Linear Regression and Artificial Neural Networks

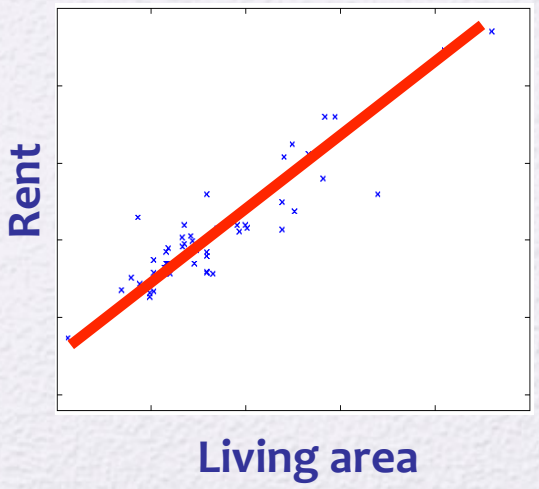
Nan Li

2011.09.27

What is Linear Regression

- Assume that Y (target) is a linear function of X (features):
  - E.g.
    - $\hat{y} = \theta_0 + \theta_1 x^1 + \theta_2 x^2 + \ldots + \theta_k x^k$
  - There are other forms, but let's consider this case for now
- Matrix form:

$$\hat{y} = \mathbf{x}^T \theta$$



What is a good LR model?

- The model that best predicts the target Y given features X
  - i.e. $\hat{y}_i(\mathbf{x}_i) - y_i = \mathbf{x}_i^T\theta - y_i$ to be close to zero
- Our goal is to seek θ that minimize the following **cost function**
  - **Least-Mean-Square (LSM)**

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i^T\theta - y_i)^2$$

How to find the optimal θ?

- Gradient descent

$$\theta_j^{t+1} = \theta_j^t - \alpha \frac{\partial}{\partial \theta_j} J(\theta)\Big|_t$$

$$= \theta_j^t - \alpha \frac{\partial}{\partial \theta_j} \frac{1}{2} \sum_{i=1}^{n} (\mathbf{x}_i^T \theta^t - y_i)^2$$

$$= \theta_j^t + \alpha \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \theta^t) x_i^j$$

  - Converge to global optimum
  - Batch: Guraanteed convergence
  - Online: Fast

How to find the optimal θ?

- Directly minimize $J(\theta)$
  - Take derivative and set to zero

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i^T\theta - y_i)^2$$

$$= \frac{1}{2}(X\theta - \vec{y})^T(X\theta - \vec{y})$$

$$= \frac{1}{2}\left(\theta^T X^T X\theta - \theta^T X^T \vec{y} - \vec{y}^T X\theta + \vec{y}^T\vec{y}\right)$$

Derivation…

- Elements

$$\nabla_\theta \mathrm{tr}\theta^T X^T X\theta = \nabla_\theta \mathrm{tr}\theta\theta^T X^T X$$
$$= X^T X\theta + (X^T X)^T \theta$$
$$= X^T X\theta + X^T X\theta$$

$$\mathrm{tr}ABC = \mathrm{tr}CAB = \mathrm{tr}BCA$$

$$\nabla_A \mathrm{tr}ABA^T C = CAB + C^T AB^T$$

$$\nabla_\theta \mathrm{tr}\vec{y}^T X\theta = \nabla_\theta \mathrm{tr}\theta\vec{y}^T X$$
$$= (\vec{y}^T X)^T$$
$$= X^T \vec{y}$$

$$\mathrm{tr}ABC = \mathrm{tr}CAB = \mathrm{tr}BCA$$

$$\nabla_A \mathrm{tr}AB = B^T$$

More Derivation…

$$\nabla_\theta J = \frac{1}{2}\nabla_\theta \mathrm{tr}\left(\theta^T X^T X\theta - \theta^T X^T \vec{y} - \vec{y}^T X\theta + \vec{y}^T \vec{y}\right)$$

$$= \frac{1}{2}\left(\nabla_\theta \mathrm{tr}\theta^T X^T X\theta - 2\nabla_\theta \mathrm{tr}\vec{y}^T X\theta + \nabla_\theta \mathrm{tr}\vec{y}^T \vec{y}\right)$$

$$= \frac{1}{2}\left(X^T X\theta + X^T X\theta - 2X^T \vec{y}\right)$$

$$= X^T X\theta - X^T \vec{y} = 0$$

$$\nabla_\theta \mathrm{tr}\theta^T X^T X\theta = X^T X\theta + X^T X\theta$$

$$\nabla_\theta \mathrm{tr}\vec{y}^T X\theta = X^T \vec{y}$$

$$\Rightarrow \boxed{X^T X\theta = X^T \vec{y}} \quad \Rightarrow \quad \theta^* = \left(X^T X\right)^{-1} X^T \vec{y}$$

The normal equations

Equilvalence of LMS and MLE

- Assume

$$y_i = \theta^T \mathbf{x}_i + \varepsilon$$

  - where ε follows a Gaussian $N(0,\sigma)$

- Then

$$p(y_i \mid x_i; \theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(- \frac{(y_i - \theta^T \mathbf{x}_i)^2}{2\sigma^2} \right)$$

Equivalence of LMS and MLE, cont.

- By independence assumption:

$$L(\theta) = \prod_{i=1}^{n} p(y_i \mid x_i;\theta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \theta^T \mathbf{x}_i)^2}{2\sigma^2}\right)$$

$$= \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left(-\frac{\sum_{i=1}^{n}(y_i - \theta^T \mathbf{x}_i)^2}{2\sigma^2}\right)$$

- The log-likelihood is:

$$l(\theta) = \log L(\theta) = n \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2}\frac{1}{2}\sum_{i=1}^{n}(y_i - \theta^T \mathbf{x}_i)^2$$

- Recall that:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i^T \theta - y_i)^2$$

- Maximizing $l(\theta)$ is equivalent to minimizing $J(\theta)$

Ridge Regression vs Lasso

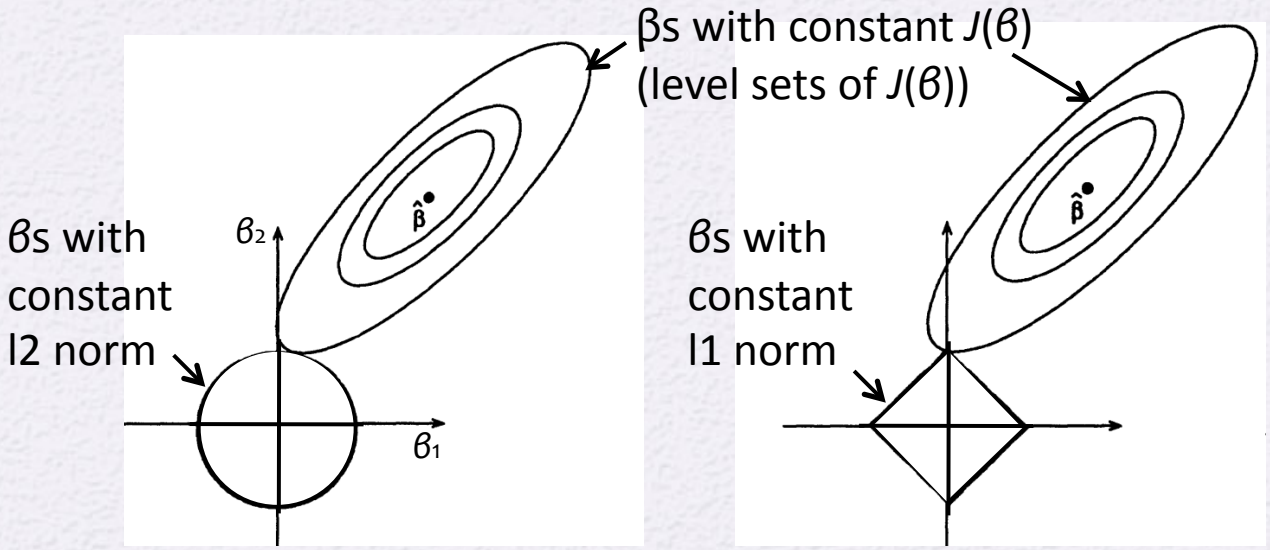
$$\min_{\beta}(\mathbf{X}\beta - \mathbf{Y})^T(\mathbf{X}\beta - \mathbf{Y}) + \lambda \text{pen}(\beta) = \min_{\beta} J(\beta) + \lambda \text{pen}(\beta)$$

Ridge Regression:

$$\text{pen}(\beta) = \|\beta\|_2^2$$

Lasso:

$$\text{pen}(\beta) = \|\beta\|_1$$



Lasso (l1 penalty) results in sparse solutions – vector with more zero coordinates
Good for high-dimensional problems – don't have to store all coordinates!

Bayesian Interpretation

- Ridge regression
  - Gaussian Prior: $p(\beta) \propto e^{-\beta^T \beta / 2\tau^2}$
  - Prior belief that β is Gaussian with zero-mean biases solution to "small" β
- Lasso regression
  - Laplace Prior: $p(\beta_i) \propto e^{-|\beta_i|/t}$
  - Prior belief that β is Laplace with zero-mean biases solution to "small" β

Something More…

- LR with non-linear basis functions
  - LR does not mean we can only deal with linear relationships
  - Linear means linear to θ
  - Features can be non-linear

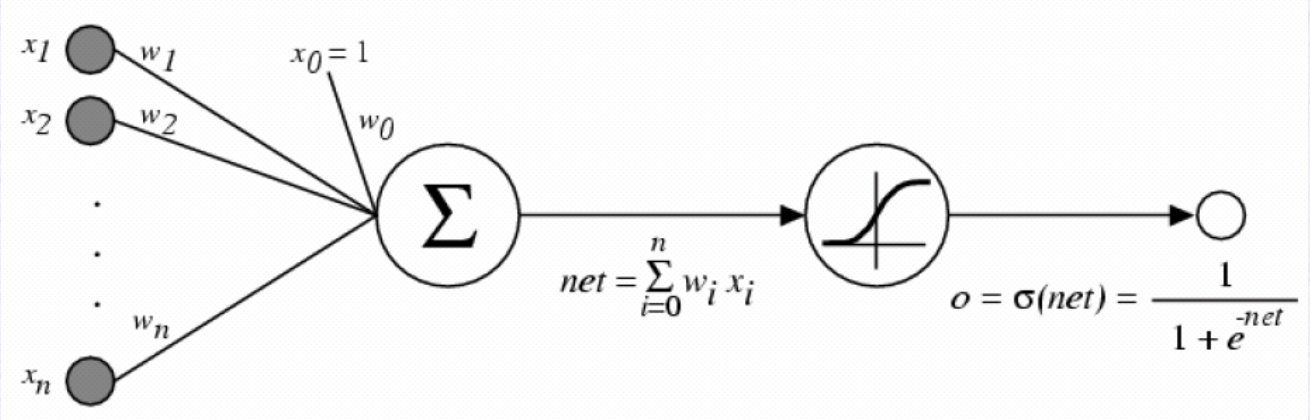
$$y = \theta_0 + \sum_{j=1}^{m} \theta_j \phi(x) = \theta^T \phi(x)$$

    - where the $\phi_j(x)$ are fixed basis functions (and we define $\phi_0(x) = 1$).

- Weighting points
  - Locally weighted LR: Higher weights for training examples closer to the query point
  - Robust regression: Higher weights for the training examples that fit well

Artificial Neural Networks

- A set of connected perceptrons that make prediction based on unit inputs
- Let's start with a single perceptron



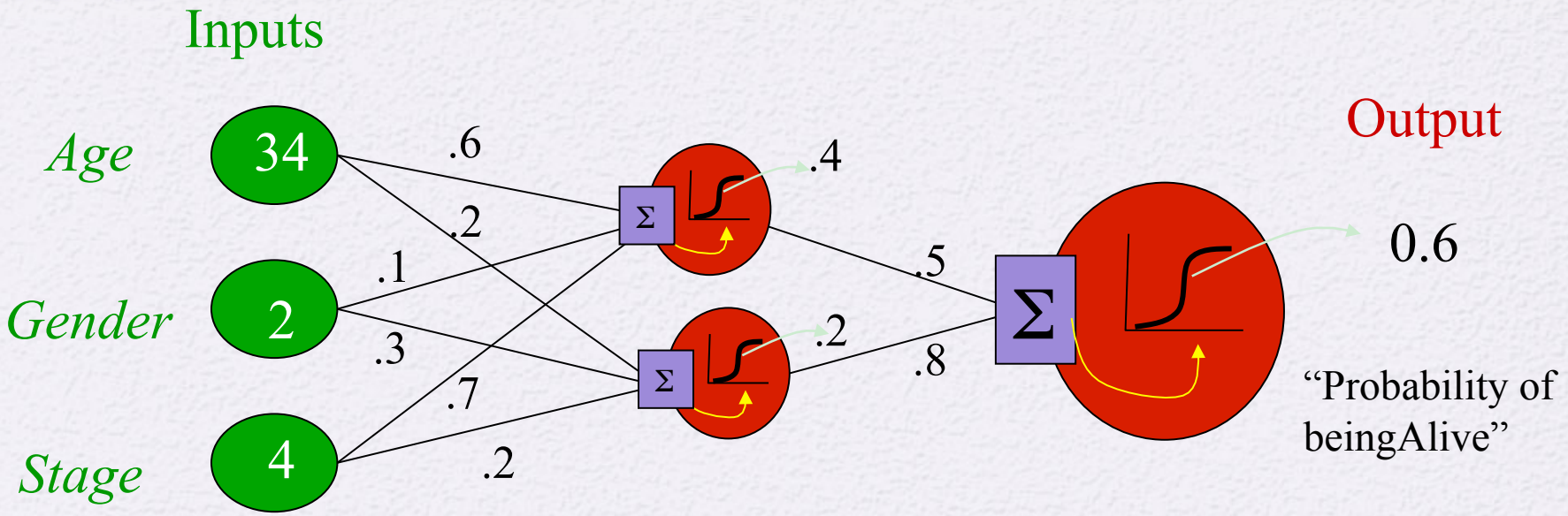
Perceptron Learning

- Find $\vec{w}$ such that it minimizes sum of squared training errors

$$\vec{w} = \arg\min_{\vec{w}} \sum_i \frac{1}{2}(y_i - \hat{f}(x_i; \vec{w}))^2$$

- How?
  - Gradient descent!
- What's the decision boundary?
  - Non-linear?
  - Why?

Multi-layer Neural Networks



- Highly flexible: Non-linear decision boundary

How to learn?

- Backpropagation
  - An iterative method
- For each example
  - Perform forward propagation
  - Start from output layer
  - Compute gradient of node with parents
  - Update weight based on the gradient
- Convergence
  - May stuck in local optimum