Foundations of Machine Learning

Linear Regression (contd)

August 25, 2020

Recap: Least Squares Linear Regression

Given: Training data containing n samples where each $\mathbf{x}_i \in \mathbb{R}^d$, $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^n$

Least squares solution to linear regression is as follows:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \underline{E(\mathbf{w}, \mathcal{D})} = \arg\min_{\mathbf{w}} \left\{ \sum_{i=1}^{n} \left(\sum_{j=0}^{m} w_j \phi_j(\mathbf{x}_i) - y_i \right)^2 \right\}$$

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \underline{\left(||\mathbf{\Phi}\mathbf{w} - \mathbf{y}||^2 \right)} = \underline{(\mathbf{\Phi}^T \mathbf{\Phi})^{-1} \mathbf{\Phi}^T \mathbf{y}}$$

where

$$\underset{n \times (m+1)}{\mathbf{\Phi} = \begin{bmatrix} 1 & \phi_1(\mathbf{x}_1) & \dots & \phi_m(\mathbf{x}_1) \\ \dots & \dots & \dots & \\ 1 & \phi_1(\mathbf{x}_n) & \dots & \phi_m(\mathbf{x}_n) \end{bmatrix}}, \underset{(m+1) \times 1}{\mathbf{w} = \begin{bmatrix} w_0 \\ \dots \\ w_m \end{bmatrix}}, \underset{n \times 1}{\mathbf{y} = \begin{bmatrix} y_1 \\ \dots \\ y_n \end{bmatrix}}$$

Probabilistic Modeling of Linear Regression

- From the training set $\mathcal{D} = \{\mathbf{x}_j, y_j\}_{j=1}^n$, assume each target y_j is drawn from a probabilistic model.

Probabilistic Modeling of Linear Regression

- From the training set $\mathcal{D} = \{\mathbf{x}_j, y_j\}_{j=1}^{n}$, assume each target y_j is drawn from a probabilistic model.
- Let us assume a noisy linear model parameterized by $\mathbf{w} \in \mathbb{R}^d$

$$\boxed{y_j = \mathbf{w}^T \mathbf{x}_j + \epsilon_j}$$

Each target is associated with an additive stochastic noise modeled as a random variable ϵ_j.

Probabilistic Modeling of Linear Regression

- From the training set $\mathcal{D} = \{\mathbf{x}_j, y_j\}_{j=1}^n$, assume each target y_j is drawn from a probabilistic model.
- Let us assume a noisy linear model parameterized by $\mathbf{w} \in \mathbb{R}^d$

$$\boxed{y_j = \mathbf{w}^T \mathbf{x}_j + \epsilon_j}$$

Each target is associated with an additive stochastic noise modeled as a random variable ϵ_j.

- Let us model the noise to come from a **Gaussian distribution:** $\epsilon_j \sim \mathcal{N}(\underbrace{0}_{\text{mean}}, \underbrace{\sigma^2}_{\text{variance}})$.

Probabilistic Modeling of Linear Regression

- From the training set $\mathcal{D} = \{\mathbf{x}_j, y_j\}_{j=1}^n$, assume each target y_j is drawn from a probabilistic model.
- Let us assume a noisy linear model parameterized by $\mathbf{w} \in \mathbb{R}^d$

  $$\boxed{y_j = \mathbf{w}^T \mathbf{x}_j + \epsilon_j}$$

  Each target is associated with an additive stochastic noise modeled as a random variable ϵ_j.
- Let us model the noise to come from a **Gaussian distribution:** $\epsilon_j \sim \mathcal{N}(0, \sigma^2)$.
- Noise variables ϵ_j all have the same mean (0), same variance (σ^2) and $\mathrm{Cov}(\epsilon_i, \epsilon_j) = 0$ for $i \neq j$.

Probabilistic Modeling of Linear Regression

- If $y_j = \mathbf{w}^T \mathbf{x}_j + \epsilon_j$ and $\epsilon_j \sim \mathcal{N}(0, \sigma^2)$, then each y_j is drawn from the following Gaussian:

$$\boxed{y_j \sim \mathcal{N}(\mathbf{w}^T \mathbf{x}_j, \sigma^2)}$$

Probabilistic Modeling of Linear Regression

- If $y_j = \mathbf{w}^T \mathbf{x}_j + \epsilon_j$ and $\epsilon_j \sim \mathcal{N}(0, \sigma^2)$, then each y_j is drawn from the following Gaussian:

$$\boxed{y_j \sim \mathcal{N}(\mathbf{w}^T \mathbf{x}_j, \sigma^2)}$$

- Objective: Estimate $\mathbf{w}$ from this probabilistic model. We will look at **Maximum Likelihood Estimation (MLE)** in this lecture.

MLE, more generally

A set of independent and identically distributed (i.i.d.) observations $\mathbf{y} = \{y_1, \ldots, y_n\}$ are generated according to a probability model parameterized by θ, i.e. $y_j \sim \Pr(\mathbf{y}|\theta)$

MLE, more generally

A set of independent and identically distributed (i.i.d.) observations $\mathbf{y} = \{y_1, \ldots, y_n\}$ are generated according to a probability model parameterized by θ, i.e. $y_j \sim \Pr(\mathbf{y}|\theta)$

What is $\Pr(\mathbf{y}|\theta)$ (also referred to as the *likelihood*)?

MLE, more generally

A set of independent and identically distributed (i.i.d.) observations $\mathbf{y} = \{y_1, \ldots, y_n\}$ are generated according to a probability model parameterized by θ, i.e. $y_j \sim \Pr(\mathbf{y}|\theta)$

What is $\Pr(\mathbf{y}|\theta)$ (also referred to as the *likelihood*)?

Find the parameters θ. What's one good way? Find θ s.t. it maximizes the likelihood of the observed data. Hence, **MLE**.

MLE contd

- Typically maximize log-likelihood i.e. $\log(\Pr(\mathbf{y}|\theta))$ instead of the likelihood. Would this affect the estimated parameters?

MLE contd

- Typically maximize log-likelihood i.e. $\log(\Pr(\mathbf{y}|\theta))$ instead of the likelihood. Would this affect the estimated parameters?

- Why apply this log transformation? Mathematical convenience
  Numerical convenience

MLE contd

- Typically maximize log-likelihood i.e. $\log(\Pr(\mathbf{y}|\theta))$ instead of the likelihood. Would this affect the estimated parameters?

- Why apply this log transformation?

$Pr(y_j ; \theta)$ or $P_\theta(y_j)$

- What is the log-likelihood?

$$\mathcal{L}(\theta) = \log(\Pr(\mathbf{y}|\theta)) = \log(\prod_{j=1}^{n} \Pr(y_j|\theta)) = \sum_{j=1}^{n} \log(\Pr(y_j|\theta))$$

- The MLE estimate is given by:

$$\theta_{\text{MLE}} = \underset{\theta}{\operatorname{argmax}}\, \underline{\mathcal{L}(\theta)} = \underset{\theta}{\operatorname{argmax}} \sum_{j=1}^{n} \log(\Pr(y_j|\theta))$$

Illustrating MLE using a simple coin toss example

Say we toss a coin N times. Each coin toss y_j is a binary random variable with a Bernoulli distribution $\Pr(y_j|\theta) = \underline{\theta}^{y_j}(1-\theta)^{1-y_j}$. Estimate θ using MLE.

$$L(\theta) = \log \prod_{j=1}^{N} P(y_j|\theta) = \sum_j \log P(y_j|\theta)$$

$$= \sum_j y_j \log \theta + (1-y_j)\log(1-\theta)$$

$$\frac{\partial L(\theta)}{\partial \theta} = 0 \Rightarrow \sum_j \frac{y_j}{\theta} - \frac{(1-y_j)}{1-\theta} = 0 \Rightarrow \boxed{\theta = \frac{\sum_j y_j}{N}}$$

Coming back to MLE for linear regression

Assuming that each target y_j is Gaussian-distributed, i.e.

$$\Pr(y_j|x_j, \mathbf{w}) = \mathcal{N}(\mathbf{w}^T x_j, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{\frac{-(y_j - \mathbf{w}^T x_j)^2}{2\sigma^2}\right\}$$

Likelihood of the data $\mathcal{L}(\mathbf{w}) = \Pr(\mathbf{y}|\mathbf{X}, \mathbf{w})$

Coming back to MLE for linear regression

Assuming that each target y_j is Gaussian-distributed, i.e.

$$\Pr(y_j|x_j,\mathbf{w}) = \mathcal{N}(\mathbf{w}^T x_j, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{\frac{-(y_j - \mathbf{w}^T x_j)^2}{2\sigma^2}\right\}$$

Likelihood of the data $\mathcal{L}(\mathbf{w}) = \Pr(\mathbf{y}|\mathbf{X},\mathbf{w})$

$$\mathcal{L}(w) = \log \prod_{j=1}^{n} P(y_j|x_j, w) = \sum_j \log P(y_j|x_j, w)$$

MLE estimate: $\mathbf{w}_{\mathrm{MLE}} = \underset{\mathbf{w}}{\operatorname{argmax}}\, \mathcal{L}(\mathbf{w})$

MLE estimate for probabilistic linear regression

To estimate $\mathbf{w}_{\text{MLE}}$, consider log-likelihood ($\log(\mathcal{L}(\mathbf{w}))$) instead of likelihood ($\mathcal{L}(\mathbf{w})$):

$\log(\mathcal{L}(\mathbf{w})) = LL(\mathbf{w})$

MLE estimate for probabilistic linear regression

To estimate $\mathbf{w}_{\text{MLE}}$, consider log-likelihood ($\log(\mathcal{L}(\mathbf{w}))$) instead of likelihood ($\mathcal{L}(\mathbf{w})$):

$\log(\mathcal{L}(\mathbf{w})) = LL(\mathbf{w})$

$$LL(w) = \sum_j \log P(y_j \mid x_j, w)$$

$$= \text{constant} - \sum_j \frac{(y_j - w^T x_j)^2}{2\sigma^2}$$

$$\mathbf{w}_{\text{MLE}} = \underset{\mathbf{w}}{\text{argmax}}\, LL(\mathbf{w}) = \underset{\mathbf{w}}{\arg\min} \sum_{j=1}^{n} (y_j - \mathbf{w}^T x_j)^2$$ **(Same as least squares!)**

Estimating $\mathbf{w}$

Instead of finding the least squares/MLE solution analytically, use a search algorithm that progressively decreases the error function $E(\mathbf{w}, \mathcal{D})$.

- Initialize $\mathbf{w}$
- Until we converge:

  Find a new value of $\mathbf{w}$ that reduces $E(\mathbf{w}, \mathcal{D})$

$$w_{t+1} \leftarrow w_t + \boxed{}$$

Estimating $\mathbf{w}$

Instead of finding the least squares/MLE solution analytically, use a search algorithm that progressively decreases the error function $E(\mathbf{w}, \mathcal{D})$.

- Initialize $\mathbf{w}$
- Until we converge:

  Find a new value of $\mathbf{w}$ that reduces $E(\mathbf{w}, \mathcal{D})$

Gradient descent is an iterative algorithm used to find the minimum of a function.

Gradient Descent

$$w = \begin{bmatrix} w_1 \\ \vdots \\ w_i \\ \vdots \\ w_d \end{bmatrix}$$

Initialize **w** $\leftarrow \vec{0}$ vector

Repeat until convergence:

$\|\nabla_w E(w)\|^2 \leq \epsilon$

$\| E(w_{t+1}) - E(w_t) \|_2 \leq \epsilon$

$w_i \leftarrow w_i - \eta \frac{\partial E(\mathbf{w})}{\partial w_i}$ (Simultaneously update each w_i)

Compute $\frac{\partial E(\mathbf{w})}{\partial w_i}$

LEARNING RATE

$$w \leftarrow w - \eta \nabla_w E(w)$$

Using Gradient Descent (GD)

$\sum_i E_i(w)$

Weight update rule in GD: $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} E(\mathbf{w})$

Faster version: **Stochastic Gradient Descent (SGD)**

The weight update rule for SGD: $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} E(\mathbf{w}; \mathbf{x}_i, y_i)$ where the loss is computed for a single example randomly sampled from the training set.

Middle ground: **Mini-batch Gradient Descent**

The weight update rule becomes: $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} E_B(\mathbf{w}; \{\mathbf{x}_i, y_i\}_{i=1}^{|B|})$ where the loss is computed for a batch of $|B|$ examples sampled from the training set.