15-884/15-484 – Machine Learning 1: Linear Regression

J. Zico Kolter

September 10, 2013
Motivation

• How much energy will we consume tomorrow?
  – Difficult to estimate from “a priori” models
  – But, we have lots of data from which to build a model
Energy 101

• Energy: “ability to do work” (apply force through a distance)

• Unit of energy: joule (J), (also btu, kilowatt hour)

\[
1 \text{ joule} = 1 \text{ newton} \cdot 1 \text{ meter} \\
= \frac{1 \text{ kilogram} \cdot 1 \text{ meter}^2}{1 \text{ second}^2}
\]

• Power is a *rate* of energy use

• Unit of power: watt (W) (commonly kilo/mega/gigawatt)

\[
1 \text{ watt} = \frac{1 \text{ joule}}{1 \text{ second}}
\]
Forms of energy

• Mechanical kinetic energy: \( E = \frac{1}{2}mv^2 \)

• Gravitational potential energy: \( E = mgh \)

• Thermal energy: \( E = \frac{3}{2} NkT \)

• Electrical energy: \( E = VQ \)

• Electromagnetic energy: \( E = hf \)

• Chemical energy

• Nuclear energy: \( E = mc^2 \)
Duquesne Light electricity consumption

Data: PJM [http://www.pjm.com](http://www.pjm.com)
Duquesne Light electricity consumption

Facts About Duquesne Light
Service Area 817 square miles
Allegheny and Beaver Counties
# of Customers 584,000
(Approx. 90% residential customers)

Source: Duquesne Light [http://www.duquesnelight.com](http://www.duquesnelight.com)
Predict peak demand from high temperature

- What will peak demand be tomorrow?

- If we know something else about tomorrow (like the high temperature), we can use this to *predict* peak demand

Data: PJM, Weather Underground (summer months, June-August)
A simple model

• A linear model that predicts demand:

\[
\text{predicted peak demand} = \theta_1 \cdot (\text{high temperature}) + \theta_2
\]

• Parameters of model: \( \theta_1, \theta_2 \in \mathbb{R} \) (\( \theta_1 = 0.046, \theta_2 = -1.46 \))
A simple model

- We can use a model like this to make predictions

- What will be the peak demand for Duquense Light tomorrow?
  - I know from weather report that high temperature will be 80°F (ignore, for the moment, that this too is a prediction)

- Then predicted peak demand is:

  $$\theta_1 \cdot 80 + \theta_2 = 0.046 \cdot 80 - 1.46 = 2.19 \text{ GW}$$
Formal problem setting

• **Input**: $x_i \in \mathbb{R}^n$, $i = 1, \ldots, m$
  - E.g.: $x_i \in \mathbb{R}^1 = \{\text{high temperature for day } i\}$

• **Output**: $y_i \in \mathbb{R}$ (*regression* task)
  - E.g.: $y_i \in \mathbb{R} = \{\text{peak demand for day } i\}$

• **Model Parameters**: $\theta \in \mathbb{R}^k$

• **Predicted Output**: $\hat{y}_i \in \mathbb{R}$
  
  E.g.: $\hat{y}_i = \theta_1 \cdot x_i + \theta_2$
• For convenience, we define a function that maps inputs to *feature vectors*

\[ \phi : \mathbb{R}^n \to \mathbb{R}^k \]

• For example, in our task above, if we define

\[ \phi(x_i) = \begin{bmatrix} x_i \\ 1 \end{bmatrix} \quad (\text{here } n = 1, \ k = 2) \]

then we can write

\[ \hat{y}_i = \sum_{j=1}^{k} \theta_j \cdot \phi_j(x_i) \equiv \theta^T \phi(x_i) \]
### Loss functions

- Want a model that performs “well” on the data we have

  \[ \hat{y}_i \approx y_i, \quad \forall i \]

- We measure “closeness” of \( \hat{y}_i \) and \( y_i \) using **loss function**

  \[ \ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+ \]

- Example: squared loss

  \[ \ell(\hat{y}_i, y_i) = (\hat{y}_i - y_i)^2 \]
Finding model parameters, and optimization

- Want to find model parameters such that minimize sum of costs over all input/output pairs

\[ J(\theta) = \sum_{i=1}^{m} \ell(\hat{y}_i, y_i) = \sum_{i=1}^{m} (\theta^T \phi(x_i) - y_i)^2 \]

- Write our objective formally as

\[ \text{minimize } J(\theta) \]

simple example of an optimization problem; these will dominate our development of algorithms throughout the course
• Let’s write $J(\theta)$ a little more compactly using matrix notation; define

$$\Phi \in \mathbb{R}^{m \times k} = \begin{bmatrix}
- & \phi(x_1)^T & - \\
- & \phi(x_2)^T & - \\
& \vdots & \\
- & \phi(x_m)^T & -
\end{bmatrix}, \quad y \in \mathbb{R}^m = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

then

$$J(\theta) = \sum_{i=1}^{m} (\theta^T \phi(x_i) - y_i)^2 = \|\Phi \theta - y\|_2^2$$

($\|z\|_2$ is $\ell_2$ norm of a vector: $\|z\|_2 \equiv \sqrt{\sum_{i=1}^{n} z_i^2} = \sqrt{z^T z}$)

• Called least-squares objective function
How do we optimize a function? 1-D case ($\theta \in \mathbb{R}$):

$J(\theta) = \theta^2 - 2\theta - 1$

$\frac{dJ}{d\theta} = 2\theta - 2$

$\theta^* \text{ minimum } \iff \frac{dJ}{d\theta} \bigg|_{\theta^*} = 0$

$\iff 2\theta^* - 2 = 0$

$\iff \theta^* = 1$
• Multi-variate case: \( \theta \in \mathbb{R}^k, \, J : \mathbb{R}^k \rightarrow \mathbb{R} \)

Generalized condition: \( \nabla_{\theta} J(\theta)|_{\theta^*} = 0 \)

• \( \nabla_{\theta} J(\theta) \) denotes gradient of \( J \) with respect to \( \theta \)

\[
\nabla_{\theta} J(\theta) \in \mathbb{R}^n \equiv \begin{bmatrix}
\frac{\partial J}{\partial \theta_1} \\
\frac{\partial J}{\partial \theta_2} \\
\vdots \\
\frac{\partial J}{\partial \theta_k}
\end{bmatrix}
\]

• Some important rules and common gradient

\[
\nabla_{\theta}(af(\theta) + bg(\theta)) = a \nabla_{\theta} f(\theta) + b \nabla_{\theta} g(\theta), \quad (a, b \in \mathbb{R})
\]

\[
\nabla_{\theta}(\theta^T A \theta) = (A + A^T)\theta, \quad (A \in \mathbb{R}^{k \times k})
\]

\[
\nabla_{\theta}(b^T \theta) = b, \quad (b \in \mathbb{R}^k)
\]
• Optimizing least-squares objective

\[ J(\theta) = \| \Phi \theta - y \|^2 \]

\[ = (\Phi \theta - y)^T (\Phi \theta - y) \]

\[ = \theta^T \Phi^T \Phi \theta - 2y^T \Phi \theta + y^T y \]

• Using the previous gradient rules

\[ \nabla_{\theta} J(\theta) = \nabla_{\theta}(\theta^T \Phi^T \Phi \theta - 2y^T \Phi \theta + y^T y) \]

\[ = \nabla_{\theta}(\theta^T \Phi^T \Phi \theta) - 2 \nabla_{\theta}(y^T \Phi \theta) + \nabla_{\theta}(y^T y) \]

\[ = 2\Phi^T \Phi \theta - 2\Phi^T y \]

• Setting gradient equal to zero

\[ 2\Phi^T \Phi \theta^* - 2\Phi^T y = 0 \iff \theta^* = (\Phi^T \Phi)^{-1} \Phi^T y \]

known as the normal equations
• Let’s see how this looks in MATLAB code

```matlab
X = load('high_temperature.txt');
y = load('peak_demand.txt');
n = size(X,2);
m = size(X,1);
Phi = [X ones(m,1)];
theta = inv(Phi' * Phi) * Phi' * y;

theta =
    0.0466
   -1.4600
```

• The normal equations are so common that MATLAB has a special operation for them

```matlab
% same as inv(Phi' * Phi) * Phi' * y
theta = Phi \ y;
```
Aside: general optimization problems

• In this class we’ll consider general optimization problems

\[
\begin{align*}
\text{minimize} & \quad J(\theta) \\
\text{subject to} & \quad g_i(\theta) \leq 0, \quad i = 1, \ldots, N_i \\
& \quad h_i(\theta) = 0, \quad i = 1, \ldots, N_e
\end{align*}
\]

A constrained optimization problem; \( g_i \) terms are the inequality constraints; \( h_i \) terms are the equality constraints.

• Many different classifications of optimization problems (linear programming, quadratic programming, semidefinite programming, integer programming), depending on the form of \( J \), the \( g_i \)'s and the \( h_i \)'s.
• Important distinctions in optimization is between \textit{convex} (where $J$, $g_i$ are convex and $h_i$ linear) and \textit{non-convex} problems

$$f \text{ convex } \iff f(a\theta + (1-a)\theta') \leq af(\theta) + (1-a)f(\theta')$$

for $0 \leq a \leq 1$

• Informally speaking, we can usually find \textit{global} solutions of convex problems efficiently, while for non-convex problems we must settle for \textit{local} solutions or time-consuming optimization
Solving optimization problems

- Many generic optimization libraries

- We will be using YALMIP (Yet Another Linear Matrix Inequality Parser): http://users.isy.liu.se/johanl/yalmip/

- YALMIP code for least squares optimization:

```matlab
theta = sdpvar(n,1);
solvesdp([], sum((Phi*theta - y).^2));
double(theta)

ans =
  0.0466
-1.4600
```
Alternative loss functions

- Nothing special about least-squares loss function
  \[ \ell(\hat{y}, y) = (\hat{y} - y)^2. \]

- Some alternatives:
  
  **Absolute loss:**
  \[ \ell(\hat{y}, y) = |\hat{y} - y| \]
  
  **Deadband loss:**
  \[ \ell(\hat{y}, y) = \max\{0, |\hat{y} - y| - \epsilon\}, \ \epsilon \in \mathbb{R}_+ \]
• How do we find parameters that minimize absolute loss?

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{m} |\theta^T \phi(x_i) - y_i| \\
\end{align*}
\]

– Non-differentiable, can’t take gradient

• Solution: frame as a \textit{constrained optimization problem}
  
  – Introduce new variables \( \nu \in \mathbb{R}^m, (\nu_i \geq |\theta^T \phi(x_i) - y_i|) \)

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{m} \nu_i \\
\text{subject to} & \quad -\nu_i \leq \theta^T \phi(x_i) - y_i \leq \nu_i
\end{align*}
\]

• \textit{Linear program} (LP): linear object and linear constraints
To solve LPs, typically need to put them in standard form:

\[
\begin{align*}
\text{minimize} & \quad c^T z \\
\text{subject to} & \quad Az \leq b
\end{align*}
\]

- \( z \in \mathbb{R}^n \), \( A \in \mathbb{R}^{N_i \times n} \), \( b \in \mathbb{R}^{N_i} \)

For absolute loss LP

\[
\begin{align*}
z &= \begin{bmatrix} \theta \\ \nu \end{bmatrix}, \quad c &= \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad A &= \begin{bmatrix} \Phi & -I \\ -\Phi & -I \end{bmatrix}, \quad b &= \begin{bmatrix} y \\ -y \end{bmatrix}
\end{align*}
\]
• MATLAB code

c = [zeros(n,1); ones(m,1)];
A = [Phi -eye(m); -Phi -eye(m)];
b = [y; -y];
z = linprog(c,A,b);
theta = z(1:n)

theta =
  0.0477
 -1.5978

• The same solution in YALMIP:

theta = sdpvar(n,1);
solvesdp([], sum(abs((Phi*theta - y))));
double(theta)

theta =
  0.0477
 -1.5978
• Which loss function should we use?
Higher-dimensional inputs

- Input: \( x \in \mathbb{R}^2 = \begin{bmatrix} \text{temperature} \\ \text{hour of day} \end{bmatrix} \)
- Output: \( y \in \mathbb{R} = \text{demand} \)
• Features: $\phi(x) \in \mathbb{R}^3 = \begin{bmatrix} \text{temperature} \\ \text{hour of day} \\ 1 \end{bmatrix}$

• Same matrices as before

$$\Phi \in \mathbb{R}^{m \times k} = \begin{bmatrix} \ldots & \phi(x_1)^T & \ldots \\ \vdots & \vdots & \vdots \\ \ldots & \phi(x_m)^T & \ldots \end{bmatrix}, \quad y \in \mathbb{R}^m = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$$

• Same solution as before

$$\theta \in \mathbb{R}^3 = (\Phi^T \Phi)^{-1} \Phi^T y$$