Linear Models for Regression

CS534
Example Regression Problems

• Predict housing price based on
  – House size, lot size, Location, # of rooms ...

• Predict stock price based on
  – Price history of the past month ...

• Predict the abundance of a species based on
  – Environmental conditions
General Set Up

**Given:** a training set, containing a total of $N$ training examples $(x_i, y_i)$, for $i = 1, \ldots, N$

**Goal:** learn a function $\hat{y}$ to minimize some loss function on the training data

To begin, we consider a linear hypothesis space (thus the name linear regression):

Let $\mathbf{x} = [1, x_1, x_2, \ldots, x_d]^T$

$$\hat{y}(\mathbf{x}) = w_0 + w_1 x_1 + \cdots + w_d x_d = \mathbf{w}^T \mathbf{x}$$

where $\mathbf{w} = [w_0, w_1, \ldots, w_d]^T$
Loss Function: Sum-Squared Error

• Given a set of training examples
  \((x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\)

• Hypothesis space:
  \(\hat{y}(x) = w^T x\)

• Loss function:
  \[E(w) = \frac{1}{2} \sum_{i=1}^{N} (w^T x_i - y_i)^2\]
Optimization of the loss function

• There are many ways one could use to optimize this objective function
• One simple approach is called gradient descent
  – Start with some random guess of the parameter
  – Improve the parameter by following the steepest descent direction
Gradient descent to optimize $f(\mathbf{w})$

1. Start from some initial guess $\mathbf{w}^0$

2. Find the direction of steepest descent – opposite of the gradient direction $\nabla f(\mathbf{w})$

3. Take a step toward that direction
   $$\mathbf{w}^{t+1} = \mathbf{w}^t - \nabla f(\mathbf{w}^t)$$

4. Repeat until no local improvement is possible
   ($|\nabla f(\mathbf{w}^t)| \leq \epsilon$)

Different starting points may lead to different local minimum if objective is not convex (like what’s shown in the above figure)
Gradient Descent for SSE

We need to compute the gradient of \( E(w) \):

\[
\nabla E(w) = \left[ \frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_d} \right]^T
\]

\[
\frac{\partial E}{\partial w_k} = \sum_{i=1}^{N} (w^T x_i - y_i)x_{ik}
\]

\[
\nabla E(w) = \sum_{i=1}^{N} (w^T x_i - y_i)x_i = \sum_{i=1}^{N} (\hat{y}_i - y_i)x_i
\]
Batch Gradient Descent for SSE

\[ w = w^0 \]

Repeat{
\[ \nabla E(w) = \sum_{i=1}^{N} (w^T x_i - y_i) x_i \]
\[ w \leftarrow w - \lambda \nabla E(w) \]
} until \( |\nabla E(w)| \leq \epsilon \)

- The update rule goes through all training examples before taking each update step – thus called batch gradient descent
  - Examples with larger error \((w^T x_i - y_i)\) contribute more to the update
- \(\lambda\): learning rate or step size
  - Large \(\lambda\) makes faster progress, but could lead to oscillation
  - Small \(\lambda\) avoids overstepping, but converges slower
- \(\epsilon\): convergence criterion
- SSE Objective is convex – converge to global optimal
Stochastic gradient descent

\[
\mathbf{w} = \mathbf{w}^0 \\
\text{Repeat until convergence} \{ \\
\quad \text{for } i = 1 \text{ to } N \quad \mathbf{w} \leftarrow \mathbf{w} - \lambda (\mathbf{w}^T \mathbf{x}_i - y_i) \mathbf{x}_i \\
\}\]

- It repeatedly runs through the training examples
- It updates \( \mathbf{w} \) immediately when seeing an example, based on the gradient w.r.t. that training instance
- Comparing batch vs. stochastic gradient descent:
  - Batch gradient descent needs to go through all training examples for each update step – not appropriate for large datasets
  - Each step of Stochastic gradient descent can see objective oscillating
  - For large datasets, Stochastic gradient descent approaches convergence much faster
Alternative way to optimizing $E(w)$

- We can represent $E(w)$ using vector form
- Define

$$X = \begin{bmatrix} x_{10} & x_{11} & \cdots & x_{1d} \\ x_{20} & x_{21} & \cdots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n0} & x_{n1} & \cdots & x_{nd} \end{bmatrix}, \quad Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

$$E(w) = \frac{1}{2} |Xw - Y|^2 = \frac{1}{2} (Xw - Y)^T (Xw - Y)$$

$$\nabla E(w) = X^T (Xw - Y) = 0$$

$$X^T X w - X^T Y = 0$$

$$X^T X w = X^T Y \quad \Rightarrow \quad w = (X^T X)^{-1} X^T Y$$

Normal equation
Probabilistic Interpretation of SSE

• One might ask, why is SSE a good objective?

• There is a natural probabilistic explanation for SSE with some simple probabilistic assumptions

• Before that, we will provide a brief of maximum likelihood estimation
Background: Maximum Likelihood Estimation
Parameter Estimation

• Given observation of some random variable(s), assuming the variable(s) follow some fixed distribution, how to estimate the parameters?

• Example:
  – coin tosses (Bernoulli distribution, parameter: $p$ – probability of head)
  – Dice or grades (Discrete distribution, parameters: $p_i$ for $i = 1, \ldots, k - 1$ with $k$ categories)
  – Height and weight of a person (continuous variable, parameters depends on the distribution, for example for Gaussian Distribution, the parameters are $\mu$, the mean and $\Sigma$, the covariance)
Maximum Likelihood Principle

• We will use a general term $M$ to denote the model (parameters) we try to estimate

• We will use a general term $D$ to denote the data that we observe
  – $D$ is a set of examples, let $D_m$ denote the $m$-th example in $D$

• Assuming a fixed model $M$, what is the probability of observing $D$?
  – This can be written as $P(D; M)$

• The maximum likelihood principle seeks to find $M$ that maximizes this probability

• This is called the likelihood function
  – $L(M) = P(D; M) = \prod_{m=1}^{n} P(D_m; M)$ --- here we are making the IID assumption, that is examples in $D$ are independently and identically distributed, thus we can use the product

• It is often more convenient to work with the log of $L(M)$
  – $l(M) = \log L(M) = \sum_{m=1}^{n} \log P(D_m; M)$
Example: Coin Toss

- You are given a coin, and want to decide $p$ - the probability of head of this coin
- You toss it for 500 times, and observe heads 242 times
- What is your estimate of $p$?

$$p = \frac{242}{500} = 0.484$$

- But why? – this is doing maximum likelihood estimation
- In other words, $p = 0.484$ leads to the highest probability of observing 242 heads out of 500 tosses among all possible values of $p$
- Now let’s go over the math to convince ourselves
Example Cont.

- We have $D = \{x_1, \ldots, x_n\}$, where $x_m$ is the outcome of the $m$-th coin toss,
  - 1 means head, and 0 means tail
- Let's write down the likelihood function: $L(p) = P(D; p) = \prod_{m=1}^{n} P(x_m; p)$
- For binary variable, we have a nice compact form to write down its probability mass function
  $$P(x_m; p) = p^{x_m} (1 - p)^{1-x_m}$$
- So we have
  $$L(p) = \prod_{m=1}^{n} p^{x_m} (1 - p)^{1-x_m}$$
- Take the log:
  $$l(p) = \log L(p) = \sum_{m=1}^{n} \log p^{x_m} (1 - p)^{1-x_m}$$
  $$= \sum_{m=1}^{n} x_m \log p + \sum_{m=1}^{n} (1 - x_m) \log(1 - p) = n_1 \log p + n_0 \log(1 - p)$$
  $n_1 = \sum_{m=1}^{n} x_m$ and $n_0 = n - n_1$
Maximizing the likelihood

• We take derivative of \( l(p) \):

\[
\frac{dl(p)}{dp} = \frac{n_1}{p} - \frac{n_0}{1 - p}
\]

• Setting it to zero:

\[
\frac{n_1}{p} = \frac{n_0}{1 - p}
\]

• Solving this leads to:

\[
p = \frac{n_1}{n}
\]
Back to Linear Regression

- **Assumption**: target variable $y$ and the input $x$ are related as follows:
  \[ y = w^T x + \epsilon \]
  where $\epsilon \sim N(0, \sigma^2)$, i.e., it is Gaussian noise

- Equivalently we have: \( y|x \sim N(w^T x, \sigma^2) \)

- Given a set of observed $x$ and $y$ pairs: $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$, independently and identically distributed (IID), use maximum likelihood estimation to estimate $w$

- **Likelihood function**: given the inputs $x_1, x_2, \ldots, x_N$, what is the probability of observing target outputs $y_1, y_2, \ldots, y_N$?

  \[ L(w) = p(y_1, \ldots, y_N | x_1, \ldots, x_N; w) = \prod_{i=1}^{N} p(y_i | x_i; w) \]
Maximum likelihood estimation of $\mathbf{w}$

We will now work with log of the likelihood:

$$l(w) = \log L(w) = \log \prod_{i=1}^{N} p(y_i|\mathbf{x}_i; \mathbf{w})$$

$$= \sum_{i=1}^{N} \log \frac{1}{\sqrt{2\pi\sigma}} \exp \left[ \frac{-(y_i - \mathbf{w}^T \mathbf{x}_i)^2}{2\sigma^2} \right]$$

$$= -N \log \sqrt{2\pi\sigma} + \sum_{i=1}^{N} \frac{-(y_i - \mathbf{w}^T \mathbf{x}_i)^2}{2\sigma^2}$$

$$\arg \max_{\mathbf{w}} l(\mathbf{w}) = \arg \min_{\mathbf{w}} \frac{1}{2} \sum_{i=1}^{N} (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

Maximum Likelihood Estimation under Gaussian noise model leads to Sum of Squared Error objective
What we have seen so far

• We introduced linear regression
  – Goal: learn a linear function $\hat{y}(x) = w^T x$
  – Approach:
    • Choose a loss function: SSE
    • Optimize the loss function (Gradient descent, or analytically solving the optimization problem)

• What if the relationship between $x$ and $y$ is not linear? For example, quadratic or cubic?
  – Linear regression can still be used
  – All we need is to introduce nonlinear features
• Revisit example: Polynomial Curve Fitting

\[ y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j \]

• In this example, there is only one feature \( x \). We learn a function of \( M \)-order polynomial

• Alternatively, we could also view this as learning a linear model considering \((1, x, x^2, \ldots, x^M)\) as the features.

• Note that this new feature space is derived from the original input \( x \)

• We refer to such derived features as the basis functions
Over-fitting issue

- What can we do to curb over-fitting
  - Use less complex model
  - Use more training examples
  - Regularization
Regularized Linear Regression

• Consider the following loss function:

\[ E_D(w) + \lambda E_W(w) \]

Data term + Regularization term (penalize complex models)

• With the SSE loss and a **quadratic regularizer**, we get

\[ \frac{1}{2} \sum_{i=1}^{N} (y_i - w^T x_i)^2 + \frac{\lambda}{2} w^T w \]

Encourage small weight values

• which is minimized by

\[ w = (\lambda I + X^T X)^{-1} X^T Y \]

• \( \lambda \): regularization coefficient, which controls the trade-off between model complexity and the fit to the data
  
  – Larger \( \lambda \) encourages simple model (driving more elements of \( w \) to 0)
  
  – Small \( \lambda \) encourages better fit of the data (driving SSE to zero)
Regularized Linear Regression

- We could replace the quadratic regularizer with a more general one:

\[ \frac{1}{2} \sum_{i=1}^{N} (y_i - w^T x_i)^2 + \frac{\lambda}{2} \sum_{j=0}^{M} |w_j|^q \]

This is equivalent to minimizing SSE subject to the constraint that \( \sum_{i=0}^{M} |w_i|^q \leq有些常数 \)
Regularized Linear Regression

- Lasso ($q = 1$) tends to generate sparser solutions (majority of the weights shrink to zero) than a quadratic regularizer ($q = 2$, often called ridge regression).
Summary

• Linear regression with SSE objective
  – Gradient descent algorithm for optimization
  – Closed-form solution by solving the normal equation
  – Can be viewed as maximum likelihood estimation assuming IID Gaussian noise with zero mean
• Using basis functions allows us to learn nonlinear functions
• Controlling over-fitting by Regularization
  – Quadratic regularization is equivalent to adding a diagonal matrix to $X^TX \rightarrow (\lambda I + X^TX)$
  – LASSO regularization encourages sparse solution