CAI 4104/6108 – Machine Learning Engineering:

Linear Models

Prof. Vincent Bindschaedler

Spring 2024

Administrivia

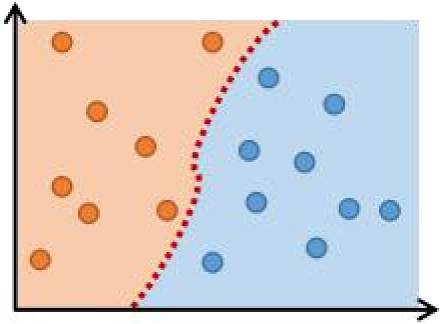
- **Homework 0** has been graded
  - If you lost points, you should be able to see the feedback
    - Note: we are still trying to optimize the process
  - For questions/comments/concerns: please contact the **grading team**

- Future homeworks
  - Important: please do **not** change cell types
    - Changing cell types or adding cells interferes with the grading process

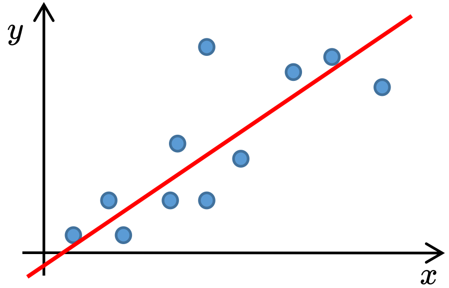
- Reminder: **Homework 1** is due **2/2**

Reminder: Supervised Learning

- Classification
  - Task: predict the corresponding label
  - Different types:
    - Binary classification: there are only two classes (0,1; +,-, etc.)
    - Multiclass: more than two classes
    - Multi-label: each instance can belong to more than one class
    - One-class: there is only one class, we want to distinguish it from everything else



- Regression
  - Task: predict the corresponding value (typically a real number) or target
    - E.g.: you want to predict a person’s future income based on their high school GPA



- Others:
  - Sequence-to-sequence, similarity learning/metric learning, learning to rank, etc.

Reminder: Training and Inference

- Training phrase:
  - We use a **learning algorithm** that takes as input:
    - a **training dataset**
    - a **loss function** or objective function, and
    - a set/class of candidate models
  - to find the one model that **best fits** the training data (under some assumptions)

- Inference phase:
  - We feed **new data** to the model to get predictions
  - Hopefully the predictions of our model are **accurate** which means it **generalizes**

(Some) Learning Theory in 5 Minutes

- General learning framework
  - Learner's input:
    - Domain set: X — stuff we want to label; often represented as a **vector of features**
    - Label set: Y — set of possible labels; e.g. $Y=\{0,1\}$ for binary classification
    - Training data set: $S = (\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)$ — sequence of labeled examples (“points”) for training
  - Learner's output:
    - Model h also called “classifier”, “prediction rule” or “hypothesis”
  - Assumptions:
    - The training data is generated using a distribution D over X
    - There exists a correct labeling function f: $X \rightarrow Y$
  - Performance measures:
    - Error of a classifier is the probability that a random $\boldsymbol{x}$ (sampled according to distribution D) is mislabeled
    - **Generalization error**: $\mathrm{Err}_{D,f}(h) = \Pr_{x\sim D}[\, h(\boldsymbol{x}) \neq f(\boldsymbol{x}) \,]$

(Some) Learning Theory in 5 Minutes

- How do we train the model (i.e., how to select h)?
  - We have to select h from a *hypothesis class* $\mathcal{H}$*, i.e.,* $h \in \mathcal{H}$
  - We also need to select h given the training data set *S*
  - Observation*:* we cannot compute the ***generalization error*** of h. Why?
    - We don't know the distribution *D* over *X*
  - However, we can compute the *training error*
    - *Training error* (also called *empirical error or empirical risk*): $Err_S(h) = Pr_{i \sim [n]}[\, h(\boldsymbol{x_i}) \neq y_i \,]$
    - In other words, the training error is the proportion of training data examples mislabeled by h
  - ***Empirical Risk Minimization*** (ERM):
    - Select h to minimize the training error, i.e.: $h^* = \text{argmin}_{h \in \mathcal{H}}\, Err_S(h)$
  - What could go wrong here? **Overfitting!**

Concrete Formulation

- Supervised Learning ERM
  - Dataset D = $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)$
  - The model to be trained is a function f that is parameterized by **parameters $\boldsymbol{\theta}$**
    - $f_\theta(\boldsymbol{x})$ is the predicted label (or a **probability distribution** over possible labels)
  - Training means finding the ***best*** parameters $\boldsymbol{\theta}$ given the dataset D and a **loss function** $l()$

$$\boxed{\boldsymbol{\theta}} = \operatorname{argmin}_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} \underbrace{l(f_{\boldsymbol{\theta}}(x_i), y_i)}_{\text{loss on example } i}$$

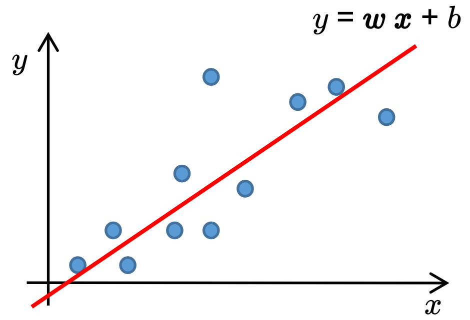
parameters that minimize the average loss over the dataset D

loss on example i

What about the loss of the model on new unseen data?

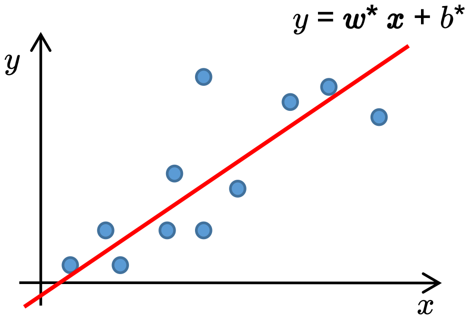
Linear Regression

- Dataset
  - Matrix $\mathbf{X}$ (n × m) and the target vector $\mathbf{y}$ (n × 1)
    - Let $\boldsymbol{x}_i$ be the feature vector for example i and $y_i \in \mathbb{R}$ is the corresponding target/value
- Prediction task:
  - Given the feature vector $\boldsymbol{x}$, predict the target/value $y \in \mathbb{R}$ as accurately as possible
    - For example: given college GPA, predict future yearly income in USD
- Linear Regression:
  - The model is: $h_{w,b}(\boldsymbol{x}) = \boldsymbol{w}\,\boldsymbol{x} + b$
    - Here $\boldsymbol{w}\,\boldsymbol{x}$ is the dot-product of $\boldsymbol{w}$ and $\boldsymbol{x}$
      - i.e.: $\boldsymbol{w}\,\boldsymbol{x} = w_1x_1+w_2x_2+\ldots+w_mx_m$
    - The prediction is: $y' = h_{w,b}(\boldsymbol{x})$
  - Training the model means finding the optimal parameters $\theta^* = (\boldsymbol{w}^*, b^*)$
    - What does *optimal* mean?



Linear Regression

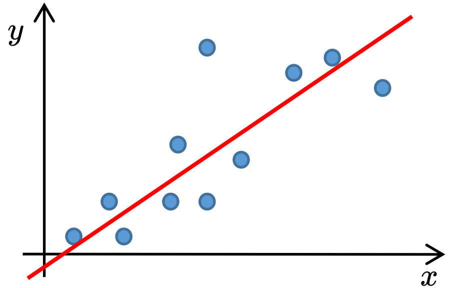
- Dataset
  - Matrix $\mathbf{X}$ (n × m) and the target vector $\mathbf{y}$ (n × 1)
    - Let $\boldsymbol{x}_i$ be the feature vector for example i and $y_i \in \mathbb{R}$ is the corresponding target/value
- Prediction task:
  - Given a feature vector $\boldsymbol{x}$, predict the target/value $y \in \mathbb{R}$ as accurately as possible
- Linear Regression:
  - The model is: $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = h_{\boldsymbol{w},b}(\boldsymbol{x}) = \boldsymbol{w}\,\boldsymbol{x} + b$
  - The prediction is: $y = h_{\boldsymbol{\theta}}(\boldsymbol{x})$
- Training:
  - We want to minimize the Mean Squared Error (MSE) [this is called OLS]
    - $\text{MSE}(\boldsymbol{w},b) := \text{MSE}(h_{\boldsymbol{w},b}, \boldsymbol{X}, \boldsymbol{y}) = 1/n \sum_i [h_{\boldsymbol{w},b}(\boldsymbol{x}_i) - y_i]^2 = 1/n \sum_i [\boldsymbol{w}\,\boldsymbol{x}_i + b - y_i]^2$
  - Optimal parameters: $\boldsymbol{\theta}^* = (\boldsymbol{w}^*, b^*) = \text{argmin}_{\boldsymbol{w},b}\, \text{MSE}(\boldsymbol{w},b)$
  - Remark: MSE is the *expected* squared error loss
    - Squared Error Loss (L_2 loss): $L(\boldsymbol{\theta}) = [y - h_{\boldsymbol{\theta}}(\boldsymbol{x})]^2$



Linear Regression

- Training:
  - In Linear Regression we want to minimize the Mean Squared Error (MSE) [this is called OLS]
  - Q: Why the squared error? Why not the sum of absolute value of error?
    - Why not minimize $L(\boldsymbol{\theta}) = |h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y|$ instead?
    - This is called least absolute deviation (LAD) regression
      - Pro: more robust to outliers
      - Con: mathematically less convenient than OLS!

- Linear regression (i.e., OLS) has a *closed-form* solution!
  - Normal equation: $\boldsymbol{\theta}^* = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$
  - Do we need to use the normal equation? No!
    - We could use some optimization procedure (e.g., gradient descent)

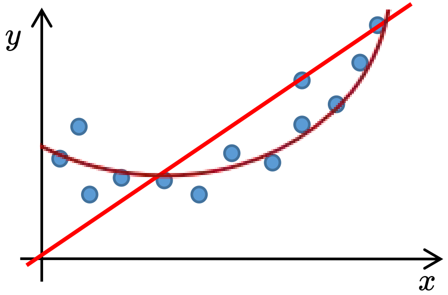


Linear Regression: Stepping Back

- What:
  - An algorithm to solve **regression** problems
  - Models the relationship between input (features) and output (target) as **linear**
  - Objective function: minimize **least squares error** (OLS)
- Advantages:
  - Simple algorithm; fast to train; optimization problem has a closed-form solution
  - Interpretable and explainable model (model parameters aka "*coefficients*")
- Disadvantages:
  - Susceptible to **underfitting**/**high bias**, especially for high-dimensional data.
  - Highly **sensitive** to outliers (e.g., noise or poor quality data)

Polynomial Regression

- What if the data is non-linear?
  - Then a linear model won't fit (it will have high bias)
- Can we still use linear regression?
  - Yes, we can fit a linear model on non-linear data!
  - How? Add features that can capture non-linearity!
  - Example: suppose we have a single feature
    - The linear regression model is: $h_\theta(\boldsymbol{x}) = wx + b$
    - If we add x^2 as a feature, then the model is: $h_\theta(\boldsymbol{x}) = w_1 x + w_2 x^2 + b$



- Polynomial regression
  - If we have several features, say x, y, z, then we can consider all combinations of features up to some degree. That is:
    - x^3, y^3, z^3, x^2y, x^2z, y^2x, y^2z, z^2x, z^2y, xyz (and x, y, z, 1)
  - Q: If we have m features and want all combinations up to degree k, how many features do we get?
    - $m+k$ choose k: $C(m+k, k) = (m+k)! / (m!\, k!)$

Next Time

- Wednesday (1/31): Logistic Regression & SVM

- Upcoming:
  - Homework 1 is due 2/2 by 11:59pm