Machine Learning 1: Linear Regression

Stefano Ermon

March 31, 2016

Plan for today

Plan for today:

- Supervised Machine Learning: linear regression

Renewable electricity generation in the U.S

	Hydropower	Solar[1]	Wind	Geothermal	Biomass	Total Renewables
2004	6.7%	0.0%	0.4%	0.4%	1.3%	8.8%
2005	6.7%	0.0%	0.4%	0.4%	1.3%	8.8%
2006	7.1%	0.0%	0.7%	0.4%	1.3%	9.5%
2007	5.9%	0.0%	0.8%	0.4%	1.3%	8.5%
2008	6.2%	0.1%	1.3%	0.4%	1.3%	9.3%
2009	6.9%	0.1%	1.9%	0.4%	1.4%	10.6%
2010	6.3%	0.1%	2.3%	0.4%	1.4%	10.4%
2011	7.8%	0.2%	2.9%	0.4%	1.4%	12.6%
2012	6.8%	0.3%	3.4%	0.4%	1.4%	12.4%
2013	6.6%	0.5%	4.1%	0.4%	1.5%	13.1%
2014	6.3%	0.8%	4.4%	0.4%	1.6%	13.5%

Source: Renewable energy data book, NREL

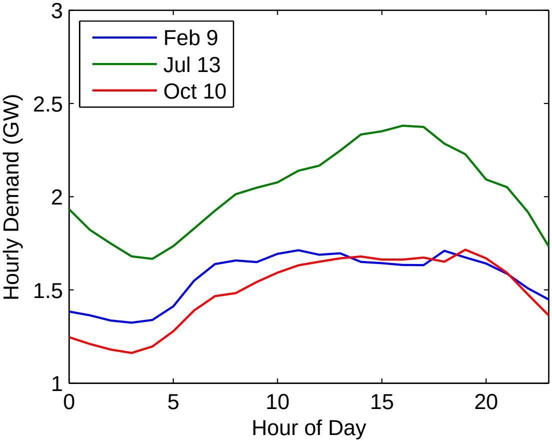
Challenges for the grid

- Wind and solar are intermittent
- We will need traditional power plants when the wind stops
  - Many power plants (e.g., nuclear) cannot be easily turned on/off or quickly ramped up/down
- With more accurate forecasts, wind and solar power become more efficient alternatives
  - A few years ago, Xcel Energy (Colorado) ran ads opposing a proposal that it use 10% of renewable sources
  - Thanks to wind forecasting (ML) algorithms developed at NCAR, they now aim for 30 percent. Accurate forecasting saved the utility \$6-\$10 million per year

Motivation

- Solar and wind are intermittent
- Can we accurately forecast 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

Typical electricity consumption

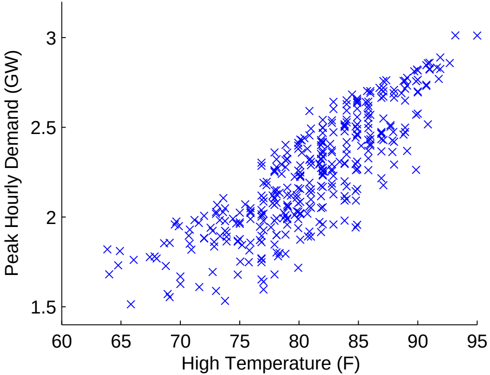


Data: PJM `http://www.pjm.com`

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

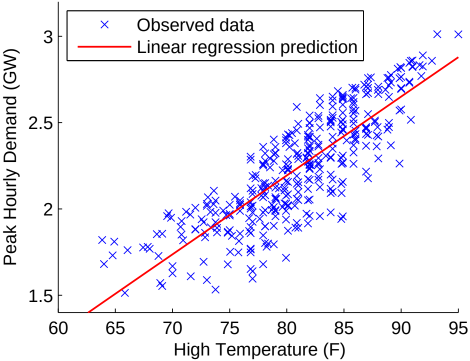


Data: PJM, Weather Underground (summer months, June-August)

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 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.: $x_i \in \mathbb{R}^1 = \{$high temperature for day $i\}$
- **Output**: $y_i \in \mathbb{R}$ (*regression* task)
  - E.g.: $y_i \in \mathbb{R} = \{$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

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

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

  I.e., $\hat{y}_i \approx y_i, \;\; \forall i$

- 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

$$\underset{\theta}{\text{minimize}} \quad J(\theta)$$

simple example of an *optimization problem*; these will dominate our development of algorithms throughout the course

How do we optimize a function

- Search algorithm: Start with an initial guess for θ. Keep changing θ (by a little bit) to reduce $J(\theta)$

- Animation `https://www.youtube.com/watch?v=vWFjqgb-ylQ`

Gradient descent

- Search algorithm: Start with an initial guess for θ. Keep changing θ (by a little bit) to reduce $J(\theta)$

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

- Gradient descent: $\theta_j = \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}$, for all j

$$\frac{\partial J}{\partial \theta_j} = \frac{\partial \sum_{i=1}^{m} (\theta^T \phi(x_i) - y_i)^2}{\partial \theta_j} = \sum_{i=1}^{m} \frac{\partial (\theta^T \phi(x_i) - y_i)^2}{\partial \theta_j}$$

$$= \sum_{i=1}^{m} 2(\theta^T \phi(x_i) - y_i) \frac{\partial (\theta^T \phi(x_i) - y_i)}{\partial \theta_j}$$

$$= \sum_{i=1}^{m} 2(\theta^T \phi(x_i) - y_i) \phi(x_i)_j$$

Gradient descent

- Repeat until "convergence":

$$\theta_j = \theta_j - \alpha \sum_{i=1}^{m} 2(\theta^T \phi(x_i) - y_i)\phi(x_i)_j, \text{ for all } j$$

- Demo:
`https://lukaszkujawa.github.io/gradient-descent.html`
- Stochastic gradient descent

- Let's write $J(\theta)$ a little more compactly using matrix notation; define

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

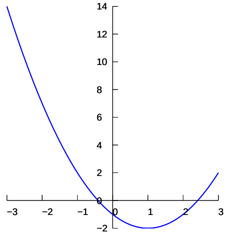
then

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

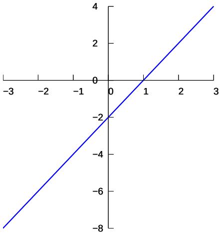
($\|z\|_2$ is ℓ_2 norm of a vector: $\|z\|_2 \equiv \sqrt{\sum_{i=1}^m z_i^2} = \sqrt{z^T z}$)

- Called *least-squares* objective function

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



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



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

$$\theta^\star \text{ minimum} \implies \left.\frac{dJ}{d\theta}\right|_{\theta^\star} = 0$$

$$\implies 2\theta^\star - 2 = 0$$

$$\implies \theta^\star = 1$$

- Multi-variate case: $\theta \in \mathbb{R}^k$, $J : \mathbb{R}^k \to \mathbb{R}$

$$\text{Generalized condition: } \nabla_\theta J(\theta)\big|_{\theta^\star} = 0$$

- $\nabla_\theta J(\theta)$ denotes *gradient* of J with respect to θ

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

- 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

$$\begin{aligned} J(\theta) &= \|\Phi\theta - y\|_2^2 \\ &= (\Phi\theta - y)^T(\Phi\theta - y) \\ &= \theta^T\Phi^T\Phi\theta - 2y^T\Phi\theta + y^T y \end{aligned}$$

- Using the previous gradient rules

$$\begin{aligned} \nabla_\theta J(\theta) &= \nabla_\theta(\theta^T\Phi^T\Phi\theta - 2y^T\Phi\theta + y^T y) \\ &= \nabla_\theta(\theta^T\Phi^T\Phi\theta) - 2\nabla_\theta(y^T\Phi\theta) + \nabla_\theta(y^T y) \\ &= 2\Phi^T\Phi\theta - 2\Phi^T y \end{aligned}$$

- Setting gradient equal to zero

$$2\Phi^T\Phi\theta^\star - 2\Phi^T y = 0 \iff \theta^\star = (\Phi^T\Phi)^{-1}\Phi^T y$$

known as the *normal equations*

- Let's see how this looks in MATLAB code

```
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

```
% same as inv(Phi' * Phi) * Phi' * y
theta = Phi \ y;
```

Higher-dimensional inputs

- Input: $x \in \mathbb{R}^2 = \left[\begin{array}{c} \text{temperature} \\ \text{hour of day} \end{array} \right]$
- 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

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

- Same solution as before

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