Linear regression

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

Supervised learning

Data: \( D = \{D_1, D_2, ..., D_n\} \) a set of \( n \) examples
\[ D_i = \langle x_i, y_i \rangle \]
\[ x_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,d}) \] is an input vector of size \( d \)
\[ y_i \] is the desired output (given by a teacher)

Objective: learn the mapping \( f : X \rightarrow Y \)
\[ \text{s.t. } y_i \approx f(x_i) \text{ 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

Data:

<table>
<thead>
<tr>
<th>Data</th>
<th>Debt/equity</th>
<th>Earnings</th>
<th>Future prod orders</th>
<th>Stock price</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20</td>
<td>115</td>
<td>20</td>
<td>123.45</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>120</td>
<td>31</td>
<td>140.56</td>
</tr>
<tr>
<td></td>
<td>…</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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
  
  \[ x = (1, x_1, x_2, \ldots x_d) \]
  
  \[ f(x) = w_0 x_0 + w_1 x_1 + w_2 x_2 + \ldots + w_d x_d = w^T x \]
  
  \( w_0, w_1, \ldots w_k \) - parameters (weights)

\[
\begin{align*}
\text{Input vector} & \quad \sum f(x, w) \\
x & \quad \begin{cases} 1 \quad w_0 \\
x_1 \quad w_1 \\
x_2 \quad w_2 \\
\vdots \quad \vdots \\
x_d \quad w_d 
\end{cases}
\end{align*}
\]

Linear regression. Error.

- **Data:**  \( D_i = \langle x_i, y_i \rangle \)
- **Function:** \( x_i \rightarrow f(x_i) \)
- We would like to have \( y_i \approx f(x_i) \) for all \( i = 1, \ldots, n \)

- **Error function**
  - measures how much our predictions deviate from the desired answers
  
  \[ J_n = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2 \]

- **Learning:**
  
  We want to find the weights minimizing the error!
Linear regression. Example

• 1 dimensional input \[ x = (x_1) \]

Linear regression. Example.

• 2 dimensional input \[ x = (x_1, x_2) \]
Linear regression. Optimization.

- We want the weights minimizing the error

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

- 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:

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

---

Linear regression. Optimization.

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

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

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

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

\[ \vdots \]

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

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

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

\[
Aw = b
\]

\[
\begin{align*}
\sum_{i=1}^{n} x_{i,0} w_0 + \sum_{i=1}^{n} x_{i,1} w_1 + \ldots + \sum_{i=1}^{n} x_{i,d} w_d &= \sum_{i=1}^{n} y_i \\
\sum_{i=1}^{n} x_{i,0} x_{i,1} w_0 + \sum_{i=1}^{n} x_{i,1} x_{i,1} w_1 + \ldots + \sum_{i=1}^{n} x_{i,d} x_{i,1} w_d &= \sum_{i=1}^{n} y_i x_{i,1} \\
\vdots & \vdots \\
\sum_{i=1}^{n} x_{i,0} x_{i,j} w_0 + \sum_{i=1}^{n} x_{i,1} x_{i,j} w_1 + \ldots + \sum_{i=1}^{n} x_{i,d} x_{i,j} w_d &= \sum_{i=1}^{n} y_i x_{i,j} \\
\end{align*}
\]

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

\[
Aw = b
\]

\[
\begin{align*}
\sum_{i=1}^{n} x_{i,0} x_{i,j} w_0 + \sum_{i=1}^{n} x_{i,1} x_{i,j} w_1 + \ldots + \sum_{i=1}^{n} x_{i,d} x_{i,j} w_d &= \sum_{i=1}^{n} y_i x_{i,j} \\
\end{align*}
\]

**Solution to SLE:**

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

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

\[
w = (X^T X)^{-1} X^T y
\]
Gradient descent solution

**Goal:** the weight optimization in the linear regression model

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

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

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

  \[ \alpha > 0 \quad \text{a learning rate (scales the gradient changes)} \]

Gradient descent method

- Descend using the gradient information

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

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

  \[ \alpha > 0 \quad \text{a learning rate (scales the gradient changes)} \]
Gradient descent method

• Iteratively approaches the optimum of the Error function

\[ Error(w) \]

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

Batch vs Online regression 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 is defined for the complete dataset $D$
  \[ J_n = Error(w) = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i, w))^2 \]
• Error for one example $D_i = \langle x_i, y_i \rangle$
  \[ J_{\text{online}} = Error_i(w) = \frac{1}{2} (y_i - f(x_i, w))^2 \]
• 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 $J_{\text{online}} = Error_i(w) = \frac{1}{2} (y_i - f(x_i, w))^2$
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:
  \[ w_j^{(i)} \leftarrow w_j^{(i-1)} - \alpha(i) \frac{\partial Error_i(w)}{\partial w_j} \big|_{w^{(i-1)}} \]
  \[ w_j^{(i)} \leftarrow w_j^{(i-1)} + \alpha(i)(y_i - f(x_i, w^{(i-1)}))x_{i,j} \]

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 \( \text{stopping}_\text{criterion} = \text{FALSE} \)

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, continuous data streams

---

**On-line learning.  Example**

![Graphs showing on-line learning example](image.png)
Extensions of simple linear model

Replace inputs to linear units with \( m \) feature (basis) functions to model non-linearities

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

\( \phi_j(x) \) - an arbitrary function of \( x \)

- 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 \( x = (x_1) \)

\[
\phi_1(x) = x \quad \phi_2(x) = x^2 \quad \phi_3(x) = x^3
\]
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 \( x = (x_1) \)
    \( \phi_1(x) = x \quad \phi_2(x) = x^2 \quad \phi_3(x) = x^3 \)
  - Multidimensional quadratic \( x = (x_1, x_2) \)
    \( \phi_1(x) = x_1 \quad \phi_2(x) = x_1^2 \quad \phi_3(x) = x_2 \quad \phi_4(x) = x_2^2 \quad \phi_5(x) = x_1 x_2 \)

---

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 \( x = (x_1) \)
    \( \phi_1(x) = x \quad \phi_2(x) = x^2 \quad \phi_3(x) = x^3 \)
  - Multidimensional quadratic \( x = (x_1, x_2) \)
    \( \phi_1(x) = x_1 \quad \phi_2(x) = x_1^2 \quad \phi_3(x) = x_2 \quad \phi_4(x) = x_2^2 \quad \phi_5(x) = x_1 x_2 \)
  - 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** \( J_n(w) = 1/n \sum_{i=1}^{n} (y - f(x_i))^2 \)

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

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

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

\[ w_0 \sum_{i=1}^{n} \phi_1(x_i) + \ldots + w_j \sum_{i=1}^{n} \phi_j(x_i) \phi_j(x_i) + \ldots + w_m \sum_{i=1}^{n} \phi_m(x_i) \phi_m(x_i) = \sum_{i=1}^{n} y_i \phi_j(x_i) \]

- 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:**
  \[ f(x, w) = w_0 + \sum_{i=1}^{m} w_i \phi_i(x) = w_0 + \sum_{i=1}^{m} w_i x^i \]
Learning with feature functions

Function to learn:

\[ f(x, w) = w_0 + \sum_{i=1}^{k} w_i \phi_i(x) \]

On line gradient update for the \( <x,y> \) pair

\[ w_0 = w_0 + \alpha(y - f(x, w)) \]

\[ \vdots \]

\[ w_j = w_j + \alpha(y - f(x, w))\phi_j(x) \]

Gradient updates are of the same form as in the linear regression models

---

Linear model example
Non-linear model

Linear regression model

- **Linear model:** \( y = f(x, w) = w^T x \)

- Notice: the above model does not try to explain variation in observed \( y \)s for the data
Statistical model of regression

A statistical model of linear regression:

\[ y = w^T x + \varepsilon \]
\[ \varepsilon \sim N(0, \sigma^2) \]

\[ \varepsilon \] is a random noise, represents things we cannot capture with \( w^T x \)

\[ y \sim N(w^T x, \sigma^2) \]
ML estimation of the parameters

- **likelihood of predictions** = the probability of observing outputs $y$ in $D$ given $w, \sigma$

\[ L(D, w, \sigma) = \prod_{i=1}^{n} p(y_i \mid x_i, w, \sigma) \]

- **Maximum likelihood estimation of parameters** $w$
  - parameters maximizing the likelihood of predictions
  \[ w^* = \arg \max_w \prod_{i=1}^{n} p(y_i \mid x_i, w, \sigma) \]

- **Log-likelihood** trick for the ML optimization
  - Maximizing the log-likelihood is equivalent to maximizing the likelihood
  \[ l(D, w, \sigma) = \log(L(D, w, \sigma)) = \log \prod_{i=1}^{n} p(y_i \mid x_i, w, \sigma) \]

ML estimation of the parameters

- **Using conditional density**
  \[ p(y \mid x, w, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} \exp\left[ -\frac{1}{2\sigma^2} (y - f(x, w))^2 \right] \]

- **We can rewrite** the log-likelihood as
  \[
  l(D, w, \sigma) = \log(L(D, w, \sigma)) = \log \prod_{i=1}^{n} p(y_i \mid x_i, w, \sigma) \\
  = \sum_{i=1}^{n} \log p(y_i \mid x_i, w, \sigma) = \sum_{i=1}^{n} \left\{ -\frac{1}{2\sigma^2} (y_i - w^T x_i)^2 - c(\sigma) \right\} \\
  = -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - w^T x_i)^2 + C(\sigma) \\
  \]

Did we see a similar expression before?
ML estimation of the parameters

- Using conditional density
  \[ p(y \mid x, w, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} \exp\left[ - \frac{1}{2\sigma^2} (y - f(x, w))^2 \right] \]

- We can rewrite the log-likelihood as
  \[
  l(D, w, \sigma) = \log(L(D, w, \sigma)) = \log \prod_{i=1}^{n} p(y_i \mid x_i, w, \sigma)
  = \sum_{i=1}^{n} \log p(y_i \mid x_i, w, \sigma) = \sum_{i=1}^{n} \left\{ - \frac{1}{2\sigma^2} (y_i - w^T x_i)^2 - c(\sigma) \right\}
  = - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - w^T x_i)^2 + C(\sigma)
  \]

- Maximizing the predictive log likelihood with regard to \( w \), is equivalent to minimizing the mean squared error function

ML estimation of parameters

- Criteria based on the mean squares error function and the log likelihood of the output are related
  \[ J_{\text{online}}(y_i, x_i) = \frac{1}{2\sigma^2} \log p(y_i \mid x_i, w, \sigma) + c(\sigma) \]

- We know how to optimize parameters \( w \)
  – the same approach as used for the least squares fit

- But what is the ML estimate of the variance of the noise?
  - Maximize \( l(D, w, \sigma) \) with respect to variance
  \[
  \hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i, \hat{w}))^2
  = \text{mean square prediction error for the best predictor}
  \]