Lecture 3: Linear Regression & Optimisation

Linear Regression

- Assume a probabilistic model
  - $y = X\beta + \epsilon$
- Assume Gaussian noise (independent of X):
  - $\epsilon \sim N(0, \sigma^2)$
- Discriminative model
  - $p(y|\mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp(-\frac{(y-\mathbf{x}\beta)^2}{2\sigma^2})$
- Unknown param: β (and σ^2)
- MLE: choose param values that maximise the probability of observed data (likelihood)
  - "Log trick": instead of maximising likelihood, we can maximise log-likelihood (since log is strictly monotonic increasing)
- Under this model ("normal" linear regression):
  - MLE is equivalent to minimising SSE (or RSS)

Optimization

- Training = Fitting = Parameter estimation
- Typical formulation (minimise loss = objective)
  - $\hat{\theta} \in \arg\min_{\theta \in \Theta} L(data, \theta)$
- Analytic (aka. closed form) solution
  - 1st order derivatives for optimality:
    - $\frac{\partial L}{\partial \theta_1} = ... = \frac{\partial L}{\partial \theta_p} = 0$
  - (Need to check 2nd derivative)
- Approximate iterative solution (e.g. IRWLS)
  - Initialisation: choose starting guess $\theta^{(1)}$, set $i = 1$
  - Update: $\theta^{(i+1)} \leftarrow SomeRule[\theta^{(i)}]$, set $i \leftarrow i + 1$
  - Termination: decide whether to stop
  - Go to step 2
  - Stop: return $\hat{\theta} \approx \theta^{(i)}$

Coordinate descent

- 一次 update 一个 θ_i
- Suppose $\theta = [\theta_1, ..., \theta_K]^T$
  1. Choose $\theta^{(1)}$ and some T
  2. For i from 1 to T (update all params T times)
     - $\theta^{(i+1)} \leftarrow \theta^{(i)}$ (copy param values)
     - For j from 1 to K: (update one param each time)
       - Fix components of $\theta^{(i+1)}$, except j-th component
       - Find $\hat{\theta}_j^{(i+1)}$ that minimises $L(\theta_j^{(i+1)})$
       - Update j-th component of $\theta^{(i+1)}$
  3. Return $\hat{\theta} \approx \theta^{(i)}$
- (Other stopping criteria can be used)

Gradient descent

- Gradient denoted as $\nabla L = [\frac{\partial L}{\partial \theta_1}, ..., \frac{\partial L}{\partial \theta_p}]^T$
- Algorithm:
  1. Choose $\theta^{(1)}$ and some T
  2. For i from 1 to T^*
     - update: $\theta^{(i+1)} = \theta^{(i)} - \eta \nabla L(\theta^{(i)})$
  3. Return $\hat{\theta} \approx \theta^{(i)}$
- Note: η (learning rate) is dynamically updated in each step
- Variants: SGD, mini batches, momentum, AdaGrad

Convex objective functions

- "Bowl" shaped functions
- Every local min is global min
- Informal definition: line segment between any two points on graph of function lies above or on the graph
- Gradient descent on (strictly) convex function guaranteed to find a (unique) global minimum

L_1 and L_2 norms

- Norm: length of vectors
- L_2 norm (aka. Euclidean distance)
  - $||a|| = ||a||_2 \equiv \sqrt{a_1^2 + ... + a_n^2}$
- L_1 norm (aka. Manhattan distance)
  - $||a||_1 \equiv |a_1| + ... + |a_n|$
- E.g. Sum of squared errors:
  - $L = \sum_{i=1}^{n}(y_i - \sum_{j=0}^{m} X_{ij}\beta_j)^2 = ||y - X\beta||^2$

Linear Regression optimization: Least Square Method

- To find β, minimize the **sum of squared errors**:
  - $SSE/RSS = \sum_{i=1}^{n}(y_i - \sum_{j=0}^{m} X_{ij}\beta_j)^2$
- Setting derivative to zero and solve for β: (normal equation)
  - $b = (X^T X)^{-1} X^T y$
  - Well defined only if the inverse exists

Lecture 4: Logistic Regression & Basis Expansion

Logistic Regression

- Why not linear regression for classification?
  - Predict "Yes" if $s \geq 0.5$
  - Predict "No" if $s < 0.5$
  - ($s = x\hat{\beta}$, estimated probability for class 1 given a data point)
  - Reason:
    - Can be susceptible (易受影响) to outliers

        - least-squares criterion looks **unnatural** in this setting
- Problem: the probability needs to be between 0 and 1
- Logistic function
    - $f(s) = \frac{1}{1+\exp(-s)}$
- Logistic regression model:
    - $P(Y=1|x) = \frac{1}{1+\exp(-x^T\beta)}$
    - In GLM:
        - Mean: $\mu = P(Y=1|x)$
        - Link function: $g(\mu) = log\frac{P(Y=1|x)}{P(Y=0|x)} = \eta = x^T\beta$ (log odds)
    - (Can use Cross-Validation to choose the threshold, usually just use 0.5)
- Logistic regression is a linear classifier
    - Logistic regression model:
        - $P(Y=1|x) = \frac{1}{1+\exp(-x^T\beta)}$
    - Classification rule:
        - Class "1"
            - If $P(Y=1|x) = \frac{1}{\exp(-x^T\beta)} > \frac{1}{2}$
            - Else class "0"
        - Decision boundary (line):
            - $P(Y=1|x) = \frac{1}{1+\exp(-x^T\beta)} = \frac{1}{2}$
            - Equivalently, $P(Y=0|x) = P(Y=1|x)$
        - (In higher dimensional problems, the decision boundary is a plane or hyperplane, vector β is perpendicular to the decision boundary)

Linear v.s. logistic probabilistic models

- Linear regression
    - Assume $\epsilon \sim N(0, \sigma^2)$
    - Therefore assume $y \sim N(X\beta, \sigma^2)$
- Logistic regression
    - Assume $y \sim Bernoulli(p = logistic(x^T\beta))$

Logistic MLE

- Doesn't have closed form solution (cannot solve $\frac{\partial L}{\partial \beta} = 0$ directly)
- Therefore use iterative optimisation
    - E.g. Newton-Raphson, IRWLS, or gradient descent
- Good news: it's a convex problem (if no irrelevant features) $\Rightarrow$ guaranteed to get global minimum

Information divergence (extra)

- To **compare models** with different num of params in an all-subsets search
- May use information theoretic criterion which estimates information divergence between true model and a given candidate model (working model)
- Best model: **smallest** criterion value
- E.g. Kullback-Leibler Information
    - $KL(f_1, f_2) = E_{f_1}[log\frac{f_2}{f_1}] = \int_x log\frac{f_2}{f_1}f(x)dx$

    - Problem: don't know the true model $\Rightarrow$ cannot compute KL
    - E.g. two binomial distribution: one for working model, one for true model
        - Choose the model minimise KL

Cross entropy

- A method for comparing two distiributions
- A measure of divergence between reference distribution $g_{ref}(a)$ and estiamted distribution $g_{est}(a)$
    - For discrete distribution:
        - $H(g_{ref}, g_{est}) = -\sum_{a \in A} g_{ref}(a) log\, g_{est}(a)$

Training as cross-entropy minimisation

- Consider log-likelihood for a single data point
    - $log\, p(y_i|x_i) = y_i log(\theta(x_i)) + (1 - y_i) log(1 - \theta(x_i))$
- Negative cross entropy
    - $H(g_{ref}, g_{est}) = -\sum_{a} g_{ref}(a) log\, g_{est}(a)$
- Reference (true) distribution:
    - $g_{ref}(1) = y_i$ and $g_{ref}(0) = 1 - y_i$
- Estimate true distribution as:
    - $g_{est}(1) = \theta(x_i)$ and $g_{est}(0) = 1 - \theta(x_i)$
- Find β that minimise sum of cross entropies per training point

Basis Expansion (Data transformation)

- Extend the utility of models via **data transformation**
- For linear regression:
    - Transformation data $\Rightarrow$ make data more linear!
    - Map data onto another feature space s.t. data is linear in that space
        - $\mathbf{x}$: the original set of features
        - ϕ: $\mathbb{R}^m \rightarrow \mathbb{R}^k$ denotes transformation
        - $\phi(\mathbf{x})$: new feature set
- Polynomial regression:
    - New features:
        - $\phi_1(x) = x$
        - $\phi_2(x) = x^2$
    - Quadratic regression (linear in new feature set):
        - $y = w_0 + w_1 \phi_1(x) + w_2 \phi_2(x) = w_0 + w_1 x + w_2 x^2$
- Can be applied for both regression and classification
- There are many possible choices of ϕ
- Binary classification:
    - If dataset not linearly separable (non-linear problem)
    - Define transformation as:
        - $\phi_i(x) = ||x - z_i||$ (euclidean distance)
            - where z_i is some pre-defined **constants**
            - Distances to each z_i as new features

Radial basis functions (RBFs)

- Motivated from approximiation theory
- Sums of RBFs are used to **approximate given functions**
- Radial basis functions:
  - $\phi(x) = \psi(||x - z||)$, where z is a constant
- Examples:
  - $\phi(x) = ||x - z||$
  - $\phi(x) = \exp(-\frac{1}{\sigma}||x - z||^2)$

Challenges of basis expansion (data transformation)

- One limitation: the transformation needs to be defined beforehand
  - Need to choose the **size** of new feature set
  - If using RBFs, need to choose z_i
- Choosing z_i:
  1. Uniformly spaced points (grids)
  2. Cluster training data and use cluster centroids
  3. Use training data $z_i \equiv x_i$ (some x_i)
  - E.g. $\phi_i(x) = \psi(||x - x_i||)$
  - For large datasets, this results in a **large number of features**

Taking the idea of basis expansion to the next level

- One idea: to learn the transformation ϕ from data
  - E.g. Artificial Neural Networks
- Another extension: use kernel trick
- In **sparse kernel machines**, training dependes ony on a few data points
  - E.g. SVM