Learning probability distributions

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

Density estimation

Data: \( D = \{D_1, D_2, \ldots, D_n\} \)
\( D_i = \mathbf{x}_i \) a vector of attribute values

Objective: try to estimate the underlying true probability distribution over variables \( \mathbf{X} \), \( p(\mathbf{X}) \), using examples in \( D \)

Standard (iid) assumptions: Samples
- are independent of each other
- come from the same (identical) distribution (fixed \( p(\mathbf{X}) \))
Learning via parameter estimation

In this lecture we consider **parametric density estimation**

**Basic settings:**
- A set of random variables $X = \{X_1, X_2, \ldots, X_d\}$
- **A model of the distribution** over variables in $X$
  - with parameters $\Theta$
- **Data** $D = \{D_1, D_2, \ldots, D_n\}$

**Objective:** find parameters $\hat{\Theta}$ that fit the data the best

- What is the best set of parameters?
  - There are various criteria one can apply here.

---

Parameter estimation. Basic criteria.

- **Maximum likelihood (ML)**
  
  maximize $p(D | \Theta, \xi)$

  \[\xi\] represents prior (background) knowledge

- **Maximum a posteriori probability (MAP)**
  
  maximize $p(\Theta | D, \xi)$

  **Selects the mode of the posterior**

  \[
p(\Theta | D, \xi) = \frac{p(D | \Theta, \xi) p(\Theta | \xi)}{p(D | \xi)}
\]
Maximum likelihood (ML) estimate.

**Coin example:** we have a coin that can be biased

**Outcomes:** two possible values -- head or tail

**Data:** $D$ a sequence of outcomes such that

- **head** $x_i = 1$
- **tail** $x_i = 0$

**Model:** probability of a head $\theta$

probability of a tail $(1-\theta)$

**ML Solution:**

\[
\theta_{ML} = \frac{N_1}{N} = \frac{N_1}{N_1 + N_2}
\]

---

**Maximum likelihood estimate. Example**

- **Assume** the unknown and possibly biased coin
- **Probability of the head is** $\theta$
- **Data:**

  H H T T H H T H T H T T H T H H H T H H H T H H T

  - **Heads:** 15
  - **Tails:** 10

What is the ML estimate of the probability of a head and a tail?
Maximum likelihood estimate. Example

- Assume the unknown and possibly biased coin
- Probability of the head is $\theta$
- **Data:**
  - Heads: 15
  - Tails: 10

What is the ML estimate of the probability of head and tail?

**Head:**

$$\theta_{ML} = \frac{N_1}{N} = \frac{N_1}{N_1 + N_2} = \frac{15}{25} = 0.6$$

**Tail:**

$$(1 - \theta_{ML}) = \frac{N_2}{N} = \frac{N_2}{N_1 + N_2} = \frac{10}{25} = 0.4$$

---

Maximum a posteriori estimate

**Maximum a posteriori estimate**
- Selects the mode of the posterior distribution

$$\theta_{MAP} = \arg \max_{\theta} p(\theta \mid D, \xi)$$

**Likelihood of data**

$$p(\theta \mid D, \xi) = \frac{P(D \mid \theta, \xi)p(\theta \mid \xi)}{P(D \mid \xi)} \quad \text{(via Bayes rule)}$$

$$P(D \mid \theta, \xi) = \prod_{i=1}^{n} \theta^{x_i} (1-\theta)^{(1-x_i)} = \theta^{N_1} (1-\theta)^{N_2}$$

$p(\theta \mid \xi)$ - is the prior probability on $\theta$

**How to choose the prior probability?**
Prior distribution

Choice of prior: Beta distribution

\[ p(\theta | \xi) = \text{Beta}(\theta | \alpha_1, \alpha_2) = \frac{\Gamma(\alpha_1 + \alpha_2)}{\Gamma(\alpha_1)\Gamma(\alpha_2)} \theta^{\alpha_1-1}(1-\theta)^{\alpha_2-1} \]

\[ \Gamma(x) \text{ - A Gamma function} \]

For integer values of \( x \)
\[ \Gamma(x) = (x-1)! \]

Why to use Beta distribution?
Beta distribution “fits” Bernoulli trials - conjugate choices

\[ P(D | \theta, \xi) = \theta^{N_1}(1-\theta)^{N_2} \]

Posterior distribution is again a Beta distribution

\[ p(\theta | D, \xi) = \frac{P(D | \theta, \xi)\text{Beta}(\theta | \alpha_1, \alpha_2)}{P(D | \xi)} = \text{Beta}(\theta | \alpha_1 + N_1, \alpha_2 + N_2) \]

Beta distribution
Maximum a posterior probability

**Maximum a posteriori estimate**
- Selects the mode of the *posterior* distribution

\[
p(\theta \mid D, \xi) = \frac{P(D \mid \theta, \xi) \cdot \text{Beta}(\theta \mid \alpha_1, \alpha_2)}{P(D \mid \xi)} = \text{Beta}(\theta \mid \alpha_1 + N_1, \alpha_2 + N_2)
\]

**MAP Solution:**
\[
\theta_{MAP} = \frac{\alpha_1 + N_1 - 1}{\alpha_1 + \alpha_2 + N_1 + N_2 - 2}
\]

**Note:** that parameters of the prior

\[
p(\theta \mid \xi) = \text{Beta}(\theta \mid \alpha_1, \alpha_2) = \frac{\Gamma(\alpha_1 + \alpha_2)}{\Gamma(\alpha_1) \cdot \Gamma(\alpha_2)} \theta^{\alpha_1-1} (1-\theta)^{\alpha_2-1}
\]
- Act like counts of heads and tails
  (sometimes they are also referred to as **prior counts**)

---

MAP estimate example

- Assume the unknown and possibly biased coin
- Probability of the head is \( \theta \)
- **Data:**
  
  H H T T H H T H T T H T H H H T H H H T H T H H H T
  
  - **Heads:** 15
  - **Tails:** 10
- Assume \( p(\theta \mid \xi) = \text{Beta}(\theta \mid 5, 5) \)

What is the MAP estimate?
MAP estimate example

- Assume the unknown and possibly biased coin
- Probability of the head is $\theta$
- **Data:**
  - Heads: 15
  - Tails: 10
- Assume $p(\theta \mid \xi) = \text{Beta}(\theta \mid 5,5)$

What is the MAP estimate?

$$
\theta_{MAP} = \frac{N_1 + \alpha_1 - 1}{N - 2} = \frac{N_1 + \alpha_1 - 1}{N_1 + N_2 + \alpha_1 + \alpha_2 - 2} = \frac{19}{33}
$$

---

MAP estimate example

- Note that the prior and data fit (data likelihood) are combined
- **The MAP can be highly biased with large prior counts**
- It is hard to overturn it with a smaller sample size
- **Data:**
  - Heads: 15
  - Tails: 10
- Assume $p(\theta \mid \xi) = \text{Beta}(\theta \mid 5,5)$

$$
\theta_{MAP} = \frac{19}{33}
$$

- Assume $p(\theta \mid \xi) = \text{Beta}(\theta \mid 5,20)$

$$
\theta_{MAP} = \frac{19}{48}
$$
Multinomial distribution

Example: Multi-way coin toss, roll of dice

- Data: a set of $N$ outcomes (multi-set)
  $N_i$ - a number of times an outcome $i$ has been seen

Model parameters: $\mathbf{\theta} = (\theta_1, \theta_2, \ldots, \theta_k)$ s.t. $\sum_{i=1}^k \theta_i = 1$

$\theta_i$ - probability of an outcome $i$

Probability of data (likelihood)

$$P(N_1, N_2, \ldots, N_k \mid \mathbf{\theta}, \xi) = \frac{N!}{N_1! N_2! \ldots N_k!} \theta_1^{N_1} \theta_2^{N_2} \ldots \theta_k^{N_k}$$

ML estimate:

$$\hat{\theta}_{i,ML} = \frac{N_i}{N}$$

MAP estimate

Choice of prior: Dirichlet distribution

$$Dir(\mathbf{\theta} \mid \alpha_1, \ldots, \alpha_k) = \frac{\Gamma\left(\sum_{i=1}^k \alpha_i\right)}{\prod_{i=1}^k \Gamma(\alpha_i)} \theta_1^{\alpha_1-1} \theta_2^{\alpha_2-1} \ldots \theta_k^{\alpha_k-1}$$

Dirichlet is the conjugate choice for multinomial

$$P(D \mid \mathbf{\theta}, \xi) = P(N_1, N_2, \ldots, N_k \mid \mathbf{\theta}, \xi) = \frac{N!}{N_1! N_2! \ldots N_k!} \theta_1^{N_1} \theta_2^{N_2} \ldots \theta_k^{N_k}$$

Posterior distribution

$$p(\mathbf{\theta} \mid D, \xi) = \frac{P(D \mid \mathbf{\theta}, \xi) Dir(\mathbf{\theta} \mid \alpha_1, \alpha_2, \ldots, \alpha_k)}{P(D \mid \xi)} = Dir(\mathbf{\theta} \mid \alpha_1 + N_1, \alpha_2 + N_2, \ldots, \alpha_k + N_k)$$

MAP estimate:

$$\hat{\theta}_{i,MAP} = \frac{\alpha_i + N_i - 1}{\sum_{i=1}^k (\alpha_i + N_i) - k}$$
Gaussian (normal) distribution

- **Gaussian:**  $x \sim N(\mu, \sigma)$
- **Parameters:**  
  - $\mu$ - mean
  - $\sigma$ - standard deviation
- **Density function:**
  $$p(x | \mu, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} \exp\left[-\frac{1}{2\sigma^2} (x - \mu)^2\right]$$

Parameter estimates

- **Log-likelihood**
  $$l(D, \mu, \Sigma) = \log \prod_{i=1}^{n} p(x_i | \mu, \Sigma)$$
- **ML estimates of the mean and covariances:**
  $$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i \quad \hat{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{\mu})(x_i - \hat{\mu})^T$$

  - Covariance estimate is biased
    $$E_n(\sigma^2) = E_n\left(\frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{\mu})^2\right) = \frac{n-1}{n} \sigma^2 \neq \sigma^2$$
  - **Unbiased estimate:**
    $$\hat{\sigma}^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \hat{\mu})^2$$
Learning complex distributions

• The problem of learning complex distributions
  – can be sometimes reduced to the problem of learning a set of simpler distributions

• Such a decomposition occurs for example in Bayesian networks
  – Builds upon independences encoded in the network

• Why learning of BBNs?
  – Large databases are available
    • uncover important probabilistic dependencies from data and use them in inference tasks

Learning of BBN parameters

Learning. Two steps:
  – Learning of the network structure
  – Learning of parameters of conditional probabilities

• Variables:
  – Observable – values present in every data sample
  – Hidden – values are never in the sample
  – Missing values – values sometimes present, sometimes not

• Here:
  – learning parameters for the fixed graph structure
  – All variables are observed in the dataset
Learning of BBN parameters. Example.

Example:

\[ P(\text{Pneumonia}) \]

\[ \begin{array}{c|c|c}
\text{T} & \text{F} \\
\hline
? & ? \\
\end{array} \]

\[ P(\text{HWBC|Pneum}) \]

\[ \begin{array}{c|c|c}
\text{Pn} & \text{T} & \text{F} \\
\hline
\text{T} & ? & ? \\
\text{F} & ? & ? \\
\end{array} \]

\[ P(\text{Paleness|Pneum}) \]

\[ P(\text{Fever|Pneum}) \]

\[ P(\text{Cough|Pneum}) \]

\[ P(\text{High WBC|Pneum}) \]

CS 1571 Intro to AI

Learning of BBN parameters. Example.

Data D (different patient cases):

<table>
<thead>
<tr>
<th>Pal</th>
<th>Fev</th>
<th>Cou</th>
<th>HWB</th>
<th>Pneu</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

CS 1571 Intro to AI
Estimates of parameters of BBN

• Much like multiple coin toss or roll of a dice problems.
• A “smaller” learning problem corresponds to the learning of exactly one conditional distribution
• Example:
  \[ P(Fever \mid Pneumonia = T) \]
• Problem: How to pick the data to learn?

Estimates of parameters of BBN

Much like multiple coin toss or roll of a dice problems.
• A “smaller” learning problem corresponds to the learning of exactly one conditional distribution

Example:

\[ P(Fever \mid Pneumonia = T) \]

Problem: How to pick the data to learn?

Answer:
1. Select data points with Pneumonia=T (ignore the rest)
2. Focus on (select) only values of the random variable defining the distribution (Fever)
3. Learn the parameters of the conditional the same way as we learned the parameters of the biased coin or dice
Learning of BBN parameters. Example.

**Learn:** $P(Fever \mid Pneumonia = T)$

**Step 1:** Select data points with Pneumonia=T

<table>
<thead>
<tr>
<th>Pal</th>
<th>Fev</th>
<th>Cou</th>
<th>HWB</th>
<th>Pneu</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Learning of BBN parameters. Example.

**Learn:** $P(Fever \mid Pneumonia = T)$

**Step 1:** Ignore the rest

<table>
<thead>
<tr>
<th>Pal</th>
<th>Fev</th>
<th>Cou</th>
<th>HWB</th>
<th>Pneu</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
Learning of BBN parameters. Example.

**Learn:** \( P(Fever \mid Pneumonia = T) \)

**Step 2:** Select values of the random variable defining the distribution of Fever

<table>
<thead>
<tr>
<th>Pal</th>
<th>Fev</th>
<th>Cou</th>
<th>HWB</th>
<th>Pneu</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

Learning of BBN parameters. Example.

**Learn:** \( P(Fever \mid Pneumonia = T) \)

**Step 2:** Ignore the rest

Fever

<table>
<thead>
<tr>
<th>Fev</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
</tr>
<tr>
<td>F</td>
</tr>
<tr>
<td>T</td>
</tr>
<tr>
<td>T</td>
</tr>
<tr>
<td>T</td>
</tr>
</tbody>
</table>
Learning of BBN parameters. Example.

Learn: \( P(\text{Fever} \mid \text{Pneumonia} = T) \)

Step 3: Learning the ML estimate

\[
P_F(\text{Fever} \mid \text{Pneumonia} = T)
\]

<table>
<thead>
<tr>
<th>T</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Learning of BBN parameters. Example.

Learn: \( P(\text{Fever} \mid \text{Pneumonia} = T) \)

Step 3: Learning the MAP estimate

Assume the prior

\[
\theta_{\text{Fever}|\text{Pneumonia}=T} \sim \text{Beta}(3,4)
\]

\[
P_F(\text{Fever} \mid \text{Pneumonia} = T)
\]

<table>
<thead>
<tr>
<th>T</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>
Linear regression

Supervised learning

Data:  \( D = \{D_1, D_2, \ldots, D_n\} \)  a set of \( n \) examples
\[ D_i = \langle x_i, y_i \rangle \]
\[ x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,d}) \] 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
Linear regression

• **Function** \( f : X \rightarrow Y \) 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)

Bias term \( \rightarrow 1 \)

Input vector \( x \)

\[
\sum f(x, w)
\]

Linear regression

• **Shorter (vector) definition of the model**
  – Include bias constant in the input vector
  \( x = (1, x_0, x_2, \ldots, x_d) \)

\[
f(x) = w_0 x_0 + w_1 x_1 + w_2 x_2 + \ldots + w_d x_d = w^T x
\]

\( w_0, w_1, \ldots, w_k \) - parameters (weights)
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

Mean-squared error $J_n = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2$

- **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 \( \mathbf{x} = (x_1, x_2) \)

\[
\begin{align*}
\text{CS 1571 Intro to AI}

\text{Linear regression. Optimization.}

• We want the \textbf{weights minimizing the error}

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

• For the optimal set of parameters, derivatives of the error with respect to each parameter must be 0

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

• \textbf{Vector of derivatives:}

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

\[
\text{CS 1571 Intro to AI}
\]
Linear regression. Optimization.

- For the optimal set of parameters, derivatives of the error with respect to each parameter must be 0

\[
J_n = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2 = \frac{1}{n} \sum_{i=1}^{n} (y_i - [w_0 + w_1 x^{(1)} + w_2 x^{(2)} + \ldots w_k x^{(k)}])^2
\]

\[\nabla_w J_n(w) = 0\] defines a set of equations in \( w \)

\[
\frac{\partial}{\partial w_0} J_n(w) = -2 \sum_{i=1}^{n} [y_i - (w_0 + w_1 x^{(1)} + w_2 x^{(2)} + \ldots w_k x^{(k)})] = 0
\]

\[
\frac{\partial}{\partial w_1} J_n(w) = -2 \sum_{i=1}^{n} [y_i - (w_0 + w_1 x^{(1)} + w_2 x^{(2)} + \ldots w_k x^{(k)})] x^{(1)} = 0
\]

\[
\vdots
\]

\[
\frac{\partial}{\partial w_j} J_n(w) = -2 \sum_{i=1}^{n} [y_i - (w_0 + w_1 x^{(1)} + w_2 x^{(2)} + \ldots w_k x^{(k)})] x^{(j)} = 0
\]

\[
\vdots
\]

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

\[
A w = b
\]


cs 1571 intro to AI

Solving linear regression

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

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

\[
A w = b
\]

\[
w_0 \sum_{i=1}^{n} x_{i,0} 1 + w_1 \sum_{i=1}^{n} x_{i,1} 1 + \ldots + w_j \sum_{i=1}^{n} x_{i,j} 1 + \ldots + w_d \sum_{i=1}^{n} x_{i,d} 1 = \sum_{i=1}^{n} y_i 1
\]

\[
w_0 \sum_{i=1}^{n} x_{i,0} x_{i,1} + w_1 \sum_{i=1}^{n} x_{i,1} x_{i,1} + \ldots + w_j \sum_{i=1}^{n} x_{i,j} x_{i,1} + \ldots + w_d \sum_{i=1}^{n} x_{i,d} x_{i,1} = \sum_{i=1}^{n} y_i x_{i,1}
\]

\[
\vdots
\]

\[
w_0 \sum_{i=1}^{n} x_{i,0} x_{i,j} + w_1 \sum_{i=1}^{n} x_{i,j} x_{i,j} + \ldots + w_j \sum_{i=1}^{n} x_{i,j} x_{i,j} + \ldots + w_d \sum_{i=1}^{n} x_{i,d} x_{i,j} = \sum_{i=1}^{n} y_i x_{i,j}
\]

\[
\vdots
\]

cs 1571 intro to AI
Solving linear regression

• The optimal set of weights satisfies:

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

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

\[ A w = b \]

\[ w_0 \sum_{i=1}^{n} x_{i,0} x_{i,j} + w_1 \sum_{i=1}^{n} x_{i,1} x_{i,j} + \ldots + w_j \sum_{i=1}^{n} x_{i,j} x_{i,j} + \ldots + w_d \sum_{i=1}^{n} x_{i,d} x_{i,j} = \sum_{i=1}^{n} y_i x_{i,j} \]

Solutions to SLE:
• e.g. matrix inversion (if the matrix is singular)

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

Gradient descent solution

• There are other ways to solve the weight optimization problem in the linear regression model

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

• A simple technique:
  – Gradient descent

  Idea:
  • Adjust weights in the direction that improves the Error
  • The gradient tells us what is the right direction

\[ w \leftarrow w - \alpha \nabla_w \text{Error}_i(w) \]

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

• Descend using the gradient information

\[ \nabla_w Error(w) \]

Direction of the descent

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

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

CS 1571 Intro to AI

Gradient descent method

• New value of the parameter

\[ w_j \leftarrow w_j^* - \alpha \frac{\partial}{\partial w_j} Error(w) \bigg|_{w^*} \quad \text{For all } j \]

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

CS 1571 Intro to AI
Gradient descent method

• Iteratively converge to the optimum of the Error function

Online regression algorithm

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

• Instead of the error for all data points we use error for each example $D_i = \langle x_i, y_i \rangle$
  \[ J_{\text{online}} = \text{Error}_i(w) = \frac{1}{2} (y_i - f(x_i, w))^2 \]

• Change regression weights after every example according to the gradient:
  \[ w_j \leftarrow w_j - \alpha \frac{\partial}{\partial w_j} \text{Error}_i(w) \]
  vector form: \[ w \leftarrow w - \alpha \nabla_w \text{Error}_i(w) \]
  $\alpha > 0$ - Learning rate that depends on the number of updates
Gradient for on-line learning

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

On-line algorithm: sequence of online updates
(i)-th update for the linear model: \[ D_i = \langle x_i, y_i \rangle \]

Vector form:
\[ w^{(i)} \leftarrow w^{(i-1)} - \alpha(i) \nabla_w Error_i(w) \bigg|_{w^{(i-1)}} = w^{(i-1)} + \alpha(i)(y_i - f(x_i, w^{(i-1)}))x_i \]

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

Annealed learning rate: \[ \alpha(i) \approx \frac{1}{i} \]
- Gradually rescales changes in weights

Online regression algorithm

**Online-linear-regression** \((D, number\ of\ iterations)\)

Initialize weights \( w = (w_0, w_1, w_2 \ldots w_d) \)
for \( i=1:1\): number of iterations
  do select a data point \( D_i = (x_i, y_i) \) from \( D \)
    set \( \alpha = 1/i \)
    update weight vector
      \( w \leftarrow w + \alpha(y_i - f(x_i, w))x_i \)
  end for
return weights \( w \)

• **Advantages:** very easy to implement, continuous data streams
On-line learning. Example

CS 1571 Intro to AI