Regression

CPS 271

Ron Parr

Supervised Learning

- Given: Training Set
- Goal: Good performance on test set
- Assumptions:
  - Training samples are independently drawn, and identically distributed (IID)
  - Test set is from same distribution as training set

Regression Specifics

- Datum i has feature vector: $x^{(i)}$
- Has real valued target: $y^{(i)}$
- Space of concepts H = linear combinations of feature vectors: $h(x)=\theta^T x$
- Learning objective: Search to find "best" θ
- (This is standard "data fitting" that most people learn in some form or another.)

Linearity of Regression

- Regression typically considered a *linear* method, but…
- Features not necessarily linear
- Features not necessarily linear
- Features not necessarily linear
- Features not necessarily linear
- and, BTW, features not necessarily linear

Regression Examples

- Predicting housing price from:
  - House size, lot size, rooms, neighborhood*, etc.
- Predicting weight from:
  - Sex, height, ethnicity, etc.
- Predicting life expectancy increase from:
  - Medication, disease state, etc.
- Predicting crop yield from:
  - Precipitation, fertilizer, temperature, etc.

What is "best"?

- No obvious answer to this question
- Three compatible answers:
  - Minimize squared error on training set
  - Maximize likelihood of the data (under certain assumptions)
  - Project data into "closest" approximation
- Other answers possible

Minimizing Squared Training Set Error

- Why is this good?
- How could this be bad?
- Minimize:

$$J(\theta)=\sum_{i=1}^{N}\left(\theta\cdot\mathbf{x}^{(i)}-y^{(i)}\right)^2$$

Maximizing Likelihood of Data

- Assume:
  - True model is in H
  - Data have Gaussian noise
- Actually might want:

$$\arg\max_H P(H\mid X)=\frac{P(X\mid H)P(H)}{P(X)}$$

- Is maximizing P(X|H) a good surrogate? (maximizing over θ)

Maximizing P(X|H)

- Assume: $y^{(i)}=\theta^T x^{(i)}+\varepsilon^{(i)}$
- Where: $P(\varepsilon^{(i)})=\frac{1}{\sigma\sqrt{2\pi}}\exp(-\frac{(\varepsilon^{(i)})^2}{2\sigma^2})$

(Gaussian distribution w/mean 0, standard deviation σ)

- Therefore:

$$P(y^{(i)}\mid x^{(i)},\theta)=\frac{1}{\sigma\sqrt{2\pi}}\exp(-\frac{(y^{(i)}-\theta^T x^{(i)})^2}{2\sigma^2})$$

Maximization Continued

- Maximizing over entire data set:

$$\prod_{i=1}^{n}P(y^{(i)}\mid x^{(i)},\theta)=\prod_{i=1}^{n}\frac{1}{\sigma\sqrt{2\pi}}\exp(-\frac{(y^{(i)}-\theta^T x^{(i)})^2}{2\sigma^2})$$

- Maximizing equivalent log formulation: (ignoring constants)

$$\sum_{i=1}^{n}-(y^{(i)}-\theta^T x^{(i)})^2$$

- Or minimizing:

$$\sum_{i=1}^{n}(y^{(i)}-\theta^T x^{(i)})^2$$

Look familiar?

Checkpoint

- So far we have considered:
  - Minimizing squared error on training set
  - Maximizing Likelihood of training set (given model, and some assumptions)
- Different approaches w/same objective!

Design Matrix

- We call A the design matrix
- Columns of A are features
- Rows of A are data
- A_{ij}=Feature j of training instance i

Geometric Interpretation

- $Y=(Y^{(1)}\ldots Y^{(n)})$ = point in n-space
- $A\theta$ = H = column space of features
- $A\theta$ = subspace of R^n occupied by H
- Goal: Find "closest" point in H to Y

- Suppose closeness = Euclidean distance

Minimizing Euclidean Distance

- Minimize: $|Y - A\theta|_2$
- For n data points:

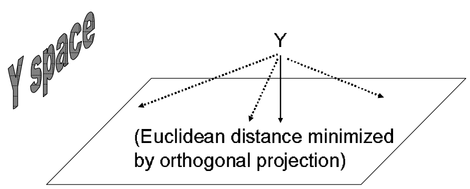
$$\sqrt{\sum_{i=1}^{n}(y^{(i)} - x^{(i)^T}\theta)^2}$$

- Equivalent to minimizing:

$$\sum_{i=1}^{n}(y^{(i)} - x^{(i)^T}\theta)^2$$

Look familiar?

Another Geometric Interpretation



H space (linear combinations of X)

Checkpoint

- Three different ways to pick H
  - Minimize squared error on training set
  - Maximize likelihood of training set
  - Distance minimizing projection into H

- All lead to same optimization problem!

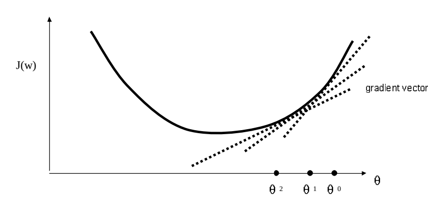
$$\arg\min_{\theta} J(\theta) = \sum_{i=1}^{N}\left(\theta \cdot \mathbf{x}^{(i)} - y^{(i)}\right)^2$$

Solving the Optimization Problem

- Nota bene: Good to keep optimization problem and optimization technique separate in your mind

- Some optimization approaches:
  - Gradient descent
  - Direct Minimization derived from
    - Calculus
    - Geometric constraints

Minimizing J by Gradient Descent



Start with initial weight vector θ_0

Compute the gradient $\nabla J(\theta) = \left(\frac{\partial J(\theta)}{\partial \theta_0}, \frac{\partial J(\theta)}{\partial \theta_1}, \ldots, \frac{\partial J(\theta)}{\partial \theta_n}\right)$

Compute $\theta = \theta - \alpha \nabla J(\theta)$ where α is the step size

Repeat until convergence

(Adapted from Lise Getoor's Slides)

Gradient Descent Issues

- For this particular problem:
  - Global minimum exists
  - Convergence "guaranteed" if done in "batch"
- In general
  - Local optimum only
  - Batch mode more stable
  - Incremental possible
    - Can oscillate
    - Use decreasing step size (Robbins-Monro) to stabilize

Direct Solution

- Geometric Approach (Strang)
- Let A be the design matrix
- Require orthogonality:

$$\forall z : (Az)^T (A\theta - Y) = 0$$

Any vector in H

Line from Y to solution

$$\forall z : z^T [A^T A\theta - A^T Y] = 0$$

Direct Solution Continued

- When is this true: $\forall z : z^T [A^T A\theta - A^T Y] = 0$
- When:

$$A^T A\theta - A^T Y = 0$$
$$\theta = (A^T A)^{-1} A^T Y$$

When does the inverse exist?

Columns of A must be independent.

What about other criteria?

- How about minimizing worse case loss?

$$\min_\theta \max_i \left(\theta \cdot \mathbf{x}^{(i)} - y^{(i)}\right)$$

- Solve by linear program…