rule as linear regression though f is non-linear

Can use Batch or Stochastic Gradient Descent methods

Binary Classification

- Logistic regression gives probability of outcome
- Convert to a ***classifier*** with output **True** or **False**
- Classification rule:

$$\text{if } f(\mathbf{w}^T\mathbf{x}) > 0.5 \quad \textbf{True.} \quad \text{Else} \quad \textbf{False}$$

- Can be extended to multiple classes also

Summary

- Logistic function can map inputs to the probability of a categorical output
  - Can be used as a classifier for **Yes/No** questions

- Another form of $f(\mathbf{x})$. Another form of loss function

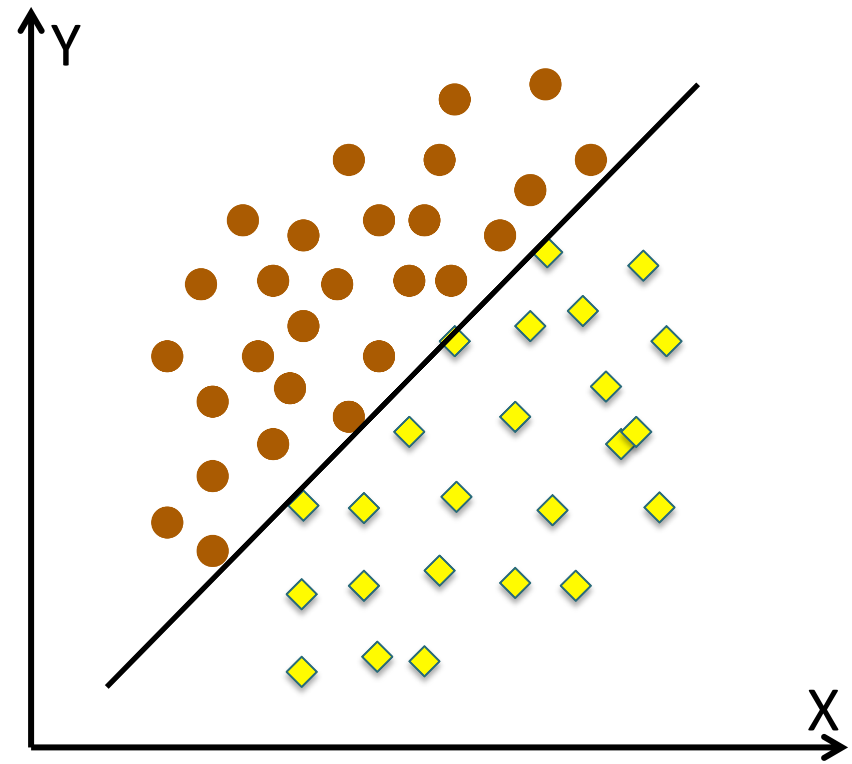
- Gradient Descent works well for this also.
  All one needs is a gradient

- Differentiability of the loss function is important!

Questions?

Linear Classifier

- Logistic regression fits a function to ***y*** (outcome) of the independent variable ***x***
- Classification can be done by partitioning the space among the classes
- Liner classifiers have linear partition or decision boundaries



Decision Boundary

- Decision boundary: hyperplane

$$\mathbf{w}^T\mathbf{x} = 0$$

- Class 1 lies on the positive side and Class 0 on the negative side
  – *t=1* for Class 1 and *t= -1* for Class 0
  – Negate features of Class 0 by *(t**x**)*

- For each training sample $\mathbf{x}^j$, distance to line $\mathbf{w}^T(t\mathbf{x}^j) \geq 0$

$$J(\mathbf{w}) = \frac{1}{2}\sum_j (\mathbf{w}^T(t\mathbf{x}^j))^2$$

- Loss function $J(\mathbf{w})$

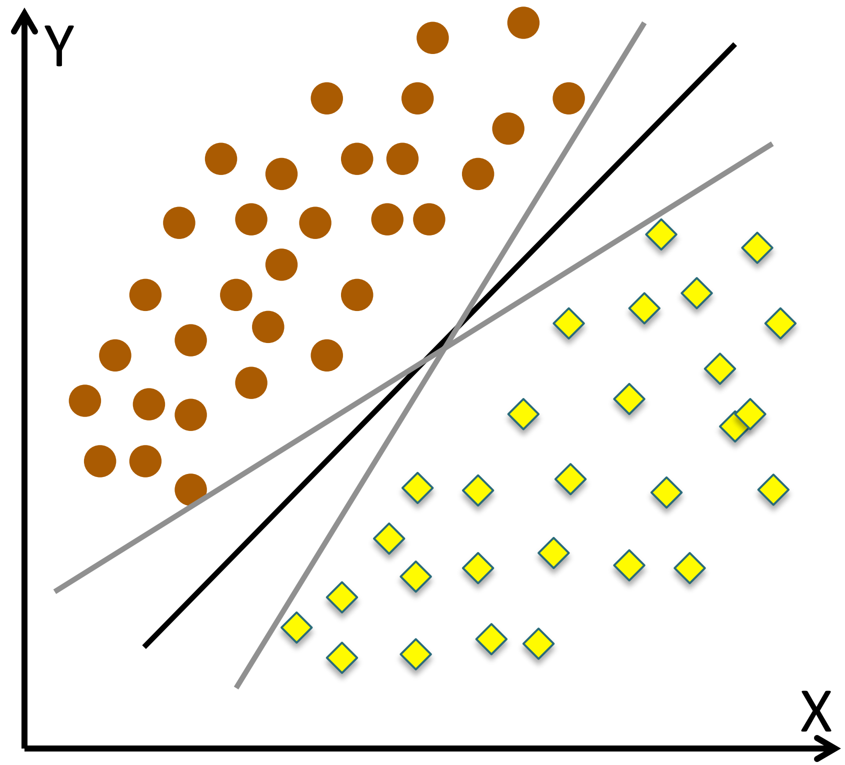
$$\nabla J(\mathbf{w}) = \sum_j [\mathbf{w}^T(t\mathbf{x}^j)]\ (t\mathbf{x}^j)$$

- Gradient Descent can work!

IIIT Hyderabad

Boundary with Margin

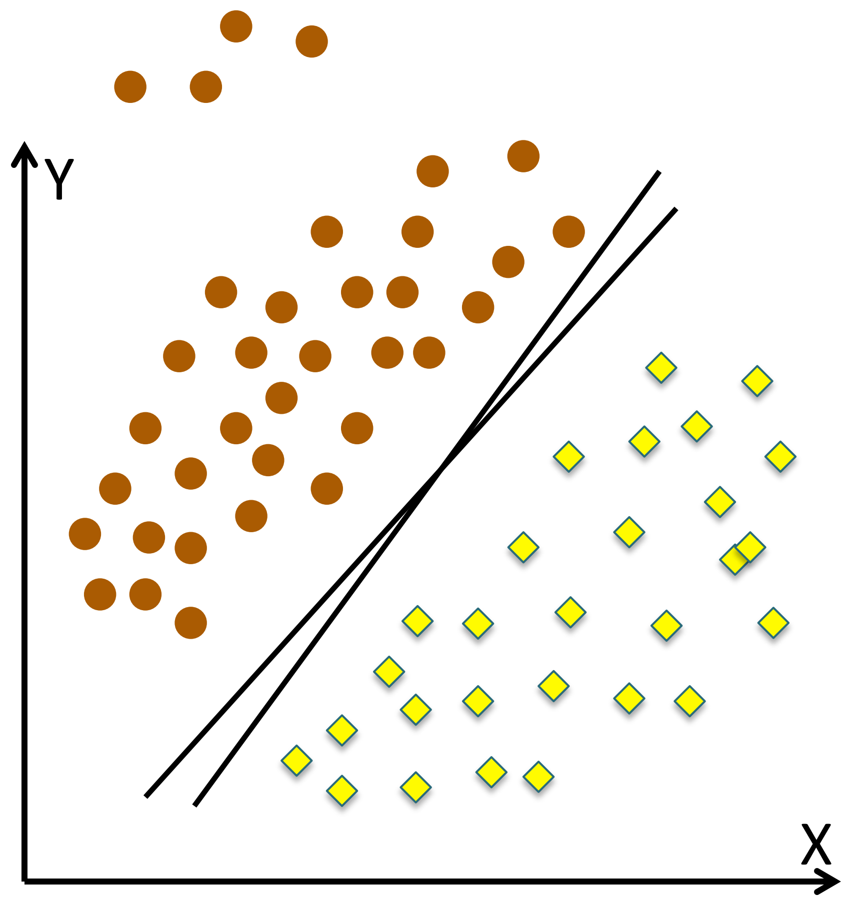
- Several lines separate the classes when there is a large gap
- Need to find the **middle line** with maximum distance to points of each class
- Require $\mathbf{w}^T (t\mathbf{x}^j) \geq b$, where b is a margin



Will see this later!

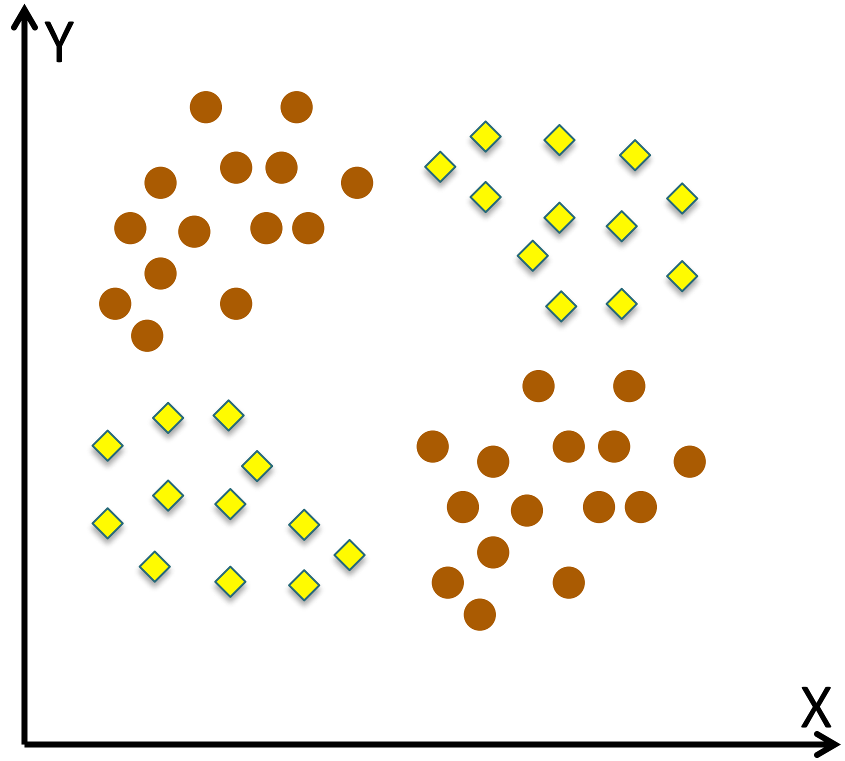
What's truly important?

- Every sample contributes by $\boldsymbol{w}^T\boldsymbol{x}$
  - Susceptible to samples that are far from the boundary
  - Called **outliers**
- Samples close to the boundary alone should matter in finding the boundary!
  - Will see it in SVM



Linear Separability

- No line can separate the classes cleanly
  - This is called the ExOR problem
- Some can be transformed to a linearly separable case
  - Map $\boldsymbol{x}$ to $\phi(\boldsymbol{x})$
  - Separable in $\phi(\boldsymbol{x})$



Summary & Questions

- Linear methods are simple and versatile
  - Several situations can be mapped to linear
- Other advanced methods are variations or extensions of simple linear methods
  - Support Vector Machines
  - Neural Networks including Deep ones

Thank You!