A brief intro to Machine Learning

Daniel Eftekhari
daniel.eftekhari@ibm.com

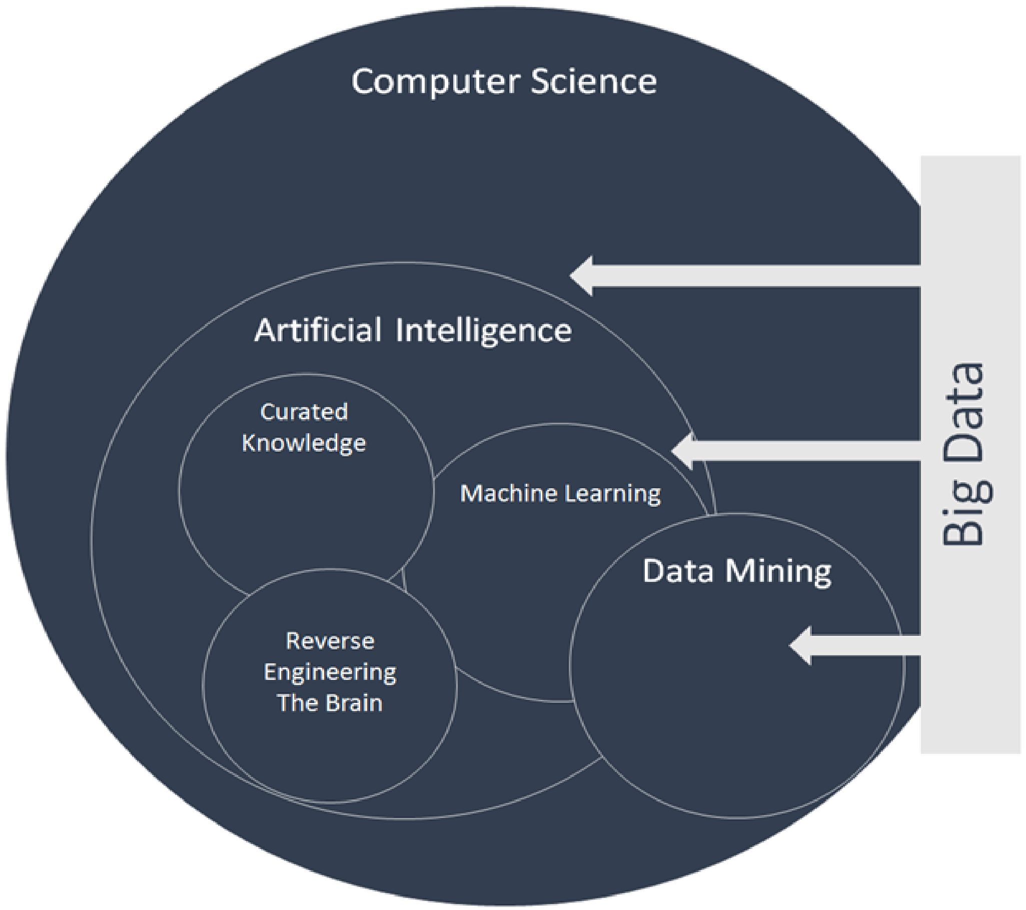
IBM

Today

- Essential ML concepts
- Introduction to two ML algorithms
- How to evaluate ML algorithms

What is Machine Learning

- *"... algorithms that can learn from and make predictions on data ..."* (Wikipedia)
- Statistical pattern recognition



[1]

[1] https://inovancetech.com/buzzwords.html

Machine Learning Subcategories

- Supervised Learning
  - Learn relation between data and known labels/outcomes
  - $X \rightarrow Y$
- Unsupervised Learning
  - Learn patterns in data – no labels/outcomes given
- Reinforcement Learning
  - Learn what actions maximize reward
  - Ex: robot-training, understanding biological decision-making

Today’s focus – Supervised Learning

- Brief introduction to
  - Linear regression
  - Logistic regression
  - Validating results

- Knowledge of the above is good preparation for future encounters with supervised learning

Before we start

- Some general advice:
  - Occam's razor: *"Among competing hypotheses, the one with the fewest assumptions should be selected."* (Wikipedia)

  - Be wary: very easy to treat ML algorithms as black boxes
    - Particularly true as there are libraries dedicated to making life easy in many programming languages
    - Invariably leads to sub-optimal algorithm selection and/or design

  - Algorithm of choice is almost always problem-specific
    - Before applying ML algorithms
      - Understand the nature of your data -> how was it obtained?
      - Visualize your data -> is there a class bias in your data?
      - Develop baseline models -> will serve as a metric when using more advanced algorithms

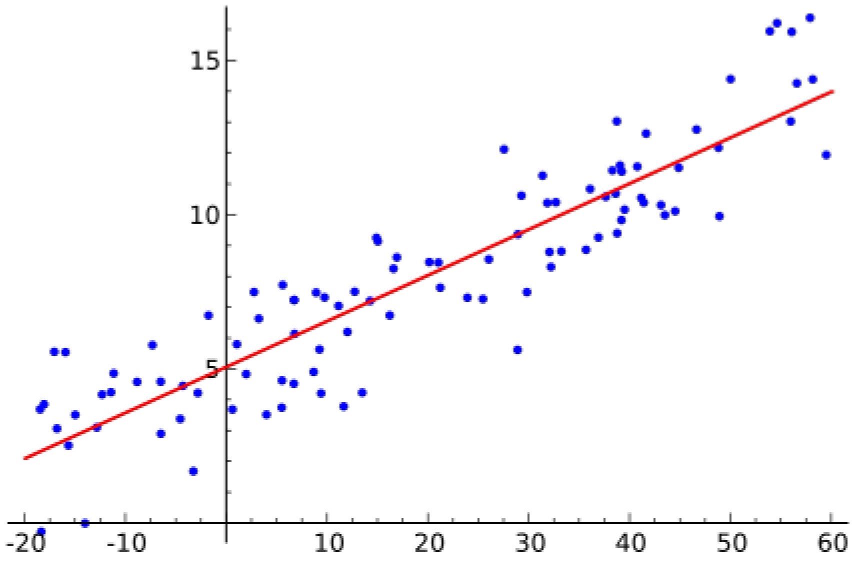
Linear Regression

Linear Regression - Essentials

- Used for prediction
  - Maps input to continuous outputs
    - ex: house prices, customer ratings
  - $\boldsymbol{y} \rightarrow$ Known label

- $\widehat{\boldsymbol{y}} = \boldsymbol{X\beta} + \boldsymbol{\epsilon}$
  - $\widehat{y}_i \rightarrow dependent\ variable$
  - $x_{i1} \ldots x_{im} \rightarrow independent\ variables$
  - $\boldsymbol{\beta} \rightarrow parameter\ vector$
  - $\epsilon_i \rightarrow noise$

- $\widehat{y}_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \cdots + \beta_m x_{i,m} + \epsilon_i$

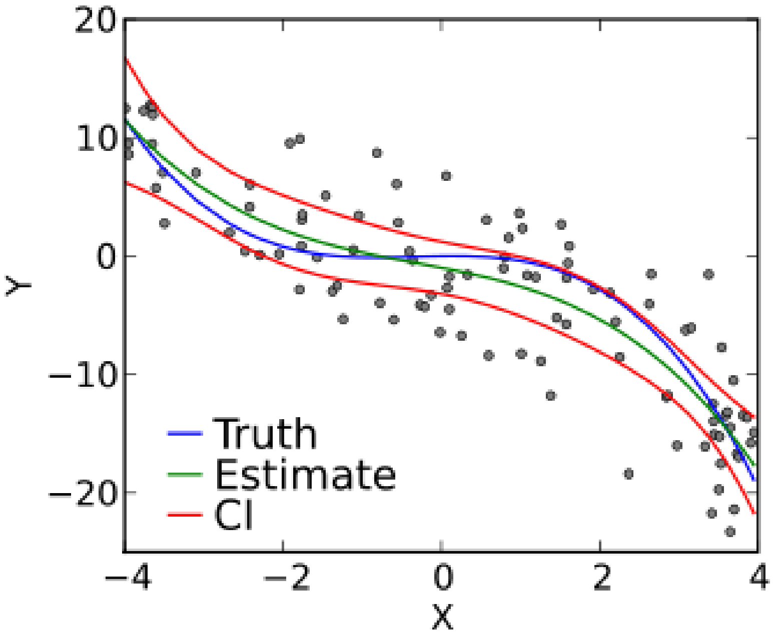


[1]

[1] https://en.wikipedia.org/wiki/Linear_regression

Linear Regression - Essentials

- Key Assumptions of Linear Regression
  - Linearity (not in the way you think)
    - Linear in $\boldsymbol{\beta}$
    - $\hat{y}_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_m x_i^m + \epsilon_i$
  - Constant variance in error (homoscedasticity)
    - Often a flawed assumption
  - Linear independence of predictors



[1]

[1] https://en.wikipedia.org/wiki/Linear_regression

Linear Regression - Optimization

- Loss Function
  - Metric for evaluating how well model fits data

- Ordinary Least Squares
  - $loss = \frac{1}{2}(\boldsymbol{y} - \widehat{\boldsymbol{y}})^2$

Linear Regression – Analytical Solution

- Minimize loss w.r.t. $\boldsymbol{\beta}$
  - $loss = \frac{1}{2}(\boldsymbol{y} - \widehat{\boldsymbol{y}})^2$
  - $loss = \frac{1}{2}(\boldsymbol{y} - \boldsymbol{X\beta})^2$
  - $\frac{\partial loss}{\partial \boldsymbol{\beta}} = \frac{\partial}{\partial \boldsymbol{\beta}}(\frac{1}{2}(\boldsymbol{y} - \boldsymbol{X\beta})^2)$
  - $0 = -\boldsymbol{X}^{\boldsymbol{T}}(\boldsymbol{y} - \boldsymbol{X\beta})$
  - $\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y} = \boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X\beta}$
  - $\boldsymbol{\beta} = \left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{-\mathbf{1}}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y}$

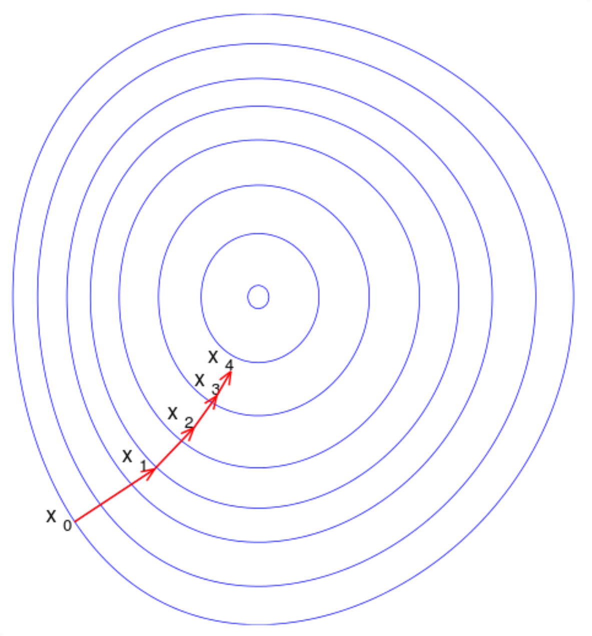
- Problem:
  - Computational complexity: $O(M^2N)$
    - M = number of features, N = number of samples

Linear Regression – Gradient Descent

- Algorithm
  1. Initialize $\boldsymbol{\beta}$ randomly
  2. Repeat until convergence

$$\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \lambda \frac{\partial loss}{\partial \boldsymbol{\beta}}$$

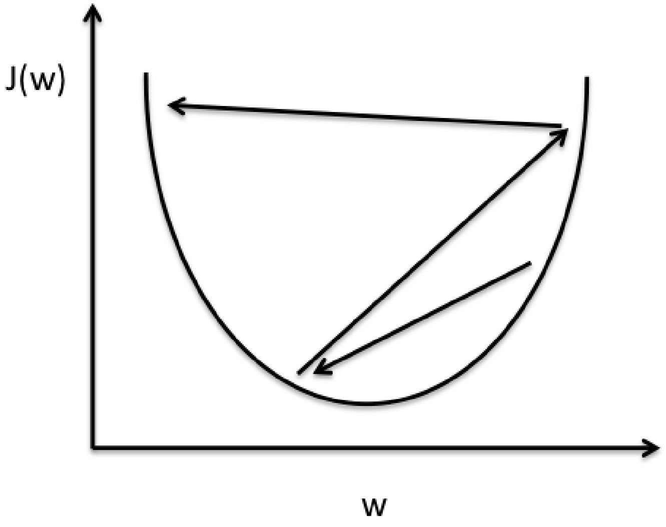
- λ = learning rate $(0 < \lambda < 1)$
  - Too low:
    - slow to converge
    - trapped in local minima
  - Too high
    - divergence



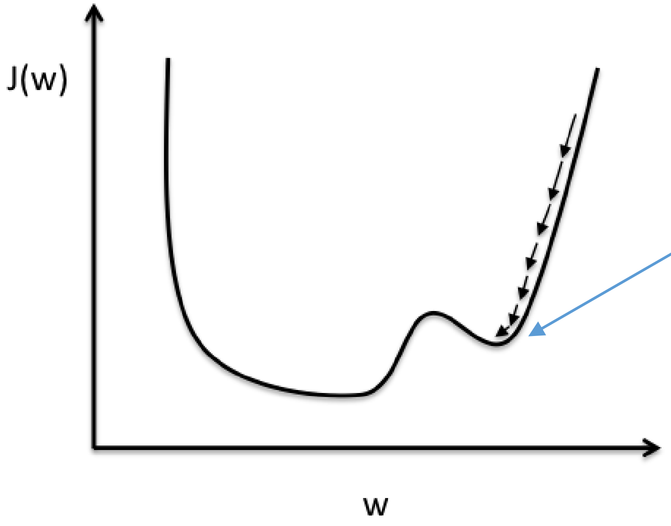
[1]

[1] https://en.wikipedia.org/wiki/Gradient_descent

Linear Regression – Gradient Descent



Large learning rate: Overshooting.



Small learning rate: Many iterations until convergence and trapping in local minima.

Can use momentum to overcome.

[1]

[1] http://sebastianraschka.com/Articles/2015_singlelayer_neurons.html

Linear Regression – Gradient Descent

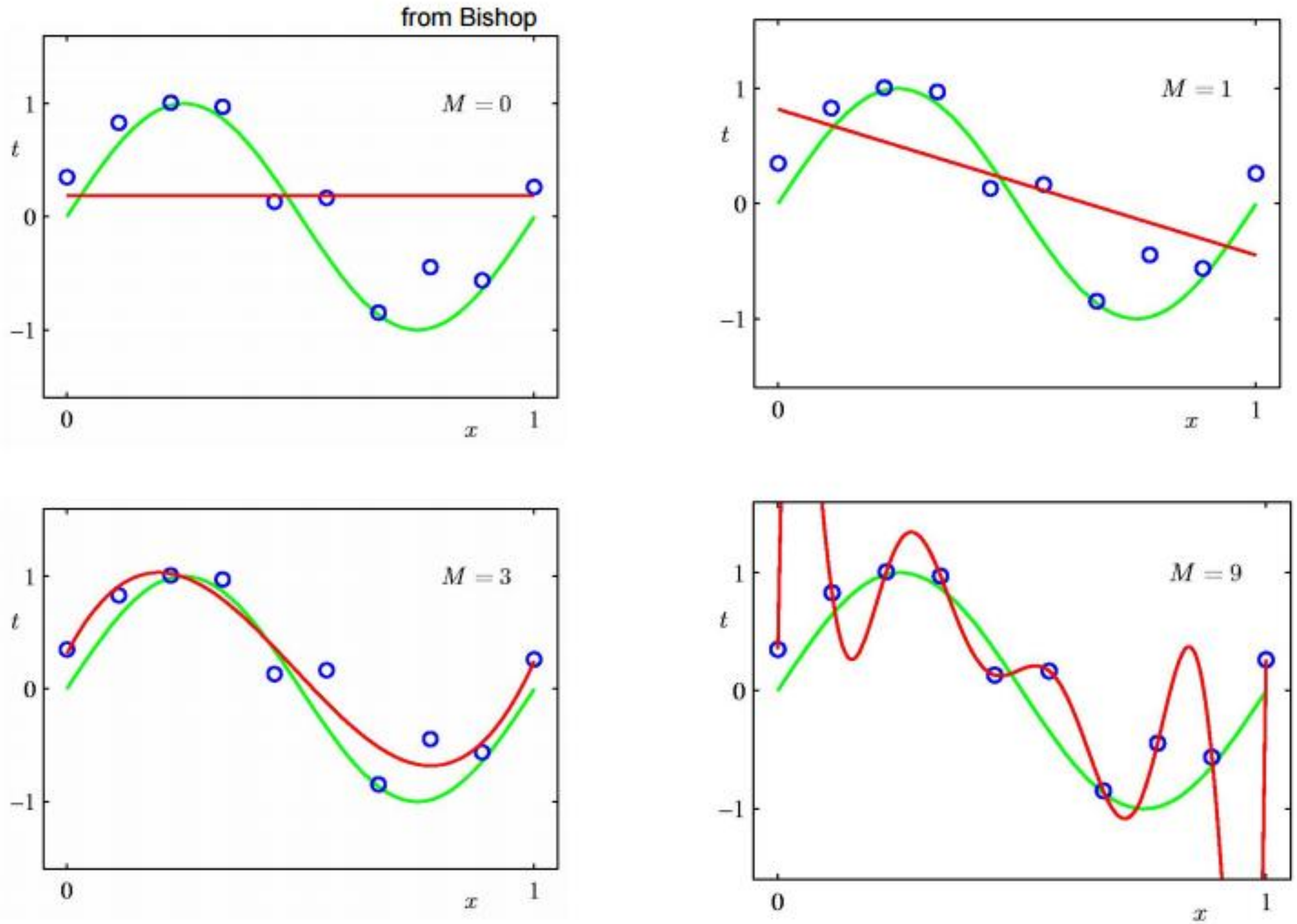
- Batch updates:
  - $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + \lambda \sum_{n=1}^{N} [(y^{(n)} - \hat{y}\,(x^{(n)}))x^{(n)}]$

- Stochastic Gradient Descent
  - Computationally faster (do not need to hold entire dataset in memory)
  - Update weights for each training case
  - $for\ i = 1\ to\ N$

    $$\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + \lambda(y^{(n)} - \hat{y}\,(x^{(n)})\,x^{(n)}$$
  - Can use mini-batches: balance between performance and speed

Linear Regression – Under/Overfitting

- Underfitting
  - Model does not capture complexity of problem
- Overfitting
  - Model is too powerful – does not generalize to unseen data



[1]

[1] Zemel, Urtasun, Fidler (UofT), CSC411 Fall 2016, taken from Bishop, Pattern Recognition and Machine Learning, 2006

Linear Regression – Regularization

- Problem
  - As we increase model power, weights increase in magnitude to compensate for noise
- Solution: Regularization
  - Restrict magnitude of $\boldsymbol{\beta}$ by penalizing large weights
- Ridge regression
  - $loss = \frac{1}{2}(\boldsymbol{X\beta} - \boldsymbol{y})^2 + \frac{\alpha}{2}\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{\beta}$
  - $\alpha =$ regularization term $(0 < \alpha < 1)$

IBM

Linear Regression – Regularization

- Analytical solution:
  - $\boldsymbol{\beta} = \left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X} + \alpha\boldsymbol{I}\right)^{-\mathbf{1}}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y}$

- Gradient descent:
  - $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + \lambda \sum_{n=1}^{N}[(y^{(n)} - \hat{y}\,(x^{(n)}))x^{(n)} - \alpha\boldsymbol{\beta}]$

Linear Regression – Key Concepts

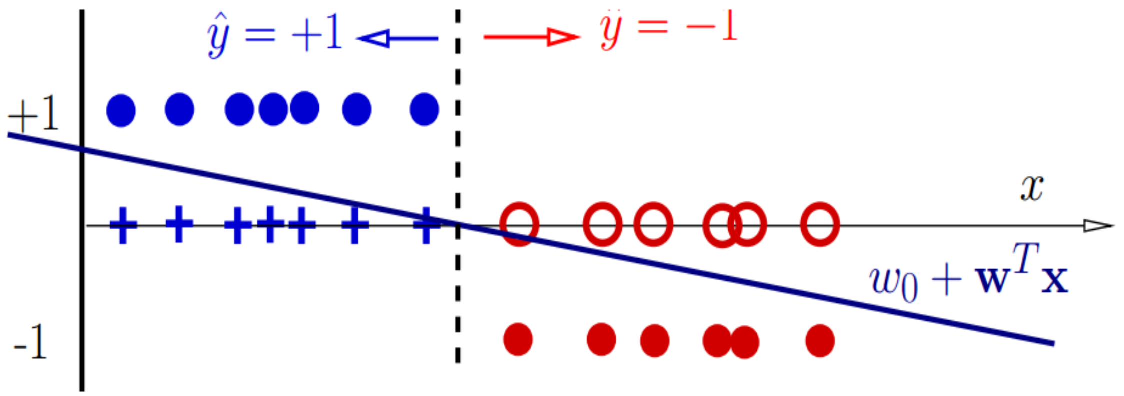
- Used for prediction
- Model assumptions
- Loss function
- Analytical solution vs. gradient descent
- Under/overfitting
- Hyperparameter tuning

Logistic Regression

Logistic Regression - Essentials

- Used for classification
  - Maps input to binary output
    - ex: medical diagnostics
  - We will focus on binary classification

- Linear regression (blue line) isn't a good solution
  - The sign function (dotted line), is more appropriate

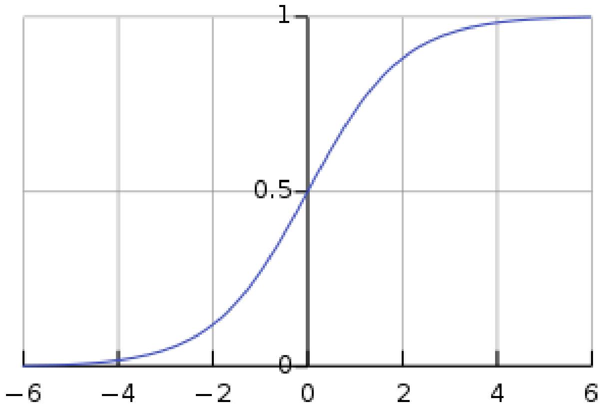


[1] Zemel, Urtasun, Fidler (UofT), CSC411 Fall 2016

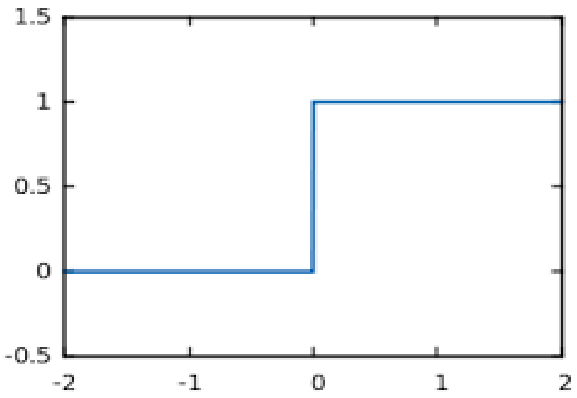
[1]

Logistic Regression – Sigmoid Function

- The sign function is an extreme case of the sigmoid function
  - Sigmoid properties:
    - probabilistic outputs
    - continuous and differentiable everywhere
    - maintains classification property
- $\sigma(z) = \frac{1}{1+\exp(-z)}$



[1]



[2]

[1] https://en.wikipedia.org/wiki/Sigmoid_function

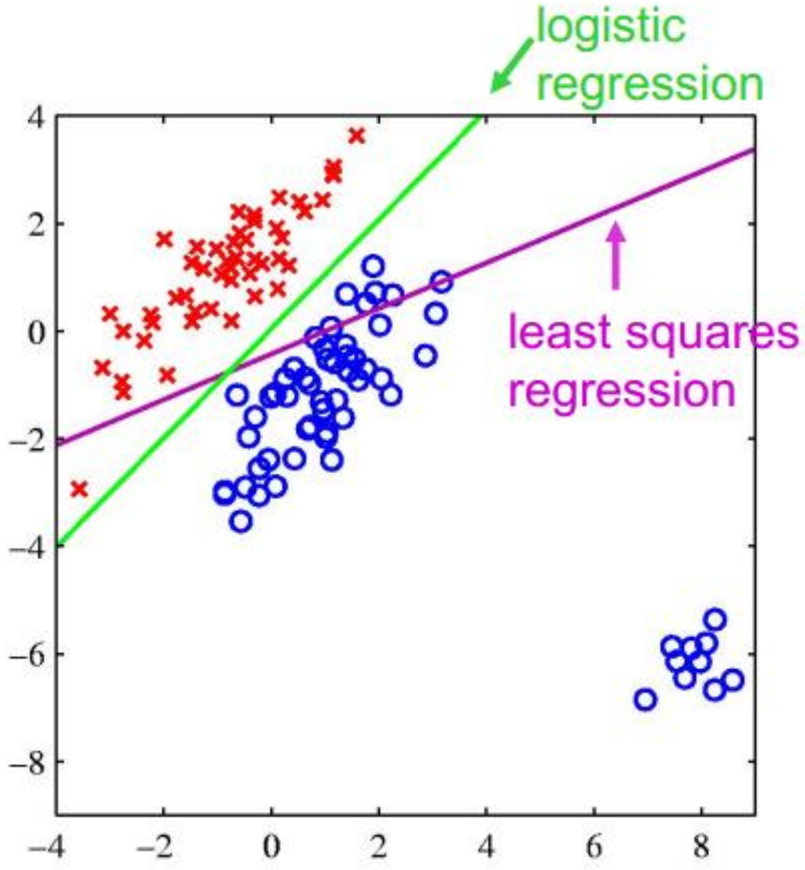
[2] http://www.gnuplotting.org/defining-piecewise-functions/

Logistic Regression – Binary Classification

- $\hat{y}(\boldsymbol{x}) = \sigma(\boldsymbol{\beta}^T \boldsymbol{x} + \beta_0)$
  - $p(C = 0|\boldsymbol{x}) = \sigma(\boldsymbol{\beta}^T \boldsymbol{x} + \beta_0)$
  - $p(C = 1|\boldsymbol{x}) = 1 - \sigma\left(\boldsymbol{\beta}^T \boldsymbol{x} + \beta_0\right)$

- Decision boundary:
  - $\boldsymbol{\beta}^T \boldsymbol{x} + \beta_0 = 0$

- Linear Regression
  - sensitive to outliers



[1]

[1] Zemel, Urtasun, Fidler (UofT), CSC411 Fall 2016

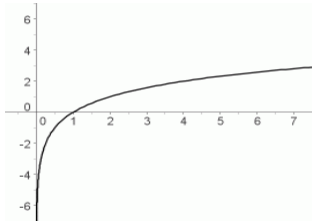
Logistic Regression – Optimization

- Assume training examples are sampled I.I.D. (Independent and Identically Distributed)
- Likelihood Function: $L(\boldsymbol{\beta}) = \prod_{i=1}^{N} p(y^{(i)} | \boldsymbol{x}^{(i)}; \boldsymbol{\beta})$
- Define the loss function as the negative log of the likelihood function, and use gradient descent to solve for optimal $\boldsymbol{\beta}$
  - Negative to make it a minimization problem
  - Log for numerical reasons
- Tune learning rate and regularization parameters

Logistic Regression – Optimization

- $p\left(y^{(i)} \middle| \boldsymbol{x}^{(i)}; \boldsymbol{\beta}\right) = \left(1 - p\left(\hat{y} = 0 \middle| \boldsymbol{x}^{(i)}; \boldsymbol{\beta}\right)\right)^{y^{(i)}} \left(p\left(\hat{y} = 0 \middle| \boldsymbol{x}^{(i)}; \boldsymbol{\beta}\right)\right)^{1-y^{(i)}}$
- $L(\boldsymbol{\beta}) = \prod_{i=1}^{N} \left(1 - p\left(\hat{y} = 0 \middle| \boldsymbol{x}^{(i)}; \boldsymbol{\beta}\right)\right)^{y^{(i)}} \left(p\left(\hat{y} = 0 \middle| \boldsymbol{x}^{(i)}; \boldsymbol{\beta}\right)\right)^{1-y^{(i)}}$
- $loss(\boldsymbol{\beta}) = -\log(L(\boldsymbol{\beta}))$
- $loss(\boldsymbol{\beta}) = -\sum_{i=1}^{N}(y^{(i)}) \log\left(1 - p\left(\hat{y} = 0 \middle| \boldsymbol{x}^{(i)}; \boldsymbol{\beta}\right)\right) - \sum_{i=1}^{N}(1 - y^{(i)}) \log\left(p\left(\hat{y} = 0 \middle| \boldsymbol{x}^{(i)}; \boldsymbol{\beta}\right)\right)$
- $loss(\boldsymbol{\beta}) = \sum_{i=1}^{N} \log\left(1 + \exp(-z^{(i)})\right) + \sum_{i=1}^{N} y^{(i)} z^{(i)}$
  - Where $z = \boldsymbol{\beta}^{\boldsymbol{T}} \boldsymbol{x} + \beta_0$

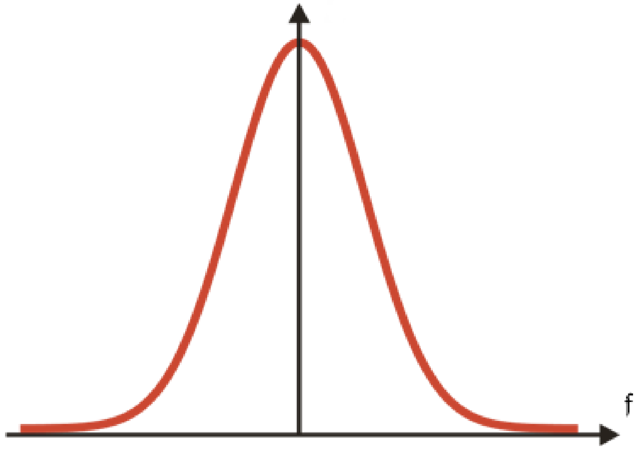
- Convex function in $\boldsymbol{\beta}$ – therefore we should be able to find the global optimum
  - $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \lambda \frac{\partial loss}{\partial \boldsymbol{\beta}}$



Logistic Regression – Regularization

- Define priors over the weights
  - $\max\left\{\log\left[p(\boldsymbol{\beta})\prod_{i=1}^{N}p\left(y^{(i)}\middle|\boldsymbol{x}^{(i)};\boldsymbol{\beta}\right)\right]\right\}$ w.r.t. $\boldsymbol{\beta}$, where $p(\boldsymbol{\beta}) = N(0, \alpha^{-1}\boldsymbol{I})$

- Prior biases the weights towards zero -> prevents weights from growing too large



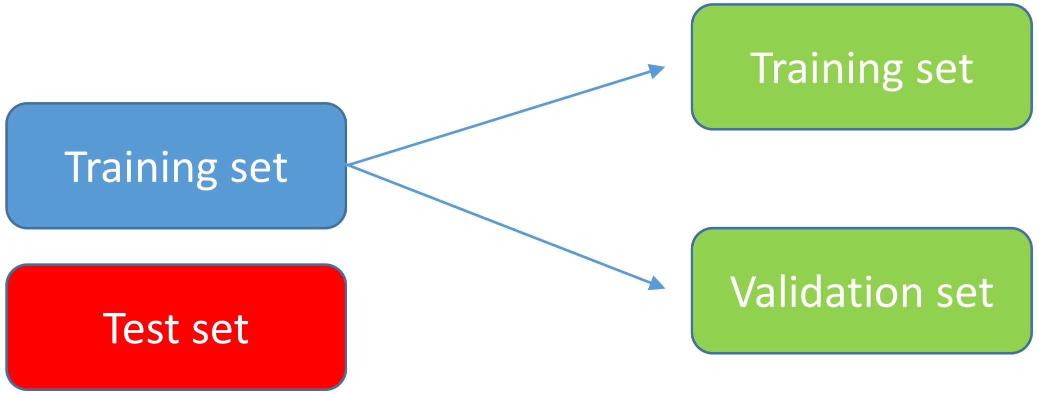
Classification – Other Algorithms

- Popular ones
  - Decision Trees
    - Random Forests -> Xbox Kinect!
  - k-Nearest Neighbour (kNN)
    - Recommender Systems
  - Naïve Bayes
    - Spam Detection
  - Neural Networks
    - Image & Speech Recognition, Machine Translation
  - Support Vector Machines (SVM)
    - Similar use cases as neural networks
  - Mixture of Experts
    - Combines decisions of different algorithms
    - Netflix recommender system!

Validating Results

Cross Validation – Hyperparameter Tuning

- Need 3 things
  - Training set
  - Validation set
  - Test set



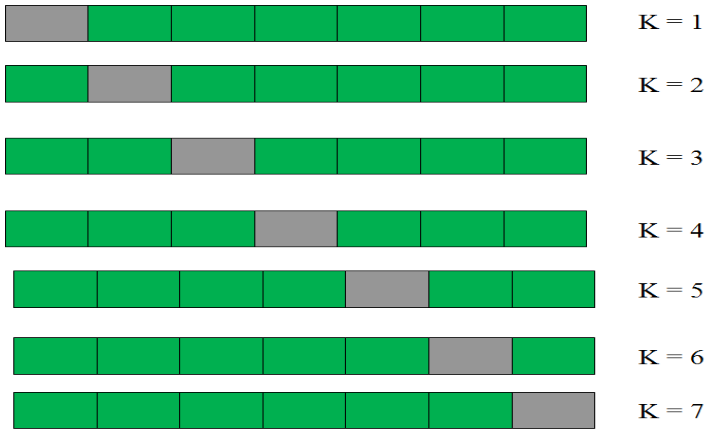
- Use training set to optimize $\boldsymbol{\beta}$
- Use validation set to optimize for hyperparameters (ex: λ, α)
- Use test set to evaluate results on unseen data set
  - DO NOT over-test on the test set

Cross Validation - Techniques

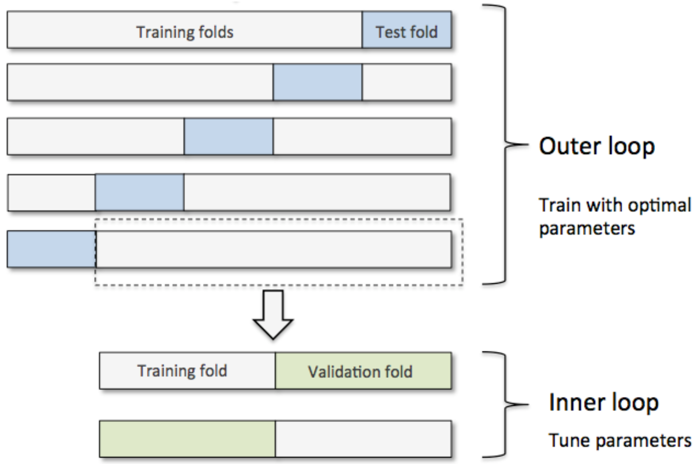
- Widely used: k-fold cross validation
  - Increases data set utility
  - Only used to evaluate a single model's performance, not to tune hyperparameters… (see below)

- Alternative: nested k-fold cross validation

[1]



[2]



[1] https://www.analyticsvidhya.com/blog/2016/02/7-important-model-evaluation-error-metrics/

[2] https://sebastianraschka.com/faq/docs/evaluate-a-model.html

Metrics for Binary Classification

- Sensitivity: proportion of positives correctly identified
- Specificity: proportion of negatives correctly identified

- $Sensitivity = Recall = \frac{TP}{TP+FN} = \frac{TP}{all\ ground\ truth\ positive\ cases}$

- $Specificity = \frac{TN}{TN+FP} = \frac{TN}{all\ ground\ truth\ negative\ cases}$

- $Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = \frac{all\ correct\ predictions}{all\ predictions\ made}$

IBM

Metrics for Binary Classification

- Recall: fraction of true events that were detected
- Precision: fraction of detections that were true events

- $Recall = \frac{TP}{TP+FN} = \frac{TP}{all\ ground\ truth\ positive\ cases}$

- $Precision = \frac{TP}{TP+FP} = \frac{TP}{all\ cases\ predicted\ as\ positive}$

- $F1 = 2\frac{P*R}{P+R}$

Conclusions

- We examined two supervised learning algorithms
  - Linear Regression
  - Logistic Regression
- We examined how to build ML algorithms from the ground up
- We saw that in supervised learning, we use ML algorithms to find $\boldsymbol{\beta}$
  - It's still up to us to acquire input data $\boldsymbol{X}$ and ground truth labels $\boldsymbol{y}$
- We examined how to validate ML models and how to measure performance