Machine Learning

Regression

Regression

- Regression refers to supervised learning problems where the target output is one or more continuous values
  - Continuous output values imply that outputs can be interpolated and that they can be (at least partially) ordered
  - Training data: $D = \left\{ \left(x^{(i)}, y^{(i)} \right) : i \in \{1..n\} \right\}$
  - The task is to learn a function (hypothesis), *h*, such that that $h(x^{(i)})$ is a "good" predictor for $y^{(i)}$

Regression

- Regression is often better formulated as a discriminative learning task
  - Continuous output would require learning a conditional probability density function with continuous conditions for the generative scenario
  - For discriminative learning, we explicitly form a representation for the the function h estimating the MLE/MAP value for the input x
    - Lower dimensional function
    - Does not need to represent the full distribution, only the estimate

Linear Regression

- The simplest representation for regression is a linear function with a set of parameters
  - In a linear representation, the function is linear in terms of the function parameters θ but not necessarily in the features of the input *x*

$$h_\theta(x) = \sum_j \theta_j \phi_j(x)$$

    - A special case of this is the traditional linear representation in the original data features

$$h_\theta(x) = \theta_0 + \sum_{j=1}^{m} \theta_j x_j$$

      - This represents a line in the original feature space

Linear Regression

- Often we will write this in vector/matrix notation

$$h_\theta(x) = \sum_j \theta_j \phi_j(x) = \theta^T \Phi(x)$$

- Since we do not have a probabilistic representation we do need a different performance function
  - Squared error

$$E(\theta) = \sum_{i=1}^{n} E\left(\theta, \left(x^{(i)}, y^{(i)}\right)\right) = \sum_{i=1}^{n} \frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$= \frac{1}{2}\sum_{i=1}^{n}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

Error Minimization

- There are different ways we can minimize an error (or other performance) function
  - Analytic optimization
    $$\partial E(\theta)/\partial\theta = 0$$
    - Find parameters where the derivative is 0
    - Ensure it is a minimum and not a maximum
  - Batch gradient descent
    $$\theta := \theta - \alpha\,\partial E(\theta)/\partial\theta$$
  - Stochastic incremental gradient descent
    $$\theta := \theta - \alpha\,\partial E\left(\theta, \left(x^{(i)}, y^{(i)}\right)\right) \Big/ \partial\theta$$

Linear Regression

- Using the Least Square error function we can derive a stochastic gradient descent method

$$\theta := \theta - \alpha \frac{\partial}{\partial \theta} E\left(\theta, \left(x^{(i)}, y^{(i)}\right)\right) = \theta - \alpha \frac{\partial}{\partial \theta} \frac{1}{2} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$= \theta - \alpha \frac{1}{2} 2 \left(h_\theta(x^{(i)}) - y^{(i)}\right) \frac{\partial}{\partial \theta} h_\theta(x^{(i)}) = \theta - \alpha \left(h_\theta(x^{(i)}) - y^{(i)}\right) \frac{\partial}{\partial \theta} \theta^T \phi\left(x^{(i)}\right)$$

$$= \theta - \alpha \left(h_\theta(x^{(i)}) - y^{(i)}\right) \phi\left(x^{(i)}\right)$$

  - This is also called the Widrow-Hoff learning rule

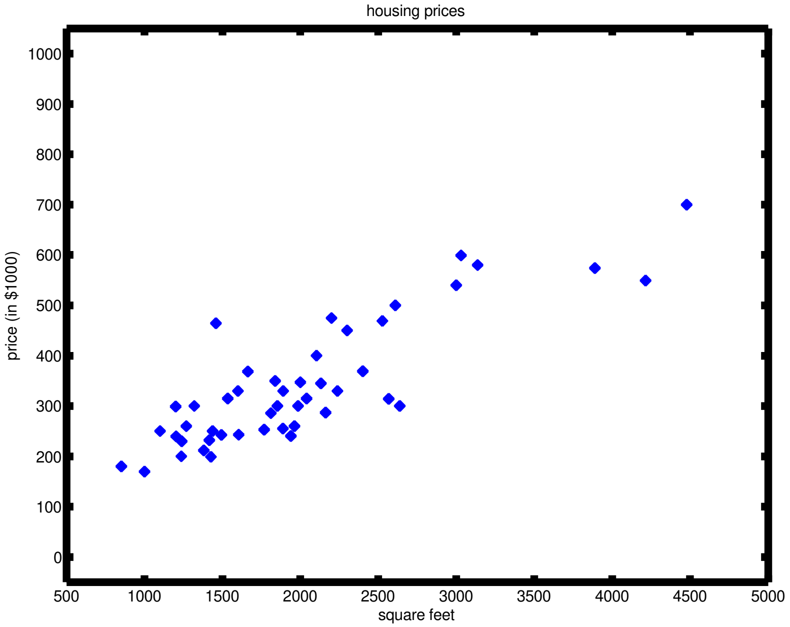
- Converting this to batch gradient descent

$$\theta := \theta - \alpha \frac{\partial}{\partial \theta} E(\theta) = \theta - \alpha \sum_{i=1}^{n} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \phi\left(x^{(i)}\right)$$

Linear Regression Example

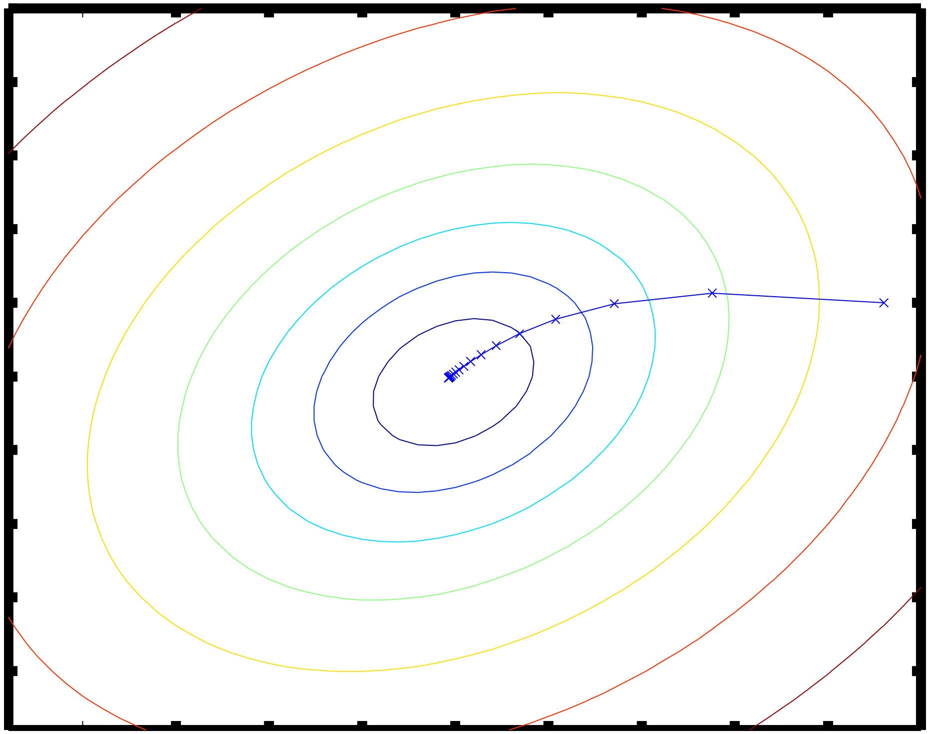
- Example from Andrew Ng.
  - Using simple representation linear in the features

Living area (feet^2)	Price (1000$s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮



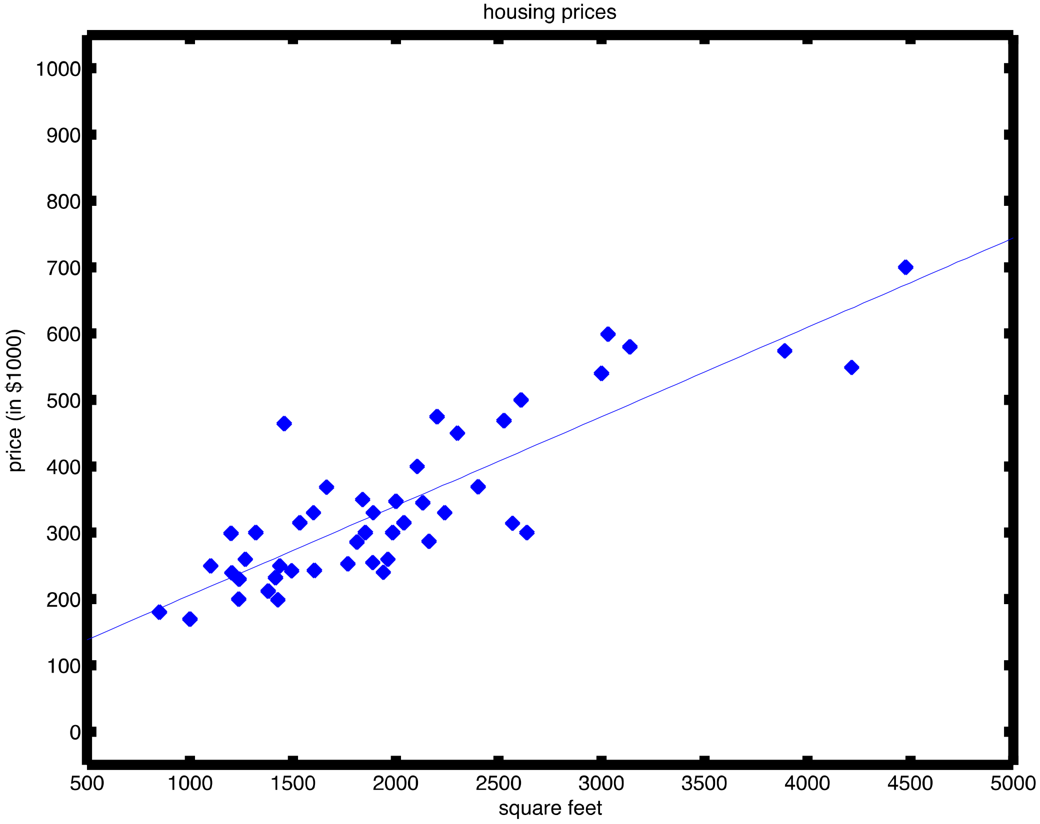
Linear Regression Example

- Using batch gradient descent



Linear Regression Example

- Resulting in a linear predictor



Least Square Error and Maximum Likelihood

- Why is least square a good performance function for Linear regression ?
  - Under certain assumption least square for linear regression leads to the maximum likelihood parameters for a linear function
    - Assume that the real function is linear with noise

$$y^{(i)} = \theta^T \phi(x^{(i)}) + \varepsilon^{(i)}$$

    - Assume that the noise is normally distributed

$$\varepsilon^{(i)} \sim N(0, \sigma^2)$$

Least Square Error and Maximum Likelihood

- Under these assumptions we can express the likelihood of the parameters

$$L(\theta) = p_\theta(\vec{y} \mid X) = \prod_{i=1}^{n} p_\theta(y^{(i)} \mid x^{(i)})$$

$$= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\left(y^{(i)} - \theta^T \phi(x^{(i)})\right)^2}{2\sigma^2}}$$

  - Converting this to log likelihood

$$\log L(\theta) = \log\left(\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\left(y^{(i)} - \theta^T \phi(x^{(i)})\right)^2}{2\sigma^2}} \right)$$

$$= n \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \frac{1}{2} \sum_{i=1}^{n} \left(y^{(i)} - \theta^T \phi(x^{(i)})\right)^2$$

    - Maximum likelihood is the same as least squares

Analytic Optimization

- To simplify notation we convert data and target values into matrices

$$\Phi(X) = \begin{pmatrix} \phi_1(x^{(1)}) & \phi_2(x^{(1)}) & \phi_3(x^{(1)}) & \cdots & \phi_m(x^{(1)}) \\ \phi_1(x^{(2)}) & \phi_2(x^{(2)}) & \phi_3(x^{(2)}) & \cdots & \phi_m(x^{(2)}) \\ \phi_1(x^{(3)}) & \phi_2(x^{(3)}) & \phi_3(x^{(3)}) & \cdots & \phi_m(x^{(3)}) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_1(x^{(n)}) & \phi_2(x^{(n)}) & \phi_3(x^{(n)}) & \cdots & \phi_m(x^{(n)}) \end{pmatrix} \; ; \; Y = \begin{pmatrix} y^{(1)^T} \\ y^{(2)^T} \\ y^{(3)^T} \\ \vdots \\ y^{(n)^T} \end{pmatrix}$$

  - Using this the vector of predicted values *h(X)* can be computed as

$$h(X) = \Phi(X)\theta$$

Analytic Optimization

- The squared error function can then be rewritten as

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

  - To optimize we have to calculate the derivative
    - Derivative of a matrix is the matrix of its partial derivatives with respect to all the parameters
      - This is also called its Jacobian

$$\nabla_\theta E(\theta) = \begin{pmatrix} \frac{\partial E}{\partial \theta_{1,1}} & \cdots & \frac{\partial E}{\partial \theta_{1,k}} \\ \vdots & \ddots & \vdots \\ \frac{\partial E}{\partial \theta_{m,1}} & \cdots & \frac{\partial E}{\partial \theta_{m,k}} \end{pmatrix}$$

Analytic Optimization

- For linear regression the derivative can be derived

$$\nabla_\theta E(\theta) = \nabla_\theta \frac{1}{2}(h(X) - Y)^T (h(X) - Y) = \nabla_\theta \frac{1}{2}(\Phi(X)\theta - Y)^T (\Phi(X)\theta - Y)$$

$$= \Phi(X)^T \Phi(X)\theta - \Phi(X)^T Y$$

  - This leads to an analytic solution

$$\Phi(X)^T \Phi(X)\theta - \Phi(X)^T Y = 0$$

$$\rightarrow \Phi(X)^T \Phi(X)\theta = \Phi(X)^T Y \longleftarrow$$ These are the Normal Equations of the Least Squares problem

$$\rightarrow \theta = \left(\Phi(X)^T \Phi(X)\right)^{-1} \Phi(X)^T Y = \Phi(X)^+ Y$$

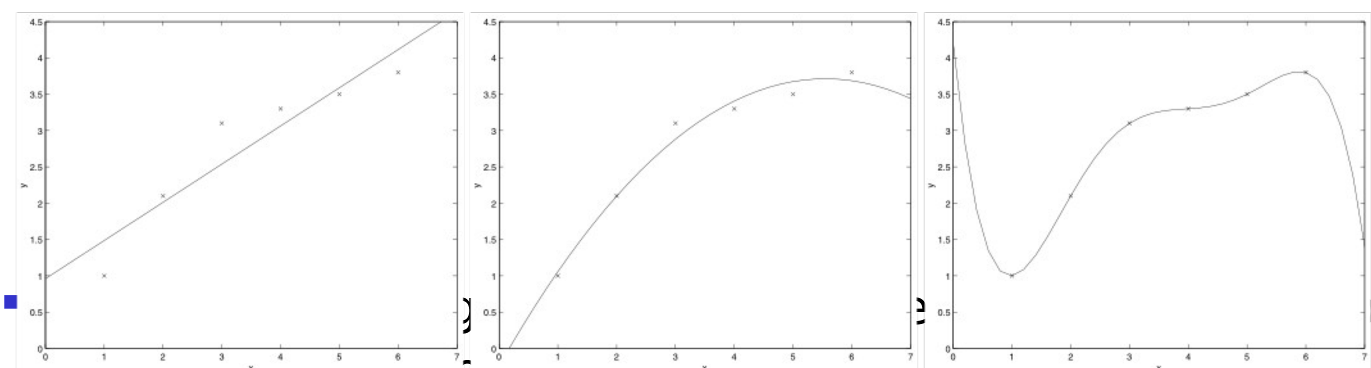
    - A^+ is called the Moore-Penrose Pseudoinverse

Analytic Optimization

- Using Pseudoinverse poses challenges
  - Computation the Pseudoinverse for matrices with a large number of rows can be numerically unstable
  - Pseudoinverses can be ill-conditioned, leading to numerically unstable/incorrect results
  - Results should always be verified
- There are more stable ways to solve this optimization problem
  - E.g. use of extended system and QR factorization
  - Incremental optimization

Linear Regression

- Choosing feature functions *$\phi(x)$* allows linear regression to represent non-linear functions
  - Still linear in the learned parameters
    - Can not learn the type of non-linearity (only choose it)
  - Adding many feature functions leads to overfitting



    - different regions of $\phi(x)$

Locally Weighted Linear Regression

- Different ways to increase the types of relations that linear regression can form exist
  - Locally weighted linear regression combines ideas of linear regression with ideas from non-parametric representations
    - Error is weighted squared error where the weight determines influence of a point and is based on similarity

$$E(\theta, x) = \sum_{i=1}^{n} w^{(i)}(x) E\left(\theta, \left(x^{(i)}, y^{(i)}\right)\right) = \frac{1}{2} \sum_{i=1}^{n} w^{(i)}(x) \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

      - Weights are not parameters to be learned but fixed functions of the point for which the error is calculated

Locally Weighted Linear Regression

- Common weights are local to the point, e.g.:

$$w^{(i)}(x) = e^{-\frac{\left(x^{(i)} - x\right)^2}{2\gamma^2}}$$

  - Since weights are positive they can be pulled into the design matrix $\Phi(X)$ and the target output Y
  - Can be solved the same way as linear regression once the data point x is known

Locally Weighted Linear Regression

- For each point weights can be computed, reducing it to a weighted least squares problem with fixed weights
  - Corresponds to optimizing a different Error function for each query
- Locally linear regression forms one regression line for each data point, leading to a smooth regression curve using linear regression
  - Can have problems when there is limited data for which the weight function is sufficiently large

Overfitting

- Locally linear regression addresses overfitting by allowing a function to be approximated with fewer features
- Another method to address overfitting is regularization
  - Introduce penalty term for the factor introducing overfitting: complexity of the representation
    - Penalize large coefficients as a measure of complexity
      - L1 regularization: Add penalty of $\sum_i \|\theta\|_1$
      - L2 regularization: Add penalty of $\sum_i \|\theta\|_2$

Linear Regression

- Linear regression is a powerful technique to build Maximum Likelihood predictors for regression problems
  - Has only to be linear in the learning parameters and can thus represent limited non-linear function in terms of the input
    - Choice of feature functions is important for its expressiveness
  - Analytic solution can be computed but is sometimes numerically unstable
  - Incremental techniques can be used for on-line learning or if analytic solution is not stable