Linear regression

COMS 4771 Fall 2019

Overview

- Statistical model for regression problems
- Linear regression models
- MLE and ERM

Real-valued predictions I

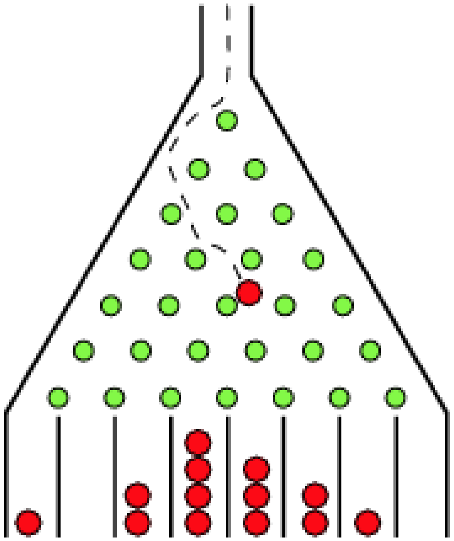


Figure 1: Galton board

Real-valued predictions II

- Physical model: hard
- Statistical model: final position of ball is random
  - *Normal (Gaussian) distribution* with mean μ and variance σ^2
  - Written $\mathrm{N}(\mu, \sigma^2)$
- Goal: predict final position accurately, measure *squared loss* (also called *squared error*)

$$(\text{prediction} - \text{outcome})^2$$

- Note: outcome is random, so look at *expected squared loss* (also called *mean squared error*)

Optimal prediction for mean squared error

- Predict $\hat{y} \in \mathbb{R}$; true final position is Y (random variable) with *mean* $\mathbb{E}(Y) = \mu$ and *variance* $\operatorname{var}(Y) = \mathbb{E}[(Y - \mathbb{E}(Y))^2] = \sigma^2$.
- Squared error is $(\hat{y} - Y)^2$.
- *Bias-variance decomposition*:

- So optimal prediction is $\hat{y} =$

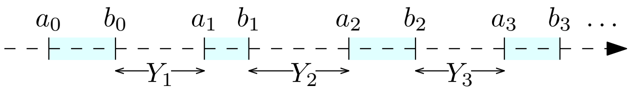
- When parameters are unknown, can estimate from related data, ...

Example: Old Faithful I

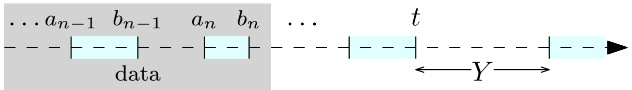
Figure 2: Old Faithful geyser in Yellowstone National Park

Example: Old Faithful II

- Example: When will "Old Faithful" geyser erupt?
- Predict "time between eruptions"
- Old Faithful Geyser Data



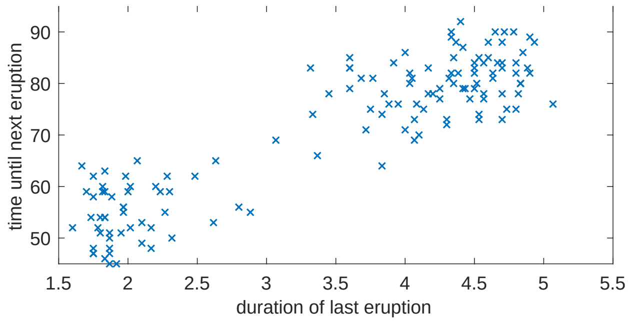
- Mean on past 136 observations: $\hat{\mu} = 70.7941$ minutes
  - So predict $\hat{y} = \hat{\mu} = 70.7941$



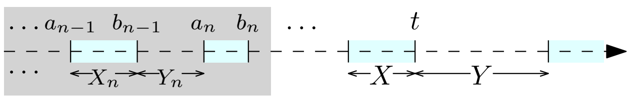
- Mean squared error on next 136 observations: 187.1894
  - Square root: 13.6817 minutes

Looking at the data

- Henry Woodward observed that "time between eruptions" seems related to "duration of latest eruption"



- Use "duration of latest eruption" as feature x
- Can use x to predict time until next eruption, y



Statistical model for regression

- Setting is same as for classification except:
  - Label is real number, rather than $\{0,1\}$ or $\{1,2,\ldots,K\}$
  - Care about squared error, rather than whether prediction is correct
  - *Risk* of f:
    $$\mathcal{R}(f) := \mathbb{E}[(f(X)-Y)^2],$$
    the expected squared loss of f on random example
- Note: "error rate" is also "risk", but with different *loss function*, called *zero-one loss* $\mathbb{1}_{\{f(x)\neq y\}}$

Optimal prediction function for regression

- If (X,Y) is random test example, then *optimal prediction function* is
  $$f^\star(x) = \mathbb{E}[Y \mid X = x]$$
  - Also called the *regression function*
  - Prediction function with smallest risk
  - Depends on conditional distribution of Y given X

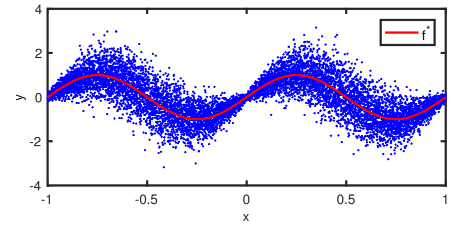


Figure 3: Example of regression function

Linear regression models

- Suppose $\boldsymbol{x}$ is given by d real-valued features, so $\boldsymbol{x} \in \mathbb{R}^d$
- *Linear regression model* for $(\boldsymbol{X}, Y)$:
  - $Y \mid \boldsymbol{X} = \boldsymbol{x} \sim \mathrm{N}(\boldsymbol{x}^\top\boldsymbol{w}, \sigma^2)$ (or really, any distribution with mean $\boldsymbol{x}^\top\boldsymbol{w}$ and variance σ^2)
  - $\boldsymbol{w} \in \mathbb{R}^d$ is parameter vector of interest
  - $\sigma^2 > 0$ is another parameter (not important for prediction)
  - $\boldsymbol{w}$ and σ^2 not involved in marginal distribution of X (which we don't care much about)

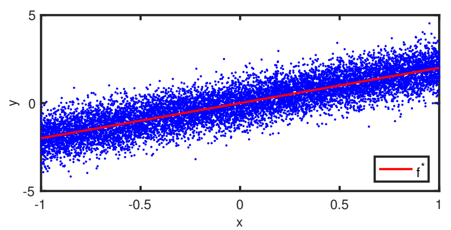


Figure 4: A linear regression function

Upgrading linear regression

- Make linear regression more powerful by being creative about features
- Instead of using $\boldsymbol{x}$ directly, use $\boldsymbol{\varphi}(\boldsymbol{x})$ for some transformation $\boldsymbol{\varphi}$ (possibly vector-valued)
- Examples:
  - *Non-linear scalar transformations*, e.g., $\varphi(x) = \ln(1+x)$
  - *Logical formula*, e.g., $\varphi(\boldsymbol{x}) = (x_1 \wedge x_5 \wedge \neg x_{10}) \vee (\neg x_2 \wedge x_7)$
  - *Trigonometric expansion*, e.g., $\boldsymbol{\varphi}(x) = (1, \sin(x), \cos(x), \sin(2x), \cos(2x), \ldots)$
  - *Polynomial expansion*, e.g., $\boldsymbol{\varphi}(\boldsymbol{x}) = (1, x_1, \ldots, x_d, x_1^2, \ldots, x_d^2, x_1x_2, \ldots, x_{d-1}x_d)$
  - *Headless neural network* $\boldsymbol{\varphi}(\boldsymbol{x}) = \boldsymbol{N}(\boldsymbol{x}) \in \mathbb{R}^k$, where $\boldsymbol{N}\colon \mathbb{R}^d \to \mathbb{R}^k$ is a map computed by a intermediate layer of a neural network

Example: Taking advantage of linearity

- Example: y is health outcome, x is body temperature
  - Physician suggests relevant feature is (square) deviation from normal body temperature $(x - 98.6)^2$
  - What if you didn't know the magic constant 98.6?

Example: Affine expansion

- Another example: Woodward used *affine expansion*
  - $\varphi(x) = (1, x)$
  - Parameter vector $\boldsymbol{w} = (a, b)$
  - $\varphi(x)^{\mathsf{T}}\boldsymbol{w} = a + bx$, so a is intercept term
  - Generalizes to d features: just prepend the constant 1 feature $\varphi(\boldsymbol{x}) = (1, \boldsymbol{x}) \in \mathbb{R}^{d+1}$

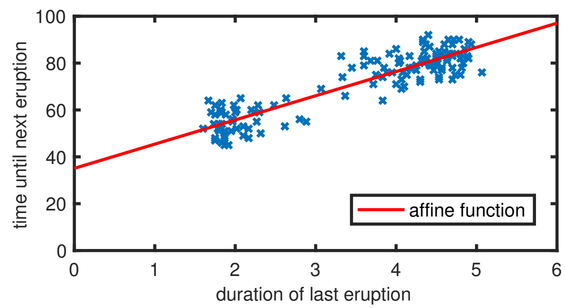


Figure 5: Affine fit to Old Faithful data

Text features

- How to get features for text?

- Suppose input is a word (sequence of characters).
  - $x_{\text{starts\_with\_anti}} = \mathbb{1}_{\{\text{starts with “anti”}\}}$
  - $x_{\text{ends\_with\_ology}} = \mathbb{1}_{\{\text{ends with “ology”}\}}$
  - ... (same for all four- & five-letter prefixes & suffixes)
  - $x_{\text{length}\leq 3} = \mathbb{1}_{\{\text{length } \leq 3\}}$
  - $x_{\text{length}\leq 4} = \mathbb{1}_{\{\text{length } \leq 4\}}$
  - ... (same with all positive integers ≤ 20)

- Suppose input is a document (sequence of words).
  - $x_{\text{contains\_aardvark}} = \mathbb{1}_{\{\text{contains “aardvark”}\}}$
  - ... (same for all words in dictionary)
  - $x_{\text{contains\_each\_day}} = \mathbb{1}_{\{\text{contains “each day”}\}}$
  - ... (same for all “bigrams” of words in dictionary)
  - $x_{\text{count\_aardvark}} = \#$ appearances of “aardvark”
  - ... (same for all words, “bigrams”, ...)

- End up with many features!

Sparse representations

- *Sparse representation* (e.g., via hash table)
  - E.g., “see spot run”
  - `x = { "contains_see":1, "contains_spot":1, "contains_run":1, "contains_see_spot":1, "contains_spot_run":1 }`
- C.f. *dense representation*, which stores a lot of zeros for all of the words / bigrams that don't appear.

- What is computational cost of computing $\boldsymbol{x}^{\mathsf{T}}\boldsymbol{z}$?

Fitting linear regression models to data

- Treat training examples as iid, same distribution as test example
  - $Y \mid \boldsymbol{X} = \boldsymbol{x} \sim \mathrm{N}(\boldsymbol{x}^{\mathsf{T}}\boldsymbol{w}, \sigma^2)$
- Log-likelihood of $(\boldsymbol{w}, \sigma^2)$ given data $(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$:

$$\sum_{i=1}^{n} \left\{ -\frac{1}{2\sigma^2}(\boldsymbol{x}_i^{\mathsf{T}}\boldsymbol{w} - y_i)^2 + \frac{1}{2}\ln\frac{1}{2\pi\sigma^2} \right\} + \left\{ \text{terms not involving } (\boldsymbol{w}, \sigma^2) \right\}$$

- The $\boldsymbol{w}$ that maximizes log-likelihood is same $\boldsymbol{w}$ that minimizes

$$\frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i^{\mathsf{T}}\boldsymbol{w} - y_i)^2.$$

MLE coincides with ERM

- *Empirical distribution* P_n on $(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_n, y_n)$: distribution that puts probability mass $1/n$ on each training example.
- Execute the plug-in principle:
  - We want to find $f \colon \mathbb{R}^n \to \mathbb{R}$ that minimizes risk

$$\mathcal{R}(f) = \mathbb{E}[(f(\boldsymbol{X}) - Y)^2],$$

    but we don't know distribution P of $(\boldsymbol{X}, Y)$ (or even conditional distribution of Y given $\boldsymbol{X}$)
  - Replace P with P_n to get *empirical risk*

$$\widehat{\mathcal{R}}(f) := \frac{1}{n}\sum_{i=1}^{n}(f(\boldsymbol{x}_i) - y_i)^2,$$

    which is the risk of f pretending that the distribution of $(\boldsymbol{X}, Y)$ is P_n.
  - So find f to minimize empirical risk: *Empirical Risk Minimizer (ERM)*
- For linear functions $f(\boldsymbol{x}) = \boldsymbol{x}^{\mathsf{T}}\boldsymbol{w}$, same as MLE for $\boldsymbol{w}$ in linear regression model (!!)

Geometric picture of empirical risk

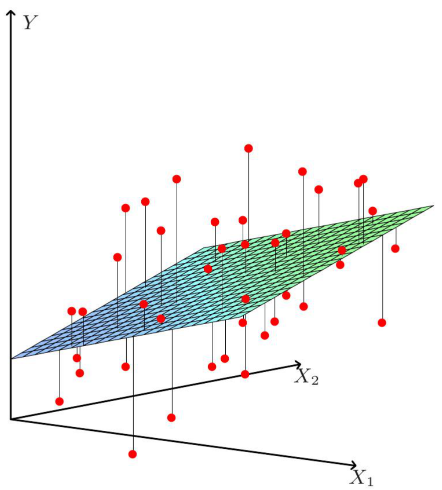


Figure 6: Empirical risk of $\boldsymbol{w}$ is average of vertical squared distances from hyperplane to data points

ERM in matrix notation

- Let $\boldsymbol{A} = \frac{1}{\sqrt{n}}\begin{bmatrix} \leftarrow & \boldsymbol{x}_1^{\mathsf{T}} & \rightarrow \\ & \vdots & \\ \leftarrow & \boldsymbol{x}_n^{\mathsf{T}} & \rightarrow \end{bmatrix} \in \mathbb{R}^{n \times d}$ and $\boldsymbol{b} = \frac{1}{\sqrt{n}}\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^n$
- Empirical risk is

$$\widehat{\mathcal{R}}(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i^{\mathsf{T}}\boldsymbol{w} - y_i)^2 = \|\boldsymbol{A}\boldsymbol{w} - \boldsymbol{b}\|_2^2.$$

Normal equations

- From calculus:
  - Necessary condition for $\boldsymbol{w}$ to be minimizer of $\widehat{\mathcal{R}}$ is that gradient of $\widehat{\mathcal{R}}$ at $\boldsymbol{w}$ should vanish: $\nabla\widehat{\mathcal{R}}(\boldsymbol{w}) = \boldsymbol{0}$
  - Equivalent to $(\boldsymbol{A}^\top\boldsymbol{A})\boldsymbol{w} = \boldsymbol{A}^\top\boldsymbol{b}$
  - System of linear equations in $\boldsymbol{w}$, called the *normal equations*
  - Every solution $\boldsymbol{w}$ to normal equations is a minimizer of $\widehat{\mathcal{R}}$:

Algorithm for ERM

- Algorithm for finding ERM: Gaussian elimination to solve normal equations
  - Running time $O(nd^2)$
  - Can get good approximate solution in linear time $O(nd)$
  - Also called *Ordinary Least Squares (OLS)*

Linear algebraic interpretation of ERM

- Write $\boldsymbol{A} = \begin{bmatrix} \uparrow & & \uparrow \\ \boldsymbol{a}_1 & \cdots & \boldsymbol{a}_d \\ \downarrow & & \downarrow \end{bmatrix}$
  - $\boldsymbol{a}_j \in \mathbb{R}^n$ is j-th column of $\boldsymbol{A}$
  - Span of $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_d$ is $\operatorname{range}(\boldsymbol{A})$, a subspace of $\mathbb{R}^n$
- Minimizing $\|\boldsymbol{A}\boldsymbol{w} - \boldsymbol{b}\|^2$ over $\boldsymbol{w} \in \mathbb{R}^d$ is same as finding vector $\hat{\boldsymbol{b}}$ in $\operatorname{range}(\boldsymbol{A})$ closest to $\boldsymbol{b}$
- Solution $\hat{\boldsymbol{b}}$ is *orthogonal projection* of $\boldsymbol{b}$ onto $\operatorname{range}(\boldsymbol{A})$

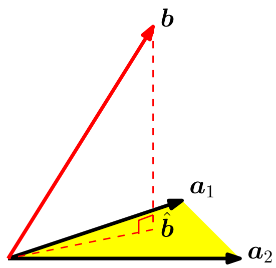


Figure 7: Projection of $\boldsymbol{b}$ onto $\operatorname{range}(\boldsymbol{A})$

Performance of ERM

- How well does ERM solution $\hat{\boldsymbol{w}}$ work?
  - Study in context of IID model
  - Best linear predictor $\boldsymbol{w}^\star$: minimizer of $\mathcal{R}(\boldsymbol{w})$.
  - Hope that $\mathcal{R}(\hat{\boldsymbol{w}}) \approx \mathcal{R}(\boldsymbol{w}^\star)$
- **Theorem**: In IID model, ERM solution $\hat{\boldsymbol{w}}$ satisfies

$$\mathcal{R}(\hat{\boldsymbol{w}}) \to \mathcal{R}(\boldsymbol{w}^\star) + \frac{\operatorname{tr}(\operatorname{cov}(\varepsilon\boldsymbol{W}))}{n}$$

  as $n \to \infty$, where $\boldsymbol{W} = \mathbb{E}[\boldsymbol{X}\boldsymbol{X}^\top]^{-1/2}\boldsymbol{X}$ and $\varepsilon = Y - \boldsymbol{X}^\top\boldsymbol{w}^\star$.

- If $(\boldsymbol{X}, Y)$ follows linear regression model $Y \mid \boldsymbol{X} = \boldsymbol{x} \sim \mathrm{N}(\boldsymbol{x}^\top\boldsymbol{w}^\star, \sigma^2)$, then theorem simplifies to

$$\mathcal{R}(\hat{\boldsymbol{w}}) \to \mathcal{R}(\boldsymbol{w}^\star) + \frac{\sigma^2 d}{n} = \left(1 + \frac{d}{n}\right)\sigma^2.$$

Risk vs empirical risk

- Let $\hat{\boldsymbol{w}}$ be ERM solution.
- How do $\widehat{\mathcal{R}}(\hat{\boldsymbol{w}})$ and $\mathcal{R}(\hat{\boldsymbol{w}})$ compare?
- **Theorem**: In IID model, $\mathbb{E}[\widehat{\mathcal{R}}(\hat{\boldsymbol{w}})] \leq \mathbb{E}[\mathcal{R}(\hat{\boldsymbol{w}})]$

- *Over-fitting*: when true risk is much higher than empirical risk.
- Note: Can estimate risk using test set, just as for classification problems.

Example of over-fitting

- $\varphi(x) = (1, x, x^2, \ldots, x^k)$, degree-$k$ polynomial expansion
- Dimension is $d = k + 1$
- Any function of $\leq k + 1$ points can be interpolated by polynomial of degree $\leq k$
- So if $n \leq k + 1 = d$, ERM solution $\hat{\boldsymbol{w}}$ will have $\widehat{\mathcal{R}}(\hat{\boldsymbol{w}}) = 0$, even if true risk is $\gg 0$.

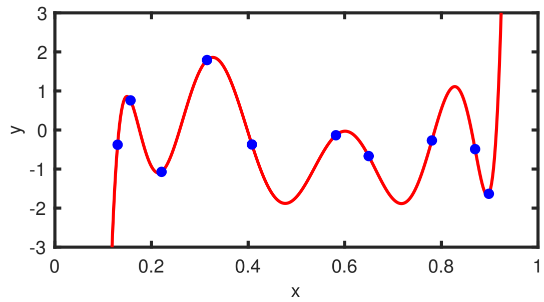


Figure 8: Polynomial interpolation

Outliers

- Common issue with using squared loss: sensitive to *outliers*
  - Roughly: data points that don't fit the same pattern as the rest
  - Does removing the data point drastically change the fit?

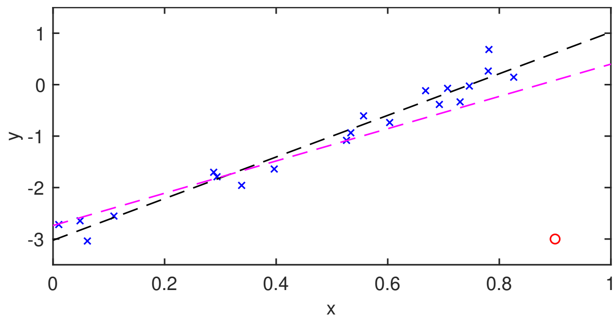


Figure 9: Effect of single outlier

Absolute loss

- One "fix": change loss function
  - Common choice: *absolute loss* $|\hat{y} - y|$

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} |\boldsymbol{x}_i^{\mathsf{T}} \boldsymbol{w} - y_i|$$

  - Instead of solving linear system, now solve a linear program
  - Less sensitive to abnormal y-values than squared loss
  - However: changes what we are estimating ...

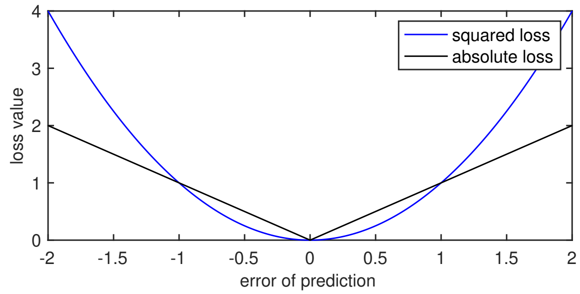


Figure 10: Absolute loss vs squared loss

Heuristics for dealing with outliers

- Heuristic I: random sample consensus (RANSAC)
  - Pick a random subsample of data points — hopefully no outliers are picked! — and fit model to this subsample
  - If most of the remaining data are "well-fit", then halt
  - Else, try again
- Heuristic II: iterative trimming
  - Fit training data as usual
  - Throw out some of the least "well-fit" data points
  - Repeat until fit does not change too much
- Both heuristics are rather drastic!
  - What if outliers correspond to a subpopulation?
  - Should manually examine the putative outliers

Beyond empirical risk

- Recall plug-in principle
  - Want to minimize risk wrt (unavailable) P; use P_n instead
- What if we can't regard data as iid from P?
  - Example: Suppose we know $P = 0.5M + 0.5F$ (*mixture distribution*)
  - We get size n_1 iid sample from M, and size n_2 iid sample from F, $n_2 \ll n_1$
  - How to implement plug-in principle?