Linear regression

Supervised learning

Data: $D = \{D_1, D_2, \ldots, D_n\}$ a set of $n$ examples

$D_i = <x_i, y_i>$

$x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,d})$ is an input vector of size $d$

$y_i$ is the desired output (given by a teacher)

Objective: learn the mapping $f : X \rightarrow Y$

s.t. $y_i \approx f(x_i)$ for all $i = 1, \ldots, n$

- Regression: $Y$ is continuous
  Example: earnings, product orders $\rightarrow$ company stock price
- Classification: $Y$ is discrete
  Example: handwritten digit in binary form $\rightarrow$ digit label
Supervised learning examples

- **Regression:** Y is **continuous**

Debt/equity  | Earnings  | Future product orders  | Stock price
---|---|---|---
20  | 115  | 20  | 123.45
18  | 120  | 31  | 140.56
...

**Data:**

**Linear regression**

- **Function** \( f : X \rightarrow Y \)
- \( Y \) is a linear combination of input components

\[
f(x) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_d x_d = w_0 + \sum_{j=1}^{d} w_j x_j
\]

\( w_0, w_1, \ldots, w_k \) - **parameters (weights)**
Linear regression

• Shorter (vector) definition of the model
  – Include bias constant in the input vector
    \[ x = (1, x_1, x_2, \cdots x_d) \]
    \[ f(x) = w_0 x_0 + w_1 x_1 + w_2 x_2 + \cdots w_d x_d = w^T x \]
    \[ w_0, w_1, \ldots, w_k \text{ - parameters (weights)} \]

Input vector \( x \) \( \sum \)

Linear regression. Example

• 1 dimensional input \( x = (x_1) \)
Linear regression. Example.

- 2 dimensional input \( x = (x_1, x_2) \)

Linear regression: error

- **Data:** \( D_i = \langle x_i, y_i \rangle \)
- **Function:** \( x_i \rightarrow f(x_i) \)
- **Goal:** find the best set of model parameters
- **Error:** a measure of misfit of the model and the data
Linear regression: Error.

- **Data:** $D_i = \langle x_i, y_i \rangle$
- **Function:** $x_i \rightarrow f(x_i)$
- **Goal:** find the best set of model parameters
- **Error function**
  - a measure of misfit of the model and the data
  - in other words, it measures how much our predictions deviate from the desired answers

**Mean-squared error:** $Error(w, D)$
\[
J_n = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i, w))^2
\]

**Learning:**
We want to find the weights minimizing the error!

Linear regression: Optimization.

- We want the weights minimizing the error
\[
J_n = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i, w))^2 = \frac{1}{n} \sum_{i=1}^{n} (y_i - w^T x_i)^2
\]
- For the optimal set of parameters, derivatives of the error with respect to each parameter must be 0
\[
\frac{\partial}{\partial w_j} J_n(w) = -\frac{2}{n} \sum_{i=1}^{n} (y_i - w_0 x_{i,0} - w_1 x_{i,1} - \ldots - w_d x_{i,d}) x_{i,j} = 0
\]
- **Vector of derivatives:**
\[
\text{grad}_w (J_n(w)) = \nabla_w (J_n(w)) = -\frac{2}{n} \sum_{i=1}^{n} (y_i - w^T x_i) x_i = \mathbf{0}
\]
Linear regression: Optimization.

- \( \text{grad}_w (J_n(w)) = 0 \) defines a set of equations in \( w \)

\[
\frac{\partial}{\partial w_j} J_n(w) = -\frac{2}{n} \sum_{i=1}^{n} (y_i - w_0 x_{i,0} - w_1 x_{i,1} - \ldots - w_d x_{i,d}) x_{i,j} = 0
\]

\[
\frac{\partial}{\partial w_0} J_n(w) = \sum_{i=1}^{n} (y_i - w_0 x_{i,0} - w_1 x_{i,1} - \ldots - w_d x_{i,d}) = 0
\]

\[
\frac{\partial}{\partial w_1} J_n(w) = \sum_{i=1}^{n} (y_i - w_0 x_{i,0} - w_1 x_{i,1} - \ldots - w_d x_{i,d}) x_{i,1} = 0
\]

\[
\vdots
\]

\[
\frac{\partial}{\partial w_j} J_n(w) = \sum_{i=1}^{n} (y_i - w_0 x_{i,0} - w_1 x_{i,1} - \ldots - w_d x_{i,d}) x_{i,j} = 0
\]

\[
\frac{\partial}{\partial w_d} J_n(w) = \sum_{i=1}^{n} (y_i - w_0 x_{i,0} - w_1 x_{i,1} - \ldots - w_d x_{i,d}) x_{i,d} = 0
\]

Solving linear regression

\[
\frac{\partial}{\partial w_j} J_n(w) = \sum_{i=1}^{n} (y_i - w_0 x_{i,0} - w_1 x_{i,1} - \ldots - w_d x_{i,d}) x_{i,j} = 0
\]

By rearranging the terms we get a system of linear equations with \( d+1 \) unknowns

\[
Aw = b
\]

\[
\begin{align*}
 w_0 &\frac{n}{i=1} x_{i,0} 1 + \frac{n}{i=1} x_{i,1} 1 + \ldots + \frac{n}{i=1} x_{i,j} 1 + \ldots + \frac{n}{i=1} x_{i,d} 1 = \frac{n}{i=1} y_i 1 \\
w_0 &\frac{n}{i=1} x_{i,0} x_{i,1} + \frac{n}{i=1} x_{i,1} x_{i,1} + \ldots + \frac{n}{i=1} x_{i,j} x_{i,j} + \ldots + \frac{n}{i=1} x_{i,d} x_{i,d} = \frac{n}{i=1} y_i x_{i,j} \\
\vdots \\
w_0 &\frac{n}{i=1} x_{i,0} x_{i,j} + \frac{n}{i=1} x_{i,1} x_{i,j} + \ldots + \frac{n}{i=1} x_{i,j} x_{i,j} + \ldots + \frac{n}{i=1} x_{i,d} x_{i,j} = \frac{n}{i=1} y_i x_{i,j}
\end{align*}
\]
Solving linear regression

- The optimal set of weights satisfies:

\[ \nabla_w (J_n(w)) = - \frac{2}{n} \sum_{i=1}^{n} (y_i - w^T x_i) x_i = 0 \]

Leads to a system of linear equations (SLE) with \( d+1 \) unknowns of the form

\[ \mathbf{A} \mathbf{w} = \mathbf{b} \]

Solution to SLE:

\[ \mathbf{w} = \mathbf{A}^{-1} \mathbf{b} \]

Assuming \( \mathbf{X} \) is an \( n \times d \) data matrix with rows corresponding to examples and columns to inputs, and \( \mathbf{y} \) is \( n \times 1 \) vector of outputs, then

\[ \mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \]

Gradient descent solution

Objective: optimize the weights in the linear regression model

\[ J_n = Error(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i, \mathbf{w}))^2 \]

An alternative to SLE solution:

- **Gradient descent**

  Idea:

  - Adjust weights in the direction that improves the Error
  - The gradient tells us what is the right direction

\[ \mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla_w Error_i(\mathbf{w}) \]

\( \alpha > 0 \) - a learning rate (scales the gradient changes)
Gradient descent method

• Descend using the gradient information

\[ Error(w) \]

\[ \nabla_w Error(w) |_{w^*} \]

Direction of the descent

• Change the value of \( w \) according to the gradient

\[ w \leftarrow w - \alpha \nabla_w Error_i(w) \]

\( \alpha > 0 \) a learning rate (scales the gradient changes)

Gradient descent method

• Iteratively approaches the optimum of the Error function

\[ Error(w) \]

\[ w^{(0)} w^{(1)} w^{(2)} w^{(3)} \]

w
Batch vs online gradient algorithm

- The error function defined on the complete dataset $D$
  \[ J_n = Error(w) = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i, w))^2 \]

- We say we are learning the model in the **batch mode**:
  - All examples are available at the time of learning
  - Weights are optimizes with respect to all training examples

- An alternative is to learn the model in the **online mode**
  - Examples are arriving sequentially
  - Model weights are updated after every example
  - If needed examples seen can be forgotten

Online gradient algorithm

- **The error function** for the complete dataset $D$
  \[ J_n = Error(w, D) = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i, w))^2 \]

- **Error for one example**  \[ D_i = \langle x_i, y_i \rangle \]
  \[ J_{online} = Error_i(w, x_i) = \frac{1}{2} (y_i - f(x_i, w))^2 \]

- **Online gradient method**: changes weights after every example
  \[ w_j \leftarrow w_j - \alpha \frac{\partial}{\partial w_j} Error_i(w) \]

- **vector form**:
  \[ w \leftarrow w - \alpha \nabla_w Error_i(w) \]

\[ \alpha > 0 \quad - \text{Learning rate that depends on the number of updates} \]
Online gradient method

Linear model \[ f(x) = w^T x \]
On-line error \[ J_{\text{online}} = \text{Error}_i(w) = \frac{1}{2} (y_i - f(x_i, w))^2 \]

On-line algorithm: generates a sequence of online updates

(i)-th update step with: \[ D_i = \langle x_i, y_i \rangle \]

j-th weight:
\[ w_j^{(i)} \leftarrow w_j^{(i-1)} - \alpha(i) \frac{\partial \text{Error}_i(w)}{\partial w_j} \bigg|_{w^{(i-1)}} \]
\[ w_j^{(i)} \leftarrow w_j^{(i-1)} + \alpha(i)(y_i - f(x_i, w^{(i-1)}))x_{i,j} \]

Fixed learning rate: \[ \alpha(i) = C \]
Annealed learning rate: \[ \alpha(i) \approx \frac{1}{i} \]
- Use a small constant
- Gradually rescales changes

Online regression algorithm

Online-linear-regression (stopping criterion)
Initialize weights \[ w = (w_0, w_1, w_2 \ldots w_d) \]
initialize \( i = 1; \)
while \( \text{stopping criterion} = \text{FALSE} \)

select the next data point \[ D_i = (x_i, y_i) \]
set learning rate \( \alpha(i) \)
update weight vector \[ w \leftarrow w + \alpha(i)(y_i - f(x_i, w))x_i \]
end
return weights

Advantages: very easy to implement, works on continuous data streams
Extensions of simple linear model

Replace inputs to linear units with \( m \) feature (basis) functions to model nonlinearities

\[
f(x) = w_0 + \sum_{j=1}^{m} w_j \phi_j(x)
\]

\( \phi_j(x) \) - an arbitrary function of \( x \)
Extensions of simple linear model

• Models linear in the parameters we want to fit

\[ f(\mathbf{x}) = w_0 + \sum_{k=1}^{m} w_k \phi_k(\mathbf{x}) \]

\( w_0, w_1 \ldots w_m \) - parameters
\( \phi_1(\mathbf{x}), \phi_2(\mathbf{x}) \ldots \phi_m(\mathbf{x}) \) - feature or basis functions

• Basis functions examples:

– a higher order polynomial, one-dimensional input \( \mathbf{x} = (x_1) \)

\( \phi_1(x) = x \quad \phi_2(x) = x^2 \quad \phi_3(x) = x^3 \)

– Multidimensional quadratic \( \mathbf{x} = (x_1, x_2) \)

\( \phi_1(\mathbf{x}) = x_1 \quad \phi_2(\mathbf{x}) = x_1^2 \quad \phi_3(\mathbf{x}) = x_2 \quad \phi_4(\mathbf{x}) = x_2^2 \quad \phi_5(\mathbf{x}) = x_1 x_2 \)
Extensions of simple linear model

- Models linear in the parameters we want to fit
  \[ f(x) = w_0 + \sum_{k=1}^{m} w_k \phi_k(x) \]
  \( w_0, w_1 \ldots w_m \) - parameters
\( \phi_1(x), \phi_2(x) \ldots \phi_m(x) \) - feature or basis functions

- Basis functions examples:
  - A higher order polynomial, one-dimensional input \( x = (x_i) \)
    \( \phi_1(x) = x \quad \phi_2(x) = x^2 \quad \phi_3(x) = x^3 \)
  - Multidimensional quadratic \( x = (x_1, x_2) \)
    \( \phi_1(x) = x_1 \quad \phi_2(x) = x_1^2 \quad \phi_3(x) = x_2 \quad \phi_4(x) = x_2^2 \quad \phi_5(x) = x_1x_2 \)
  - Other types of basis functions
    \( \phi_1(x) = \sin x \quad \phi_2(x) = \cos x \)

---

Extensions of simple linear model

The same techniques as for the linear model to learn the weights

- Error function
  \[ J_n = 1/n \sum_{i=1}^{n} (y_i - f(x_i, w))^2 \]

Assume:
\( \varphi(x_i) = (1, \phi_1(x_i), \phi_2(x_i), \ldots, \phi_m(x_i)) \)

\[ \nabla_w J_n(w) = -2/n \sum_{i=1}^{n} (y_i - f(x_i)) \varphi(x_i) = 0 \]

- Leads to a system of \( m \) linear equations

\[ w_0 \sum_{i=1}^{n} \varphi_j(x_i) + \ldots + w_j \sum_{i=1}^{n} \varphi_j(x_i) \phi_j(x_i) + \ldots + w_m \sum_{i=1}^{n} \varphi_m(x_i) \phi_m(x_i) = \sum_{i=1}^{n} y_i \phi_j(x_i) \]

- Can be solved exactly like the linear case
Example. Regression with polynomials.

Regression with polynomials of degree m

- **Data instances:** pairs of \( <x, y> \)
- **Feature functions:** \( m \) feature functions
  \[
  \phi_i(x) = x^i \quad i = 1, 2, \ldots, m
  \]
- **Function to learn:**
  \[
  f(x, w) = w_0 + \sum_{i=1}^{m} w_i \phi_i(x) = w_0 + \sum_{i=1}^{m} w_i x^i
  \]

Linear model example
Non-linear (quadratic) model