LONDON
METROPOLITAN
UNIVERSITY

Machine Learning
CS7052
Lecture 5, Gradient Descent

Dr. Elaheh Homayounvala

week 5

Outline of today's lecture

- Review last week
  - Linear Models, Regression and Classification
- Supervised learning
  - Gradient descent

Review last week

Linear Models

Linear Regression

Linear Classification

Types of prediction tasks

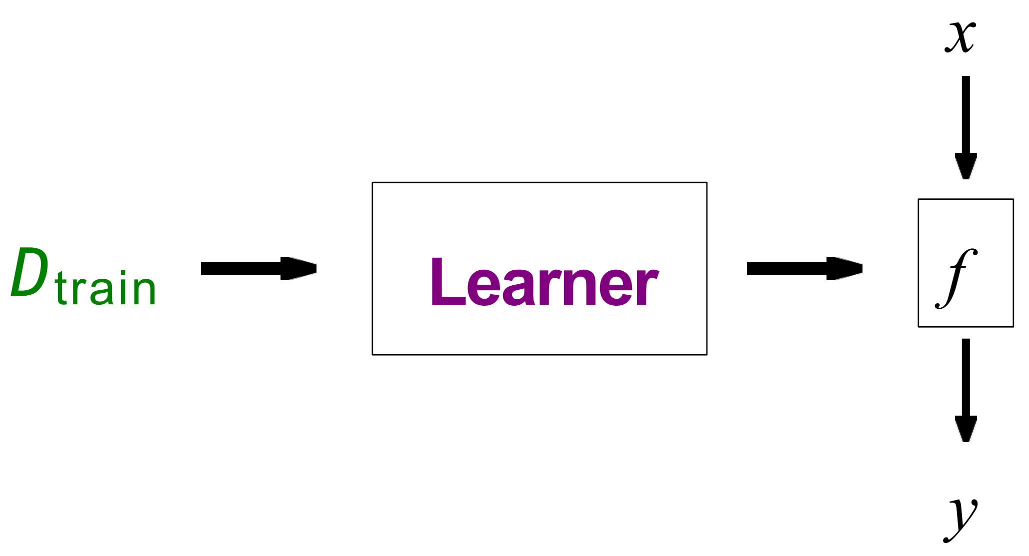
Binary classification (e.g., email ⇒ spam/not spam):

$$x \longrightarrow \boxed{f} \longrightarrow y \in \{-1, +1\}$$

Regression (e.g., location, year ⇒ housing price):

$$x \longrightarrow \boxed{f} \longrightarrow y \in \mathrm{R}$$

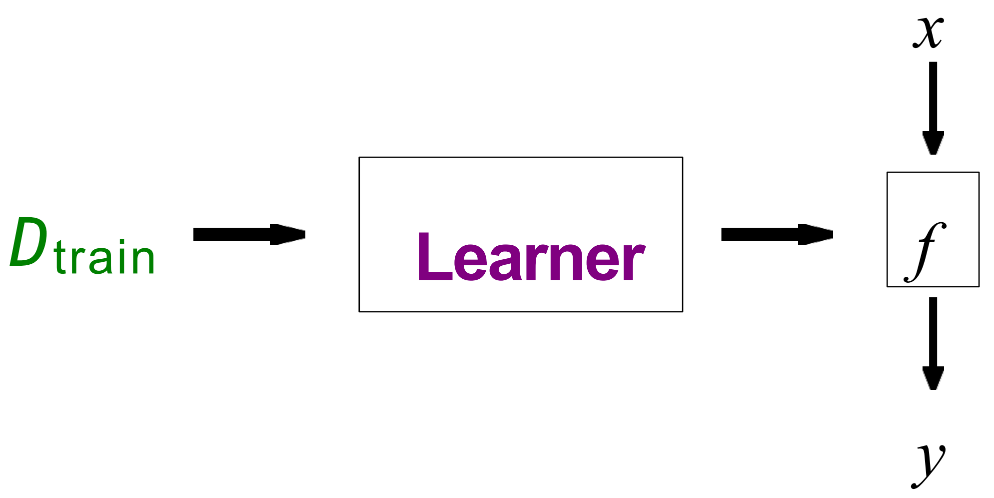
Supervised Learning, Linear Models



Linear Regression: f(x) = ŷ = w[0] * x[0] + w[1] * x[1] + ... + w[p] * x[p] + b

Supervised Learning, Linear Models

The learner finds the optimum values of *w[i]*s and *b*



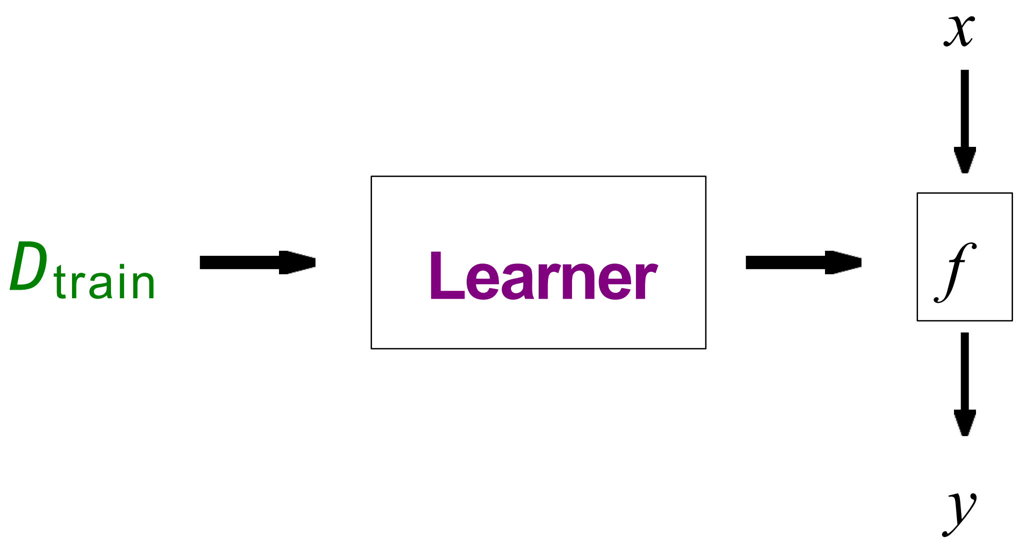
Linear Regression:

$$f(x) = \hat{y} = w[0] * x[0] + w[1] * x[1] + ... + w[p] * x[p] + b$$

w[i] is the slope
b is the y-axis offset or the intercept

Linear Classification (Binary)

- $\hat{y} = sign\ (w[0] * x[0] + w[1] * x[1] + \ldots + w[p] * x[p] + b)$



The learner finds the optimum values of w[i]s and b

Linear Regression and Cost function

$\hat{y} = w[0] * x[0] + w[1] * x[1] + ... + w[p] * x[p] + b$

- Linear regression looks for optimizing *w* and *b* such that it minimizes the cost function
- The cost function can be written as:

$$\sum_{i=1}^{M} (y_i - \hat{y_i})^2 = \sum_{i=1}^{M} \left(y_i - \sum_{j=0}^{p} w_j \times x_{ij} \right)^2$$

- The data-set has M instances and p features

Linear Regression, Ridge

- In Ridge regression, the cost function is altered by adding a penalty equivalent to square of the magnitude of the coefficients.

$$\sum_{i=1}^{M} (y_i - \hat{y_i})^2 = \sum_{i=1}^{M} \left(y_i - \sum_{j=0}^{p} w_j \times x_{ij} \right)^2 + \lambda \sum_{j=0}^{p} w_j^2$$

Cost function for ridge regression

Linear Regression, Lasso

- In Lasso Regression, coefficients are restricted to be close to zero and some coefficients are exactly zero.
- Cost function in Lasso:

$$\sum_{i=1}^{M}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{M}\left(y_i - \sum_{j=0}^{p} w_j \times x_{ij}\right)^2 + \lambda \sum_{j=0}^{p} |w_j|$$

Parul Pandey

Just like Ridge regression cost function, for lambda =0, the equation above

Linear Classification Algorithms

- Logistic Regression
- Linear Support Vector Machines (Linear SVMs)

- Both use L2 regularisation similar to Ridge

High Dimensions and Parameter C

- In high dimensions, linear models for classification become very powerful

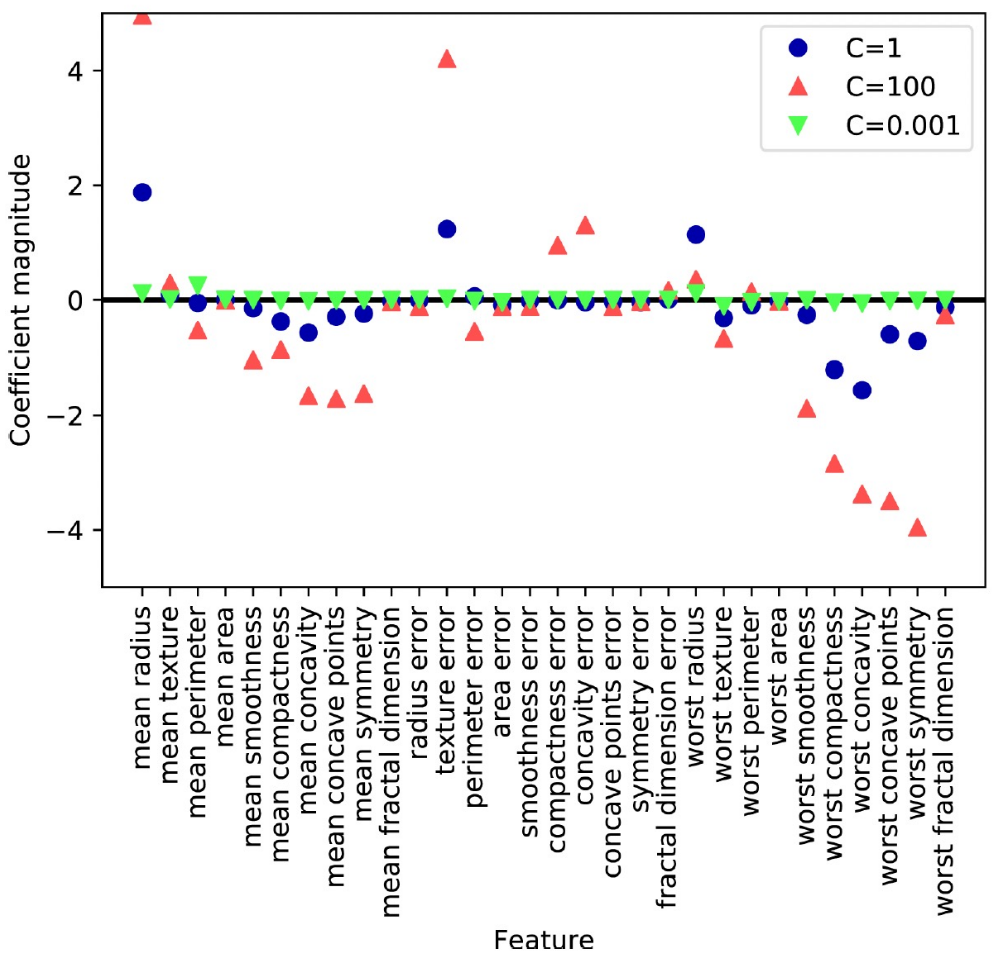


Figure 2-17. Coefficients learned by logistic regression on the Breast Cancer dataset for different values of C

Source: Muller and Guido’s book, page 63

High Dimensions and L1

- L1 regularization
- More interpretable model

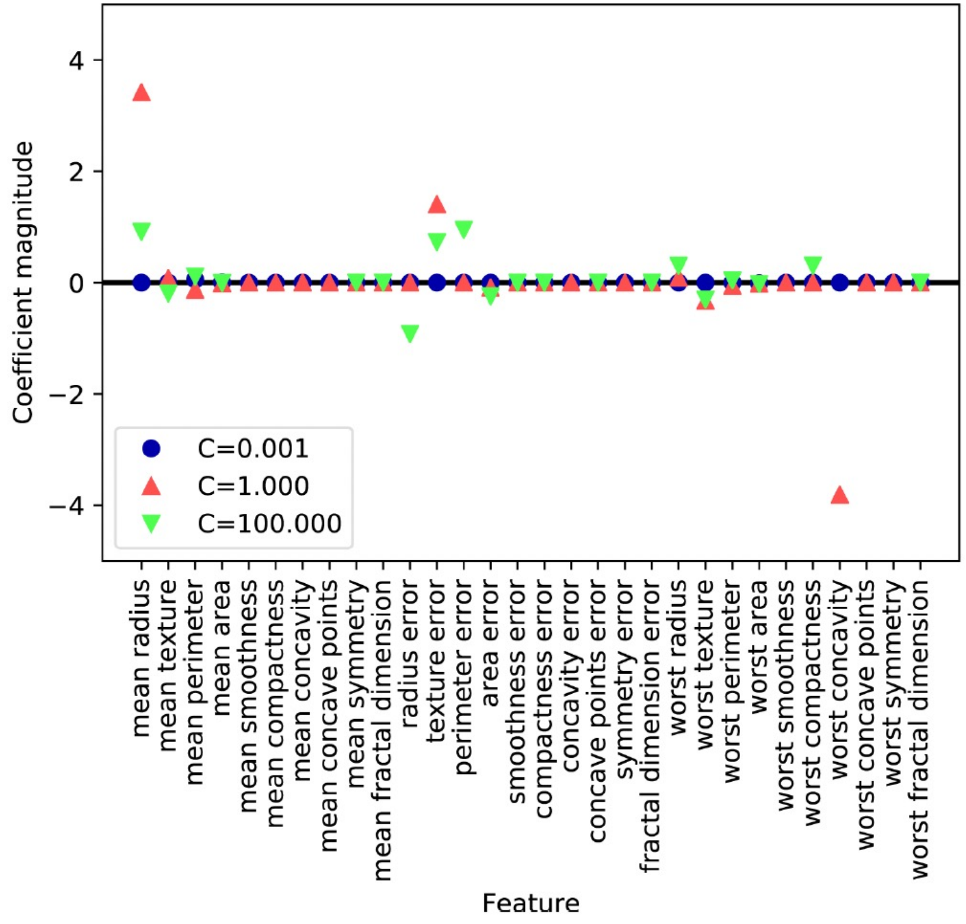


Figure 2-18. Coefficients learned by logistic regression with L1 penalty on the Breast Cancer dataset for different values of C

Source: Muller and Guido’s book, page 65

Linear Models for Multi-class Classification

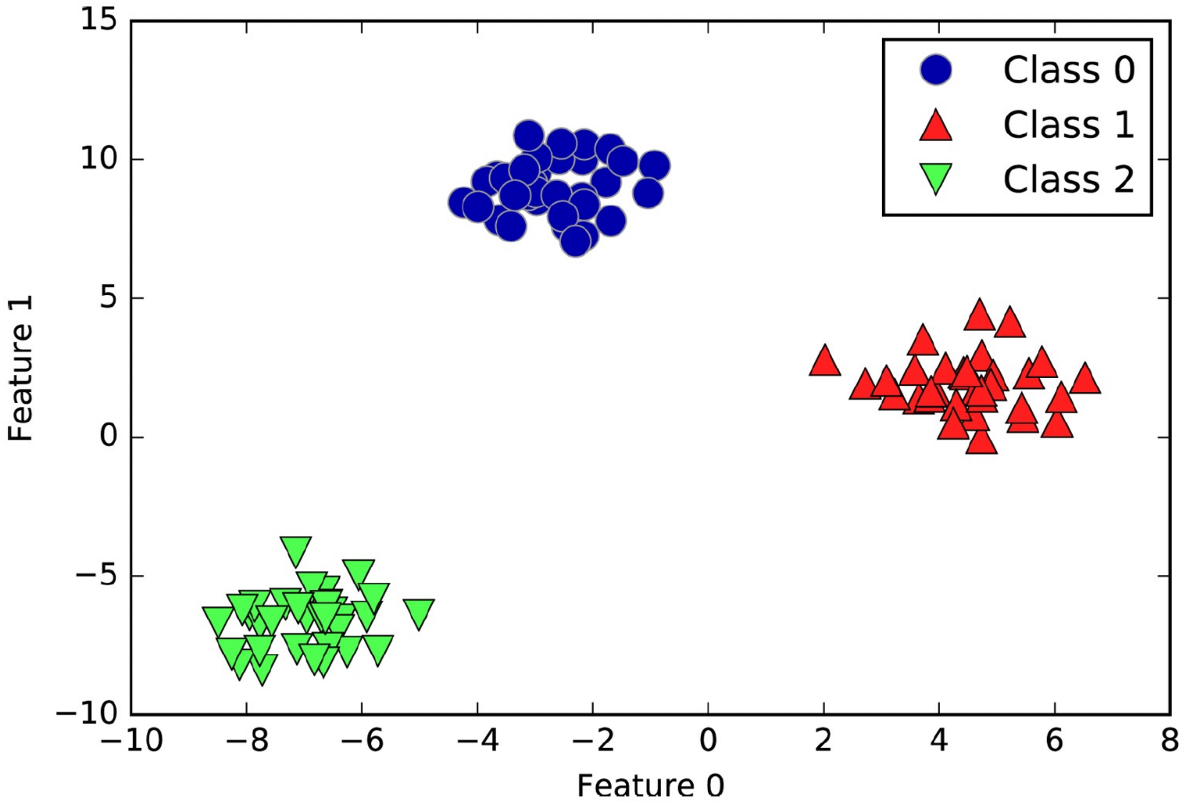


Figure 2-19. Two-dimensional toy dataset containing three classes

Source: Muller and Guido’s book, page 66

Multi-class classifier, one-vs.-rest classifier

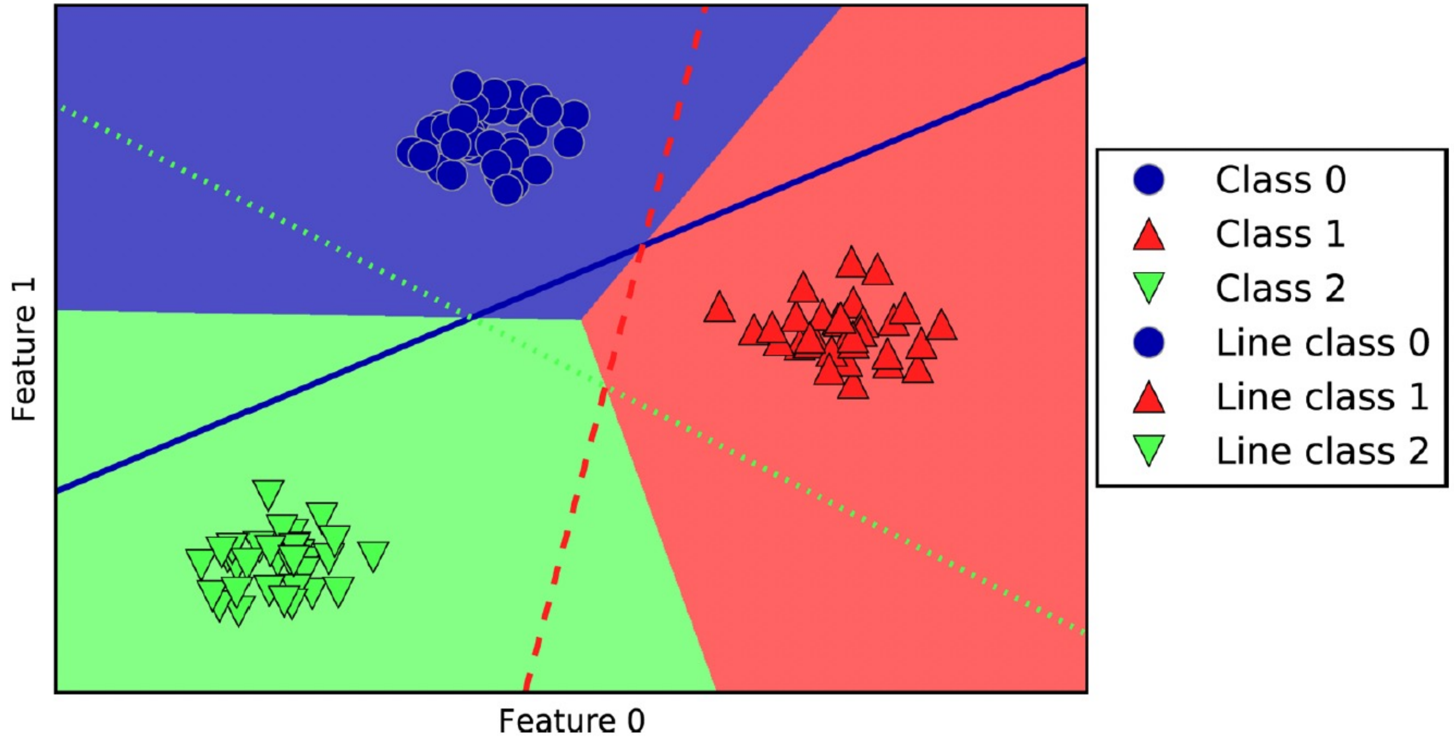


Figure 2-21. Multiclass decision boundaries derived from the three one-vs.-rest classifiers

Source: Muller and Guido's book, page 69

Strengths, Weaknesses and Parameters

Linear models

Parameters, alpha and C

- alpha in regression models
- C in Linear SVM and Logistic Regression

- Large values for alpha or small values for C mean simple models

Parameters, L1 or L2

- Use L1 regularization
  - If you assume that only a few of your features are important,
  - if interpretability of the model is important

- L2 regularization
  - default

Strengths and Weaknesses

- Fast in training
- Fast in prediction
- Scale well to very large datasets
- Work well with sparse data
- Relatively easy to understand how a prediction is made
- Not entirely clear why coefficients are the way they are
- Perform well when the number of features is large compared to the number of samples.
- in lower-dimensional spaces, other models might yield better generalization performance

Gradient Descent

Linear Regression and Gradient Descent

Learning as Optimisation

Simple Gradient Decent

Problems with simple Gradient Descent

Recap, Linear Regression and Cost function

*ŷ = w[0] * x[0] + w[1] * x[1] + ... + w[p] * x[p] + b*

- Linear regression looks for optimizing *w* and *b* such that it minimises the cost function
- The cost function can be written as:

$$\sum_{i=1}^{M}(y_i - \hat{y_i})^2 = \sum_{i=1}^{M}\left(y_i - \sum_{j=0}^{p} w_j \times x_{ij}\right)^2$$

The data-set has M instances and p features

- This cost function is in form of Residual Sum of Squares (RSS)

Cost or Loss function as Mean Squared Error

- Mean Square Error (MSE) is another form of Cost function
- MSE is RSS divided by total observed points (hence mean)

$$MSE = 1/N \times RSS$$

- You may also see Loss function in form of:

$$1/2N \times RSS$$

Gradient-based Optimisation & cost function

$$\hat{y} = w[0] * x[0] + w[1] * x[1] + ... + w[p] * x[p] + b$$

- Linear regression looks for optimizing ***w*** and ***b*** such that it minimises the cost function

- How can we find ***w***s and ***b*** that minimises the cost function?
- This is an optimisation problem
- One way to do optimisation is: **Gradient-based optimisation**

Gradient-based Optimisation

- Most ML algorithms involve optimisation
- Learning as optimisation
- Minimise/maximise a function $f\ (\boldsymbol{x})$ by altering $\boldsymbol{x}$
  - – Usually stated a minimization
  - – Maximization accomplished by minimizing $-f\ (\boldsymbol{x})$
- $f\ (\boldsymbol{x})$ referred to as objective or criterion
  - In minimization also referred to as loss function, cost, or error
  - Example is loss function of linear regression

Gradient Descent an iterative optimisation

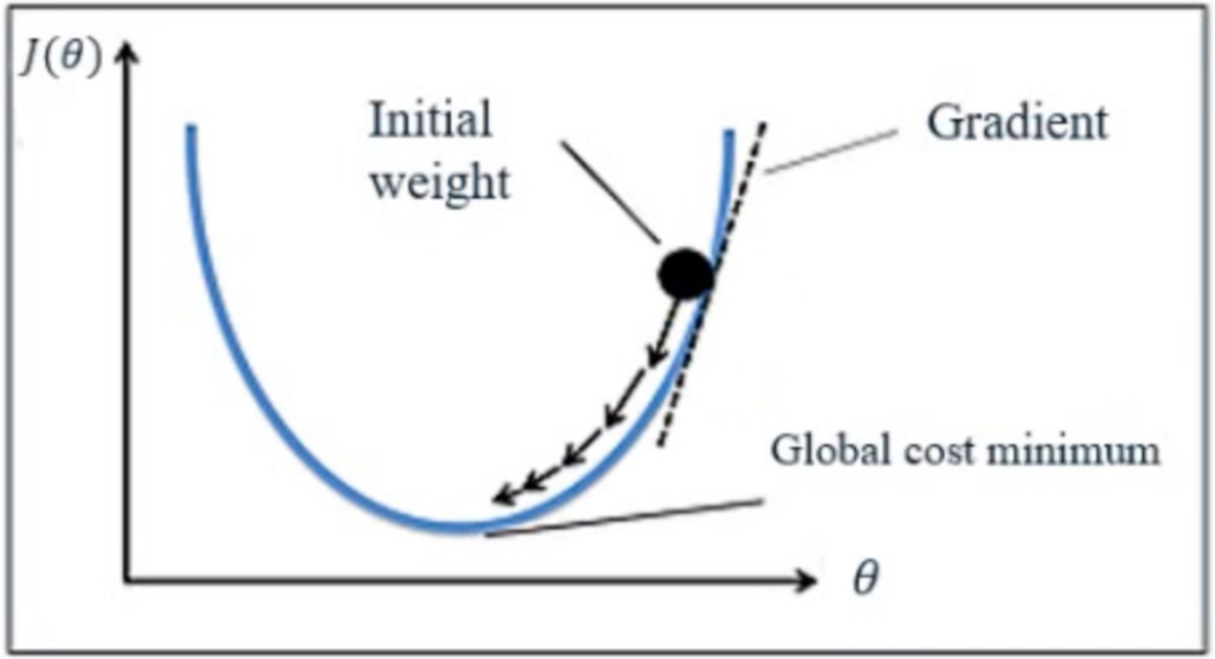
- Denote optimum value by

$$\boldsymbol{x}^* = \text{argminx}\ f(\boldsymbol{x})$$

- argminx f(x) is the value of x for which f(x) attains its minimum

- In many cases, there is NO analytical solution for x*.
- The most commonly employed iterative optimisation is gradient descent

Minimising Loss function by Gradient Descent



Calculus in Optimisation

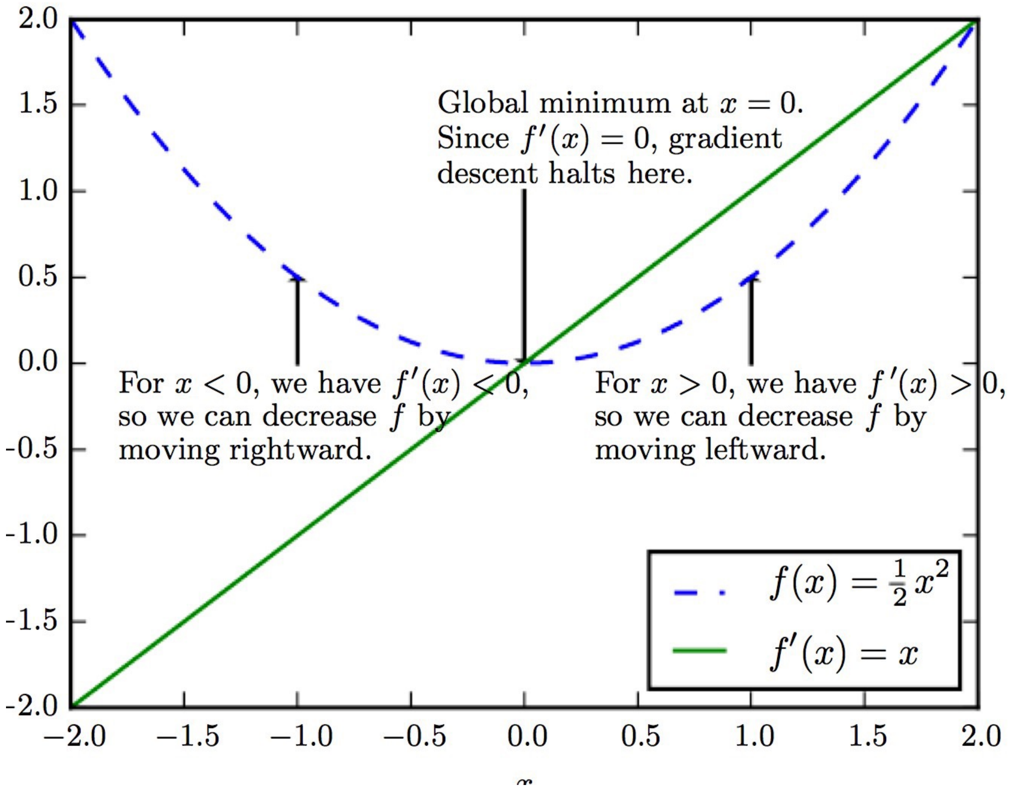
- Consider function $y = f(x)$, x, y real numbers.
  - Derivative of function denoted: $f'(x)$ or as dy/dx
    - Derivative $f'(x)$ gives the slope of $f(x)$ at point x
    - It specifies how to scale a small change in input to obtain a corresponding change in the output:

$$f(x + \varepsilon) \approx f(x) + \varepsilon f'(x)$$

  - It tells how you make a small change in input to make a small improvement in y
  - We know that $f(x - \varepsilon \operatorname{sign}(f'(x)))$ is less than $f(x)$ for small ε. Thus, we can reduce $f(x)$ by moving x in small steps with opposite sign of derivative
    - This technique is called ***gradient descent*** (Cauchy 1847)

Gradient Descent Illustrated

- Given function is $f(x)=½\,x^2$ which has a bowl shape with global minimum at $x=0$
  - Since $f'(x)=x$
    - For $x>0$, $f(x)$ increases with x and $f'(x)>0$
    - For $x<0$, $f(x)$ decreases with x and $f'(x)<0$
- Use $f'(x)$ to follow function downhill
  - Reduce $f(x)$ by going in direction opposite sign of derivative $f'(x)$



Linear Regression, Find the best Line

- Which line is a better fit to the training data?
- In many cases there is NO analytical solution
- So Iterative optimisation solution:
  - Start with a random line (*w* and *b*)
  - Measure error or loss between **line** and **data**
  - Move line to try and lower loss
  - Loop until best line found (loss minimised)

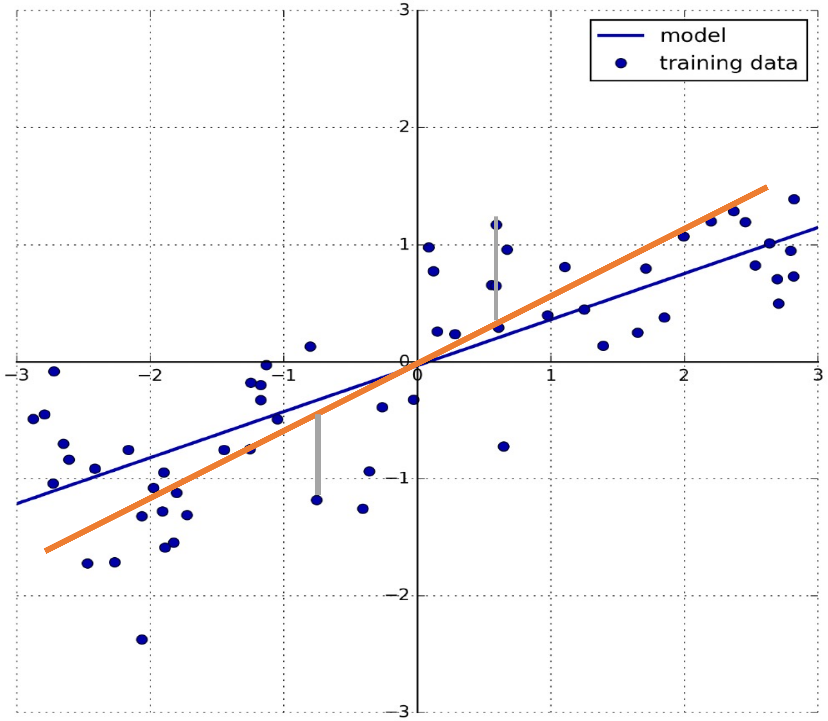


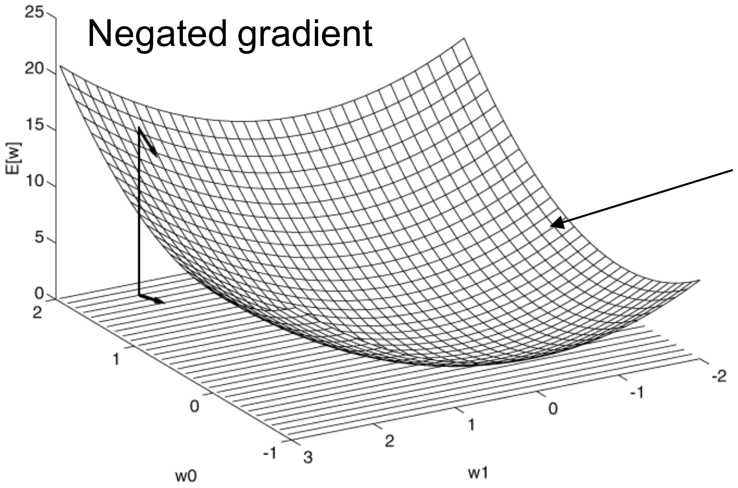
Figure 2-11. Predictions of a linear model on the wave dataset

Minimising with multiple inputs

- We often minimize functions with multiple inputs: $f\colon R^n \rightarrow R$
- For minimization to make sense there must still be only one (scalar) output

Application in ML: Minimize Error

Direction in w_0-w_1 plane producing steepest descent

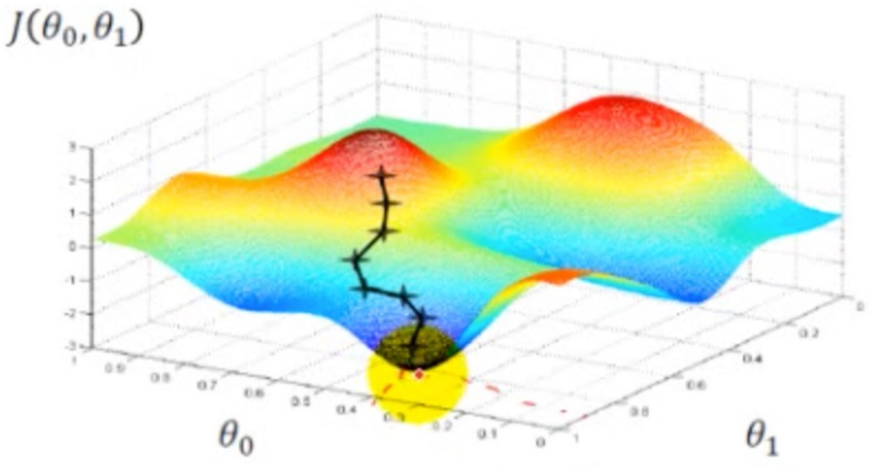


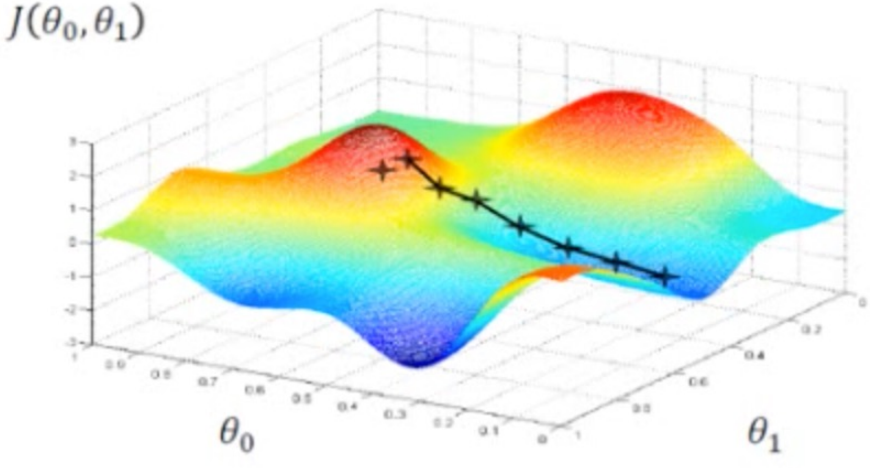
Error surface shows desirability of every weight vector, a parabola with a single global minimum

- Gradient descent search determines a weight vector w that minimizes $E(\mathrm{w})$ by
  - Starting with an arbitrary initial weight vector
  - Repeatedly modifying it in small steps
  - At each step, weight vector is modified in the direction that produces the steepest descent along the error surface

Examples of 3D Loss functions

- How many features we had in our training data, when cost functions look like below?





Gradient Descent, another notation

- When *w* denotes the set of parameters
- *E(w|x)* is the error with parameters *w* on the given training set *X*, we look for

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} E(\boldsymbol{w}|X)$$

Definition of Gradient Vector

- When *E(w)* is a differentiable function of a vector of variables, we have gradient vector composed of the partial derivatives
- The *Gradient (derivative)* of *E* with respect to each component of the vector w:

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n} \right]$$

Notice $\nabla E[w]$ is a vector of partial derivatives

Gradient Descent, Gradient Vector

- Specifies the direction that produces steepest increase in E
- Negative of this vector specifies direction of steepest decrease

- The gradient descent procedure to minimize E starts from a random w, and at each step, updates w, in the opposite direction of the gradient

Gradient Descent Rule, learning rate

$$w \leftarrow w + \Delta w$$

- where

$$\Delta w = -\eta \nabla E[w]$$

η is a positive constant called the learning rate

- Determines step size of gradient descent search

- Component Form of Gradient Descent

Can also be written as

$$w_i \leftarrow w_i + \Delta w_i$$

- where

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

Method of Gradient Descent

- The gradient points directly uphill, and the negative gradient points directly downhill
- Thus, we can decrease f by moving in the direction of the negative gradient
  - This is known as the method of steepest descent or gradient descent
- Steepest descent proposes a new point

$$\boldsymbol{x}' = \boldsymbol{x} - \eta \nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}\right)$$

  - where ε is the learning rate, a positive scalar.
    - Set to a small constant.

Simple Gradient Descent

Procedure Gradient-Descent (
$\boldsymbol{\theta}^1$ //Initial starting point
f //Function to be minimized
δ //Convergence threshold
)

1 t ← 1
2 do
3 $\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta}^{t} - \eta \nabla f\left(\boldsymbol{\theta}^{t}\right)$
4 $t \leftarrow t + 1$
5 while $\|\boldsymbol{\theta}^{t} - \boldsymbol{\theta}^{t-1}\| > \delta$
6 return $\left(\boldsymbol{\theta}^{t}\right)$

One-dimensional example

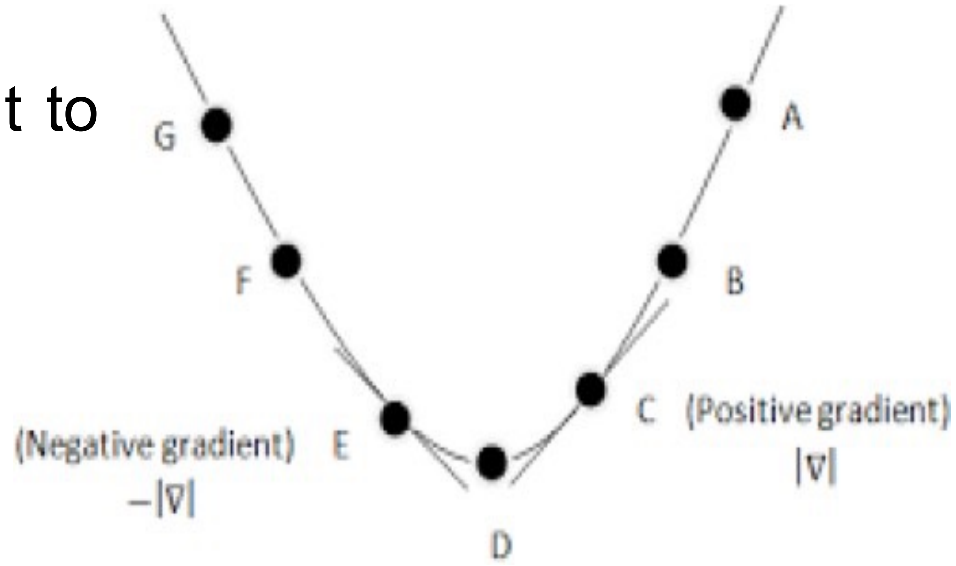
Let $f(\theta)=\theta^2$

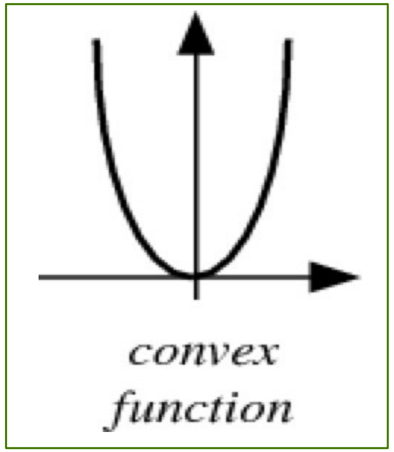
This function has minimum at $\theta=0$ which we want to determine using gradient descent

We have $f'(\theta)=2\theta$

For gradient descent, we update by $-f'(\theta)$

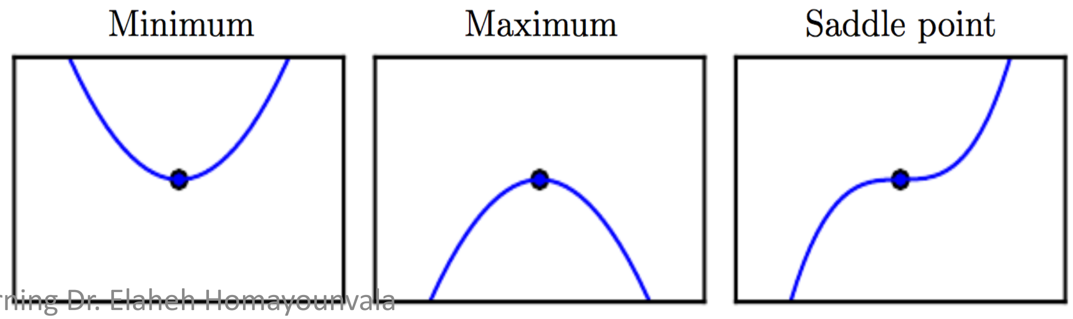
If $\theta^t > 0$ then $\theta^{t+1}<\theta^t$
If $\theta^t<0$ then $f'(\theta^t)=2\theta^t$ is negative, thus $\theta^{t+1}>\theta^t$





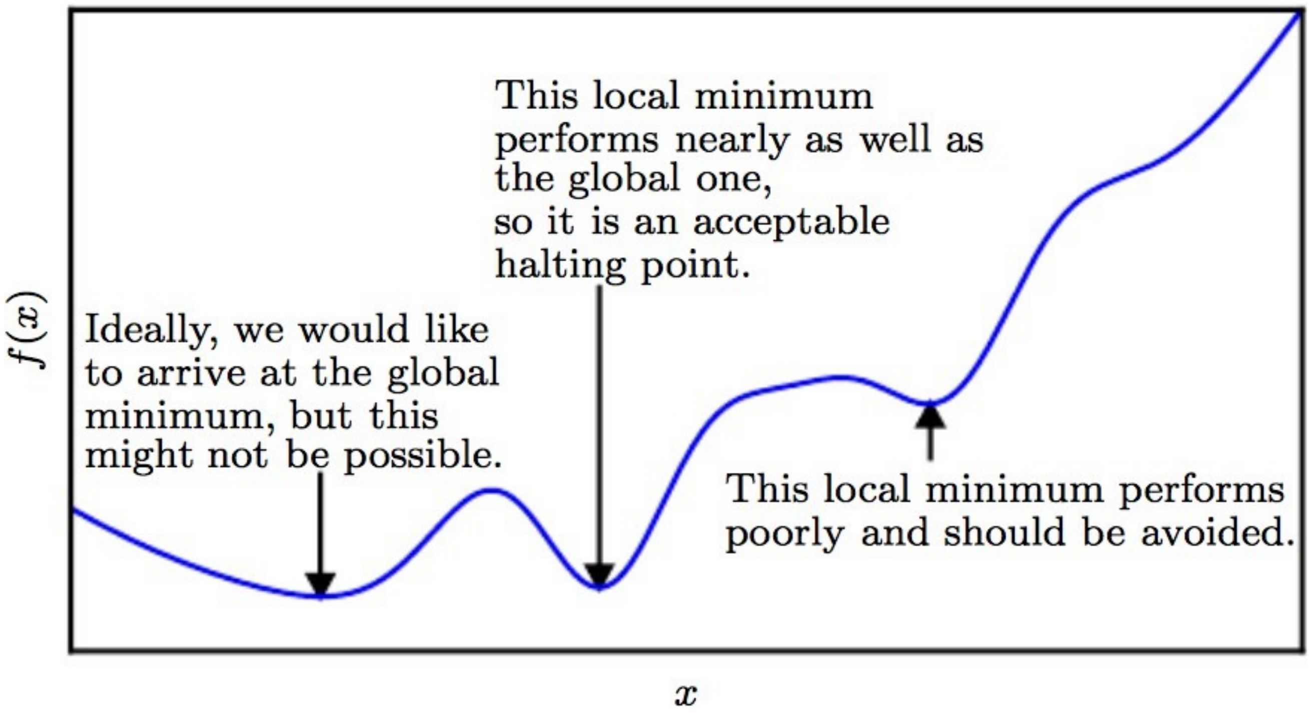
Stationary points, Local Optima

- When $f'(x)=0$ derivative provides no information about direction of move
- Points where $f'(x)=0$ are known as *stationary* or critical points
  - Local minimum/maximum: a point where $f(x)$ lower/ higher than all its neighbours
  - Saddle Points: neither maxima nor minima



Presence of multiple minima

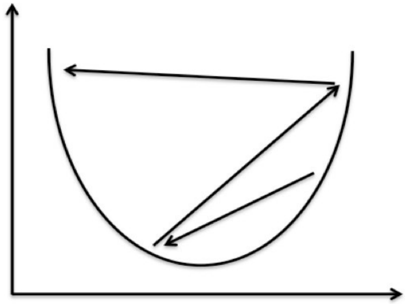
- Optimization algorithms may fail to find global minimum
- Generally, accept such solutions



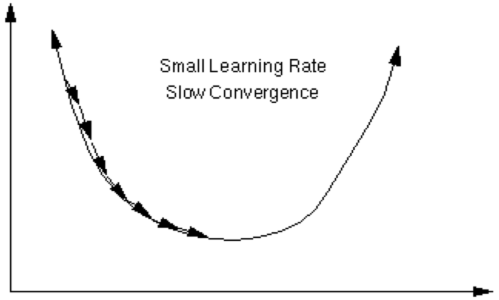
Difficulties with Simple Gradient Descent

- Performance depends on choice of learning rate η

$$\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta}^{t} - \eta \nabla f\left(\boldsymbol{\theta}^{t}\right)$$



- Large learning rate
  - May "overshoot" the minimum, May fail to converge, May even diverge
- Small learning rate
  - Extremely slow convergence



- Solution
  - Start with large η and settle on optimal value
  - Need a schedule for shrinking η

Convergence of Steepest Descent

- Steepest descent converges when every element of the gradient is zero
  - – In practice, very close to zero
- We may be able to avoid iterative algorithm and jump to the critical point by solving the equation for x

$$\boxed{\nabla_x f(x) = 0}$$