

Introduction to Deep Learning



Arijit Mondal

Dept. of Computer Science & Engineering

Indian Institute of Technology Patna

`arijit@iitp.ac.in`

Feature Engineering

Machine Learning

- A form of applied statistics with
 - **Increased** emphasis on the use of computers to statistically estimate complicated function
 - **Decreased** emphasis on proving confidence intervals around these functions
- Two primary approaches
 - Frequentist estimators
 - Bayesian inference

Types of Machine Learning Problems

- Supervised
- Unsupervised
- Other variants
 - Reinforcement learning
 - Semi-supervised

Learning algorithm

- A ML algorithm is an algorithm that is able to learn from data
- Mitchell (1997)
 - A computer program is said to learn from experience E with respect to some class of task T and performance measure P , if its performance at task in T as measured by P , improves with experience E .

Task

- A ML task is usually described in terms of how ML system should process an example
 - Example is a collection of features that have been quantitatively measured from some objects or events that we want the learning system process
 - Represented as $x \in \mathbb{R}^n$ where x_i is a feature
 - Feature of an image — pixel values

Common ML Task

- Classification

- Need to predict which of the k categories some input belongs to
- Need to have a function $f: \mathbb{R}^n \rightarrow \{1, 2, \dots, k\}$
- $y = f(x)$ input x is assigned a category identified by y
- Examples
 - Object identification
 - Face recognition

- Regression

- Need to predict numeric value for some given input
- Need to have a function $f: \mathbb{R}^n \rightarrow \mathbb{R}$
- Examples
 - Energy consumption
 - Amount of insurance claim

Common ML Task (contd.)

- Classification with missing inputs
 - Need to have a set of functions
 - Each function corresponds to classifying x with different subset of inputs missing
 - Examples
 - Medical diagnosis (expensive or invasive)

Common ML Task (contd.)

- Classification with missing inputs
 - Need to have a set of functions
 - Each function corresponds to classifying x with different subset of inputs missing
 - Examples
 - Medical diagnosis (expensive or invasive)
- Transcription
 - Need to convert relatively unstructured data into discrete, textual form
 - Optical character recognition
 - Speech recognition

Common ML Task (contd.)

- Classification with missing inputs
 - Need to have a set of functions
 - Each function corresponds to classifying x with different subset of inputs missing
 - Examples
 - Medical diagnosis (expensive or invasive)
- Transcription
 - Need to convert relatively unstructured data into discrete, textual form
 - Optical character recognition
 - Speech recognition
- Machine translation
 - Conversion of sequence of symbols in one language to some other language
 - Natural language processing (English to Spanish conversion)

Common ML Task (contd.)

- Structured output
 - Output is a vector with important relationship between the different elements
 - Mapping natural language sentence into a tree that describes grammatical structure
 - Pixel based image segmentation (eg. identify roads)

Common ML Task (contd.)

- Structured output
 - Output is a vector with important relationship between the different elements
 - Mapping natural language sentence into a tree that describes grammatical structure
 - Pixel based image segmentation (eg. identify roads)
- Anomaly detection
 - Observes a set of events or objects and flags if some of them are unusual
 - Fraud detection in credit card

Common ML Task (contd.)

- Structured output
 - Output is a vector with important relationship between the different elements
 - Mapping natural language sentence into a tree that describes grammatical structure
 - Pixel based image segmentation (eg. identify roads)
- Anomaly detection
 - Observes a set of events or objects and flags if some of them are unusual
 - Fraud detection in credit card
- Synthesis and sampling
 - Generate new example similar to past examples
 - Useful for media application
 - Text to speech

Performance measure

- Accuracy is one of the key measures
 - The proportion of examples for which the model produces correct outputs
 - Similar to error rate
 - Error rate often referred as expected 0-1 loss
- Mostly interested how ML algorithm performs on unseen data
- Choice of performance measure may not be straight forward
 - **Transcription**
 - Accuracy of the system at transcribing entire sequence
 - Any partial credit for some elements of the sequence are correct

Experience

- Kind of experience allowed during learning process
 - Supervised
 - Unsupervised

Supervised learning

- Allowed to use labeled dataset
- Example — Iris
 - Collection of measurements of different parts of Iris plant
 - Each plant means each example
 - Features
 - Sepal length/width, petal length/width
 - Also record which species the plant belong to

Supervised learning (contd.)

- A set of labeled examples $\langle x_1, x_2, \dots, x_n, y \rangle$
 - x_i are input variables
 - y output variable
- Need to find a function $f: X_1 \times X_2 \times \dots \times X_n \rightarrow Y$
- Goal is to minimize error/loss function
 - Like to minimize over all dataset
 - We have limited dataset

Unsupervised learning

- Learns useful properties of the structure of data set
- Unlabeled data
 - Tries to learn entire probability distribution that generated the dataset
 - Examples
 - Clustering, dimensionality reduction

Supervised vs Unsupervised learning

- Unsupervised attempts to learn implicitly or explicitly probability distribution of $p(x)$
- Supervised tries to predict y from x ie. $p(y|x)$

Supervised vs Unsupervised learning

- Unsupervised attempts to learn implicitly or explicitly probability distribution of $p(x)$
- Supervised tries to predict y from x ie. $p(y|x)$
- Unsupervised learning can be decomposed as supervised learning

$$p(x) = \prod_{i=1}^n p(x_i | x_1, x_2, \dots, x_{i-1})$$

Supervised vs Unsupervised learning

- Unsupervised attempts to learn implicitly or explicitly probability distribution of $p(x)$
- Supervised tries to predict y from x ie. $p(y|x)$
- Unsupervised learning can be decomposed as supervised learning

$$p(x) = \prod_{i=1}^n p(x_i | x_1, x_2, \dots, x_{i-1})$$

- Solving supervised learning using traditional unsupervised learning

$$p(y|x) = \frac{p(x, y)}{\sum_{y'} p(x, y')}$$

Linear regression

- Prediction of the value of a continuous variable
 - Example — price of a house, solar power generation in photo-voltaic cell, etc.

Linear regression

- Prediction of the value of a continuous variable
 - Example — price of a house, solar power generation in photo-voltaic cell, etc.
- Takes a vector $\mathbf{x} \in \mathbb{R}^n$ and predict scalar $y \in \mathbb{R}$
 - Predicted value will be represented as $\hat{y} = \mathbf{w}^T \mathbf{x}$ where \mathbf{w} is a vector of parameters
 - x_i receives positive weight — Increasing the value of the feature will increase the value of y
 - x_i receives negative weight — Increasing the value of the feature will decrease the value of y
 - Weight value is very high/large — Large effect on prediction

Performance

- Assume, we have m examples not used for training
 - This is known as test set

Performance

- Assume, we have m examples not used for training
 - This is known as **test set**
- Design matrix of inputs is $X^{(\text{test})}$ and target output is a vector $y^{(\text{test})}$
 - Performance is measured by Mean Square Error (MSE)

$$\text{MSE}_{(\text{test})} = \frac{1}{m} \sum_i \left(\hat{y}^{(\text{test})} - y^{(\text{test})} \right)_i^2 = \frac{1}{m} \|\hat{y}^{(\text{test})} - y^{(\text{test})}\|_2^2$$

- Error increases when the Euclidean distance between target and prediction increases

Performance

- Assume, we have m examples not used for training
 - This is known as **test set**
- Design matrix of inputs is $X^{(\text{test})}$ and target output is a vector $y^{(\text{test})}$
 - Performance is measured by Mean Square Error (MSE)

$$\text{MSE}_{(\text{test})} = \frac{1}{m} \sum_i \left(\hat{y}^{(\text{test})} - y^{(\text{test})} \right)_i^2 = \frac{1}{m} \|\hat{y}^{(\text{test})} - y^{(\text{test})}\|_2^2$$

- Error increases when the Euclidean distance between target and prediction increases
- The learning algorithm is allowed to gain experience from training set $(X^{(\text{train})}, y^{(\text{train})})$
- One of the common ideas is to minimize $\text{MSE}_{(\text{train})}$ for training set

Minimization of MSE

- We have the following now

$$\nabla_w \text{MSE}_{(\text{train})} = 0$$

Minimization of MSE

- We have the following now

$$\nabla_w \text{MSE}_{(\text{train})} = 0$$

$$\Rightarrow \nabla_w \frac{1}{m} \|\hat{y}^{(\text{train})} - y^{(\text{train})}\|_2^2 = 0$$

Minimization of MSE

- We have the following now

$$\nabla_w \text{MSE}_{(\text{train})} = 0$$

$$\Rightarrow \nabla_w \frac{1}{m} \|\hat{y}^{(\text{train})} - y^{(\text{train})}\|_2^2 = 0$$

$$\Rightarrow \frac{1}{m} \nabla_w \|\mathbf{X}^{(\text{train})} \mathbf{w} - \mathbf{y}^{(\text{train})}\|_2^2 = 0$$

Minimization of MSE

- We have the following now

$$\nabla_w \text{MSE}_{(\text{train})} = 0$$

$$\Rightarrow \nabla_w \frac{1}{m} \|\hat{y}^{(\text{train})} - y^{(\text{train})}\|_2^2 = 0$$

$$\Rightarrow \frac{1}{m} \nabla_w \|\mathbf{X}^{(\text{train})} \mathbf{w} - \mathbf{y}^{(\text{train})}\|_2^2 = 0$$

$$\Rightarrow \nabla_w (\mathbf{X}^{(\text{train})} \mathbf{w} - \mathbf{y}^{(\text{train})})^T (\mathbf{X}^{(\text{train})} \mathbf{w} - \mathbf{y}^{(\text{train})}) = 0$$

Minimization of MSE

- We have the following now

$$\nabla_w \text{MSE}_{(\text{train})} = 0$$

$$\Rightarrow \nabla_w \frac{1}{m} \|\hat{y}^{(\text{train})} - y^{(\text{train})}\|_2^2 = 0$$

$$\Rightarrow \frac{1}{m} \nabla_w \|\mathbf{X}^{(\text{train})} \mathbf{w} - \mathbf{y}^{(\text{train})}\|_2^2 = 0$$

$$\Rightarrow \nabla_w (\mathbf{X}^{(\text{train})} \mathbf{w} - \mathbf{y}^{(\text{train})})^T (\mathbf{X}^{(\text{train})} \mathbf{w} - \mathbf{y}^{(\text{train})}) = 0$$

$$\Rightarrow \nabla_w (\mathbf{w}^T \mathbf{X}^{(\text{train})}{}^T \mathbf{X}^{(\text{train})} \mathbf{w} - 2\mathbf{w}^T \mathbf{X}^{(\text{train})}{}^T \mathbf{y}^{(\text{train})} + \mathbf{y}^{(\text{train})}{}^T \mathbf{y}^{(\text{train})}) = 0$$

Minimization of MSE

- We have the following now

$$\nabla_w \text{MSE}_{(\text{train})} = 0$$

$$\Rightarrow \nabla_w \frac{1}{m} \|\hat{y}^{(\text{train})} - y^{(\text{train})}\|_2^2 = 0$$

$$\Rightarrow \frac{1}{m} \nabla_w \|\mathbf{X}^{(\text{train})} \mathbf{w} - \mathbf{y}^{(\text{train})}\|_2^2 = 0$$

$$\Rightarrow \nabla_w (\mathbf{X}^{(\text{train})} \mathbf{w} - \mathbf{y}^{(\text{train})})^T (\mathbf{X}^{(\text{train})} \mathbf{w} - \mathbf{y}^{(\text{train})}) = 0$$

$$\Rightarrow \nabla_w (\mathbf{w}^T \mathbf{X}^{(\text{train})} T \mathbf{X}^{(\text{train})} \mathbf{w} - 2 \mathbf{w}^T \mathbf{X}^{(\text{train})} T \mathbf{y}^{(\text{train})} + \mathbf{y}^{(\text{train})} T \mathbf{y}^{(\text{train})}) = 0$$

$$\Rightarrow 2 \mathbf{X}^{(\text{train})} T \mathbf{X}^{(\text{train})} \mathbf{w} - 2 \mathbf{X}^{(\text{train})} T \mathbf{y}^{(\text{train})} = 0$$

Minimization of MSE

- We have the following now

$$\nabla_w \text{MSE}_{(\text{train})} = 0$$

$$\Rightarrow \nabla_w \frac{1}{m} \|\hat{y}^{(\text{train})} - y^{(\text{train})}\|_2^2 = 0$$

$$\Rightarrow \frac{1}{m} \nabla_w \|\mathbf{X}^{(\text{train})} \mathbf{w} - \mathbf{y}^{(\text{train})}\|_2^2 = 0$$

$$\Rightarrow \nabla_w (\mathbf{X}^{(\text{train})} \mathbf{w} - \mathbf{y}^{(\text{train})})^T (\mathbf{X}^{(\text{train})} \mathbf{w} - \mathbf{y}^{(\text{train})}) = 0$$

$$\Rightarrow \nabla_w (\mathbf{w}^T \mathbf{X}^{(\text{train})} T \mathbf{X}^{(\text{train})} \mathbf{w} - 2 \mathbf{w}^T \mathbf{X}^{(\text{train})} T \mathbf{y}^{(\text{train})} + \mathbf{y}^{(\text{train})} T \mathbf{y}^{(\text{train})}) = 0$$

$$\Rightarrow 2 \mathbf{X}^{(\text{train})} T \mathbf{X}^{(\text{train})} \mathbf{w} - 2 \mathbf{X}^{(\text{train})} T \mathbf{y}^{(\text{train})} = 0$$

$$\Rightarrow \mathbf{w} = (\mathbf{X}^{(\text{train})} T \mathbf{X}^{(\text{train})})^{-1} \mathbf{X}^{(\text{train})} T \mathbf{y}^{(\text{train})}$$

Minimization of MSE

- We have the following now

$$\nabla_w \text{MSE}_{(\text{train})} = 0$$

$$\Rightarrow \nabla_w \frac{1}{m} \|\hat{y}^{(\text{train})} - y^{(\text{train})}\|_2^2 = 0$$

$$\Rightarrow \frac{1}{m} \nabla_w \|\mathbf{X}^{(\text{train})} \mathbf{w} - \mathbf{y}^{(\text{train})}\|_2^2 = 0$$

$$\Rightarrow \nabla_w (\mathbf{X}^{(\text{train})} \mathbf{w} - \mathbf{y}^{(\text{train})})^T (\mathbf{X}^{(\text{train})} \mathbf{w} - \mathbf{y}^{(\text{train})}) = 0$$

$$\Rightarrow \nabla_w (\mathbf{w}^T \mathbf{X}^{(\text{train})} T \mathbf{X}^{(\text{train})} \mathbf{w} - 2 \mathbf{w}^T \mathbf{X}^{(\text{train})} T \mathbf{y}^{(\text{train})} + \mathbf{y}^{(\text{train})} T \mathbf{y}^{(\text{train})}) = 0$$

$$\Rightarrow 2 \mathbf{X}^{(\text{train})} T \mathbf{X}^{(\text{train})} \mathbf{w} - 2 \mathbf{X}^{(\text{train})} T \mathbf{y}^{(\text{train})} = 0$$

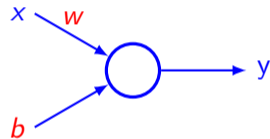
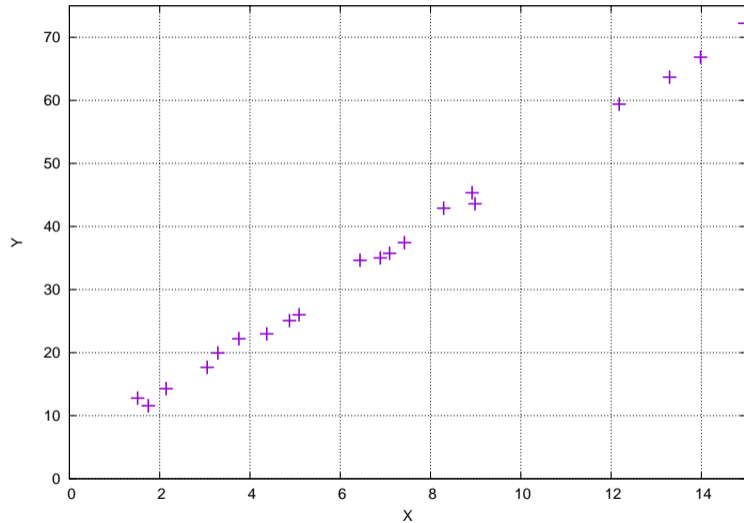
$$\Rightarrow \mathbf{w} = (\mathbf{X}^{(\text{train})} T \mathbf{X}^{(\text{train})})^{-1} \mathbf{X}^{(\text{train})} T \mathbf{y}^{(\text{train})}$$

- Linear regression with bias term $\hat{y} = [\mathbf{w}^T \quad w_0][\mathbf{x} \quad 1]^T$

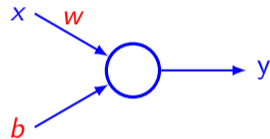
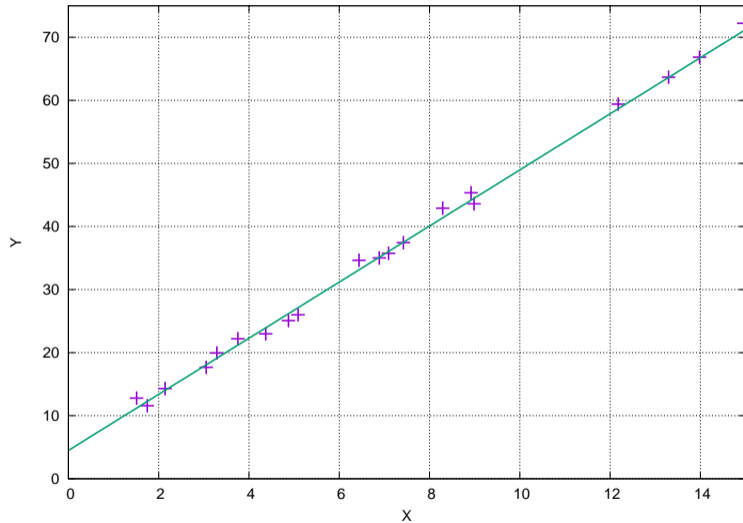
Moore-Penrose Pseudoinverse

- Let $A \in \mathbb{R}^{n \times m}$
- Every A has pseudoinverse $A^+ \in \mathbb{R}^{m \times n}$ and it is unique
 - $AA^+A = A$
 - $A^+AA^+ = A^+$
 - $(AA^+)^T = AA^+$
 - $(A^+A)^T = A^+A$
- $A^+ = \lim_{\alpha \rightarrow 0} (A^T A + \alpha I)^{-1} A^T$
- Example
 - If $A = [1 \ 2]^T$ then $A^+ = [\frac{1}{5} \ \frac{2}{5}]$
 - If $A = \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 1 & 5 \end{bmatrix}$ then $A^+ = \begin{bmatrix} 0.121212 & 0.515152 & -0.151515 \\ 0.030303 & -0.121212 & 0.212121 \end{bmatrix}$

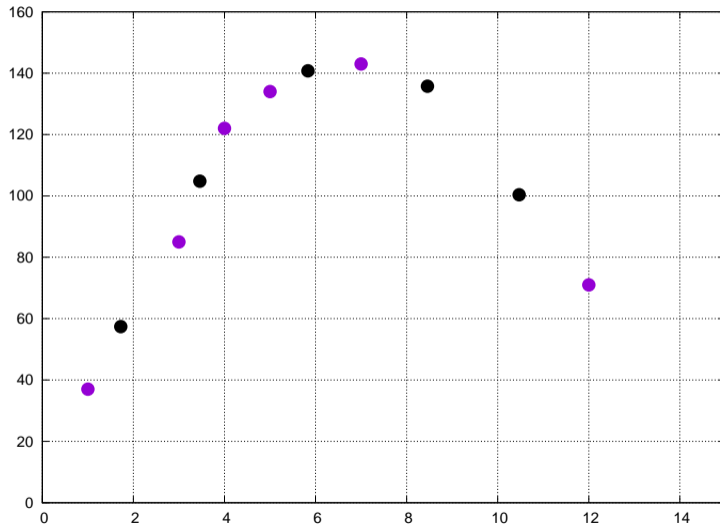
Regression example



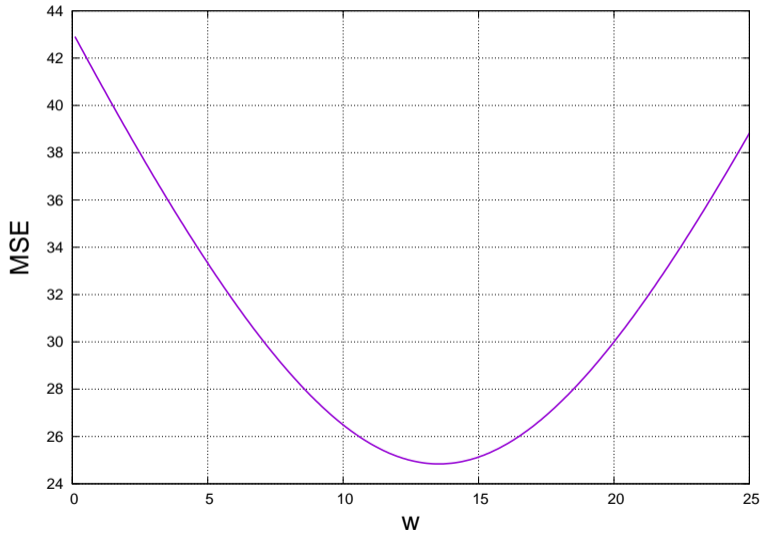
Regression example



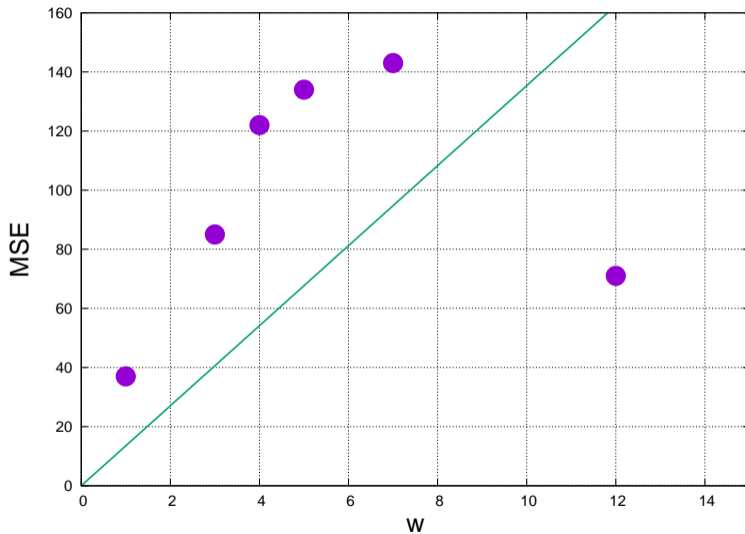
Example



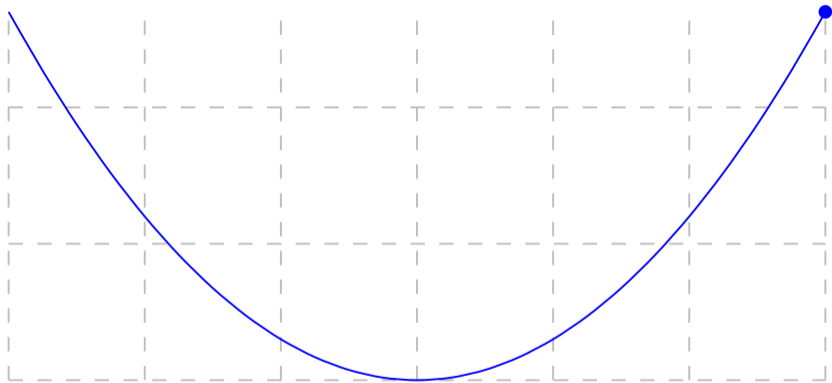
Example: Variation of MSE wrt w



Example: Best fit



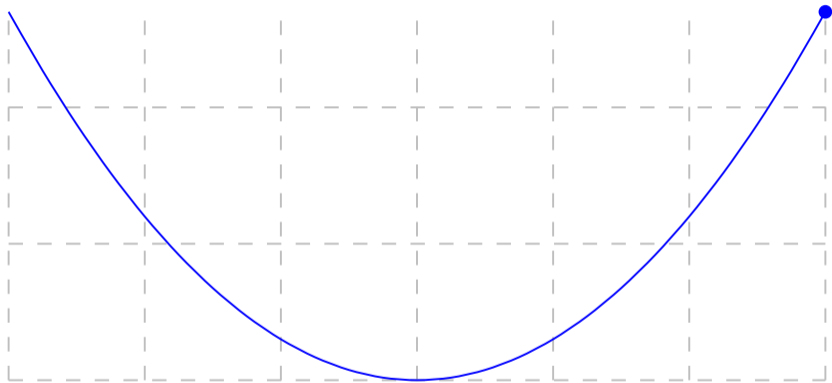
Gradient descent



$$y = 0.3x^2, x_0 = 3, \alpha = 0.8$$

$$\text{gradient} = 1.80002$$

Gradient descent

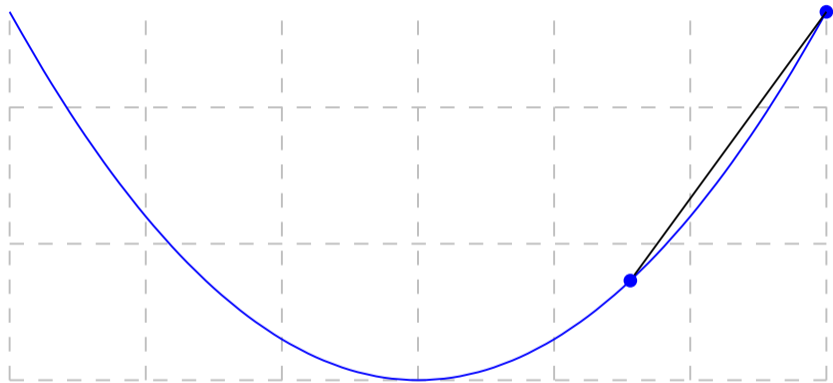


$$y = 0.3x^2, x_0 = 3, \alpha = 0.8$$

$$\text{gradient} = 1.80002$$

$$x_{\text{new}} = 1.56001$$

Gradient descent

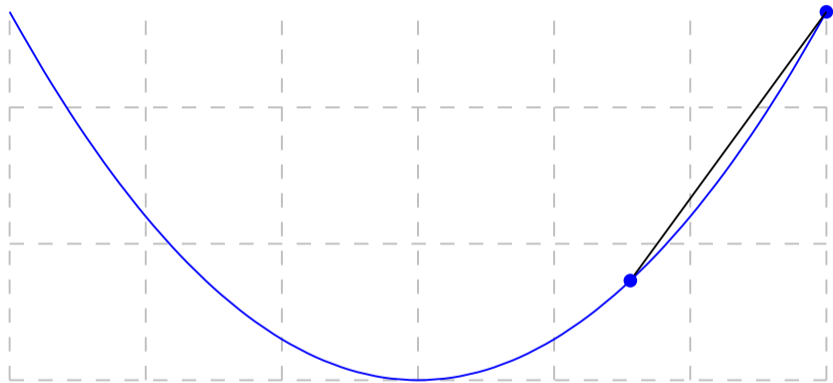


$$y = 0.3x^2, x_0 = 3, \alpha = 0.8$$

$$\text{gradient} = 1.80002$$

$$x_{\text{new}} = 1.56001$$

Gradient descent

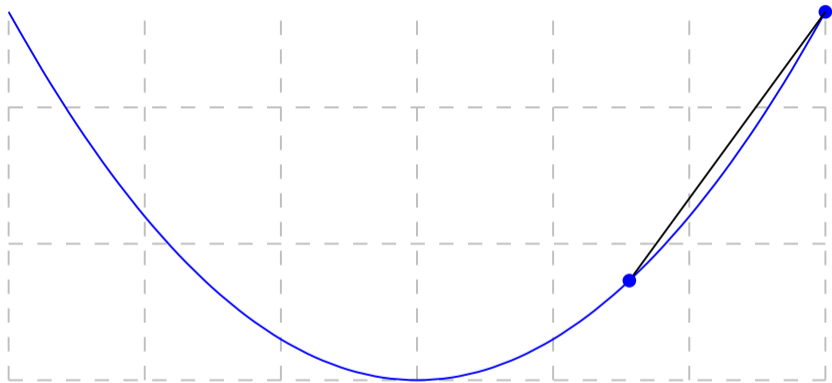


$$y = 0.3x^2, x_0 = 3, \alpha = 0.8$$

$$\text{gradient} = 0.936$$

$$x_{\text{new}} = 1.56001$$

Gradient descent

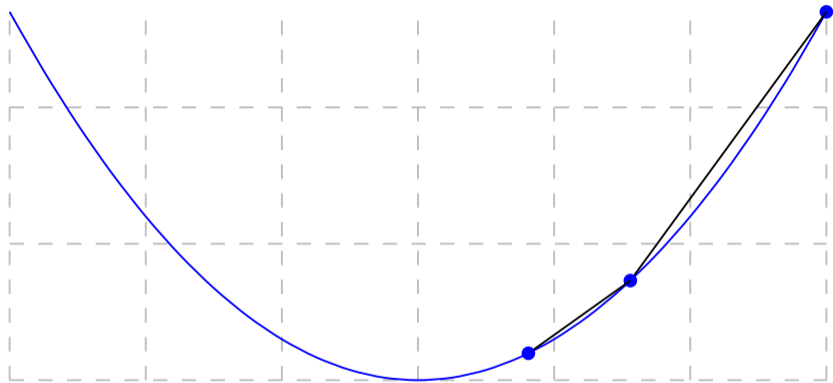


$$y = 0.3x^2, x_0 = 3, \alpha = 0.8$$

$$\text{gradient} = 0.936$$

$$x_{\text{new}} = 0.81122$$

Gradient descent

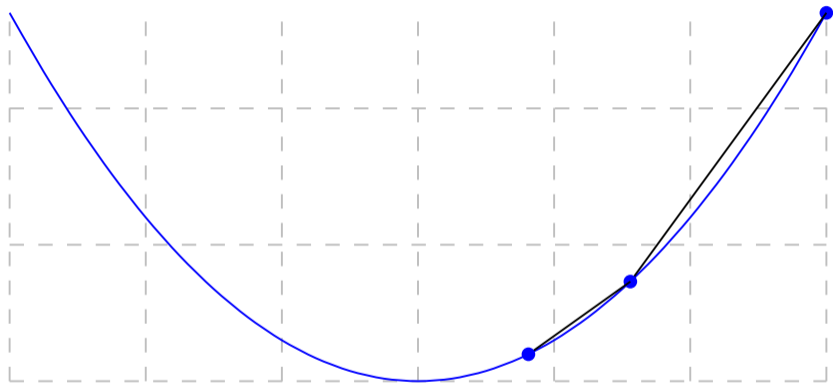


$$y = 0.3x^2, x_0 = 3, \alpha = 0.8$$

$$\text{gradient} = 0.936$$

$$x_{\text{new}} = 0.81122$$

Gradient descent

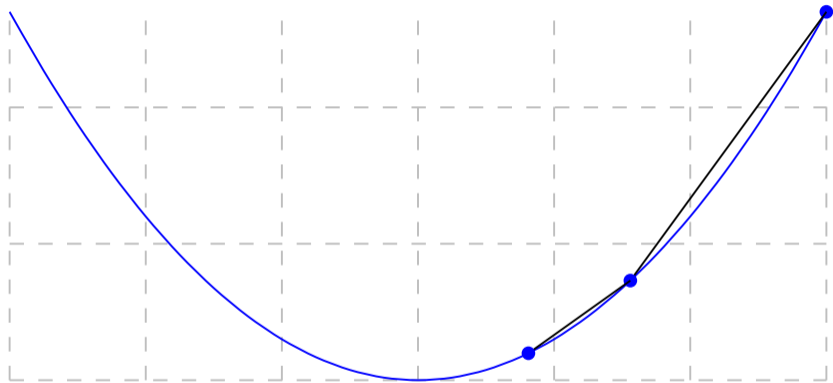


$$y = 0.3x^2, x_0 = 3, \alpha = 0.8$$

$$\text{gradient} = 0.48672$$

$$x_{\text{new}} = 0.81122$$

Gradient descent

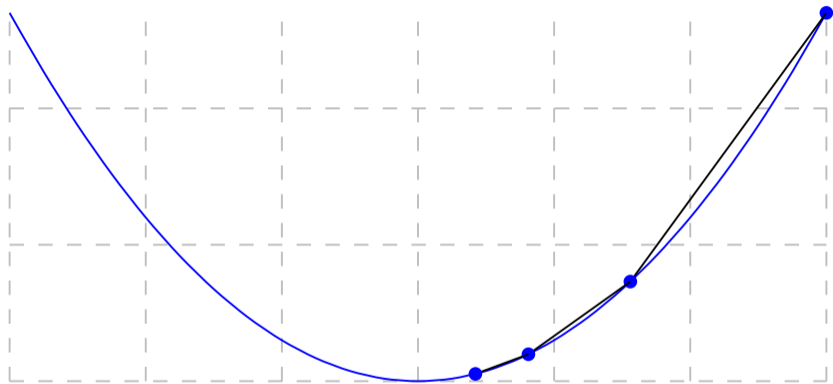


$$y = 0.3x^2, x_0 = 3, \alpha = 0.8$$

$$\text{gradient} = 0.48672$$

$$x_{\text{new}} = 0.42184$$

Gradient descent

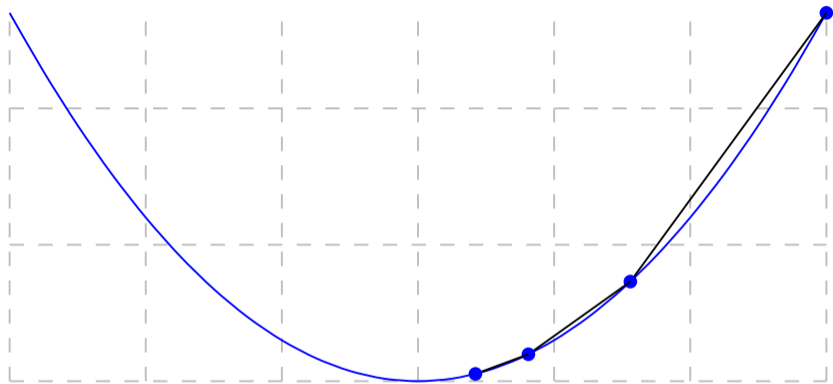


$$y = 0.3x^2, x_0 = 3, \alpha = 0.8$$

$$\text{gradient} = 0.48672$$

$$x_{\text{new}} = 0.42184$$

Gradient descent

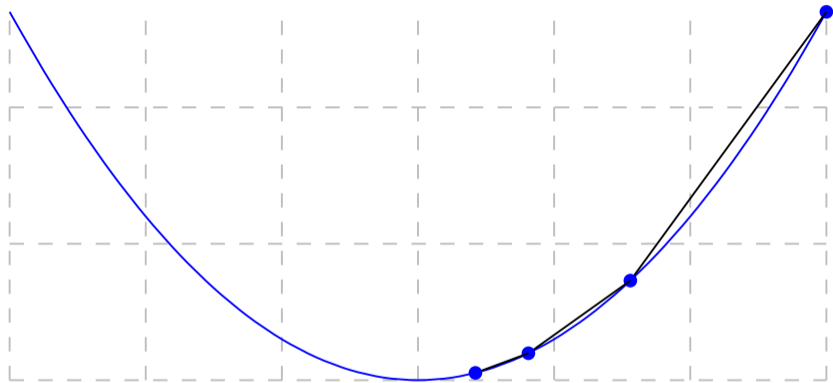


$$y = 0.3x^2, x_0 = 3, \alpha = 0.8$$

$$\text{gradient} = 0.2531$$

$$x_{\text{new}} = 0.42184$$

Gradient descent

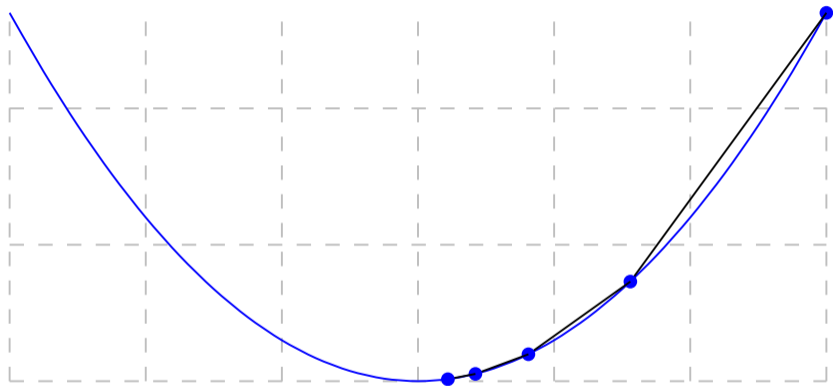


$$y = 0.3x^2, x_0 = 3, \alpha = 0.8$$

$$\text{gradient} = 0.2531$$

$$x_{\text{new}} = 0.21938$$

Gradient descent

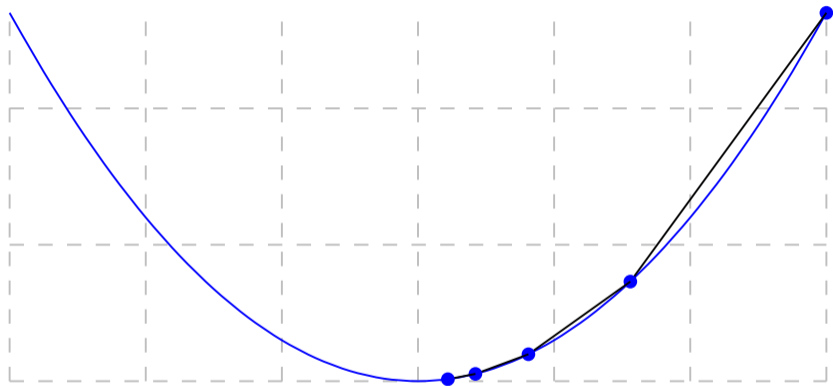


$$y = 0.3x^2, x_0 = 3, \alpha = 0.8$$

$$\text{gradient} = 0.2531$$

$$x_{\text{new}} = 0.21938$$

Gradient descent

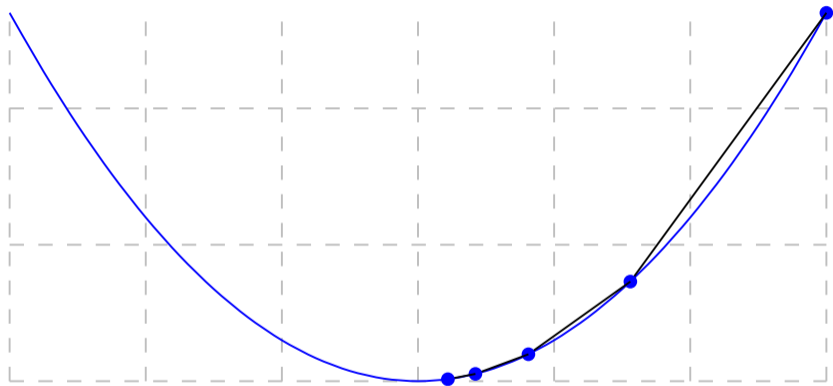


$$y = 0.3x^2, x_0 = 3, \alpha = 0.8$$

$$\text{gradient} = 0.13162$$

$$x_{\text{new}} = 0.21938$$

Gradient descent

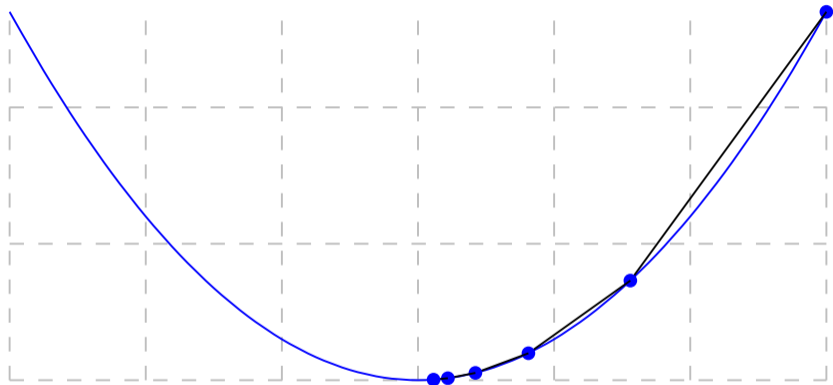


$$y = 0.3x^2, x_0 = 3, \alpha = 0.8$$

$$\text{gradient} = 0.13162$$

$$x_{\text{new}} = 0.11409$$

Gradient descent

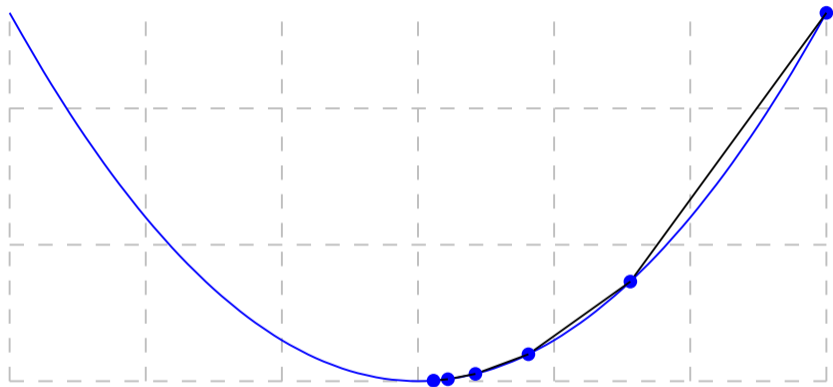


$$y = 0.3x^2, x_0 = 3, \alpha = 0.8$$

$$\text{gradient} = 0.13162$$

$$x_{\text{new}} = 0.11409$$

Gradient descent

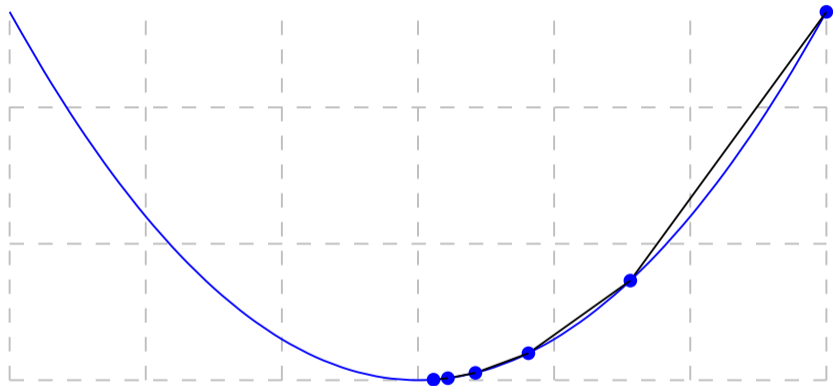


$$y = 0.3x^2, x_0 = 3, \alpha = 0.8$$

$$\text{gradient} = 0.06845$$

$$x_{\text{new}} = 0.11409$$

Gradient descent

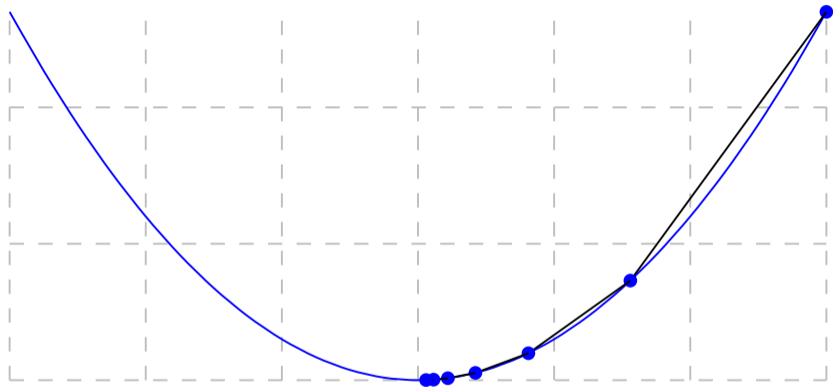


$$y = 0.3x^2, x_0 = 3, \alpha = 0.8$$

$$\text{gradient} = 0.06845$$

$$x_{\text{new}} = 0.05934$$

Gradient descent

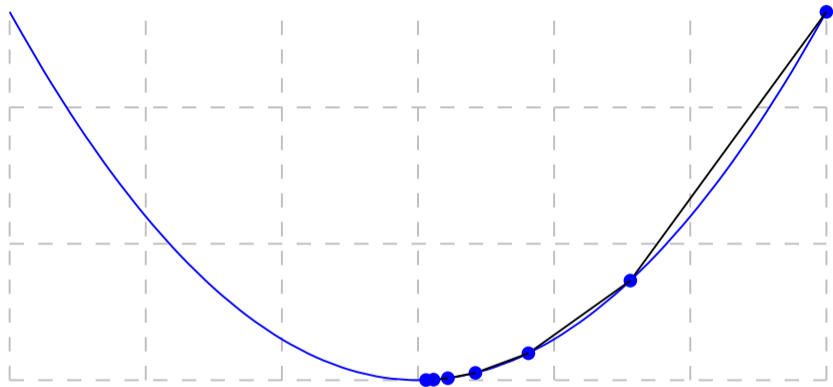


$$y = 0.3x^2, x_0 = 3, \alpha = 0.8$$

$$\text{gradient} = 0.06845$$

$$x_{\text{new}} = 0.05934$$

Gradient descent

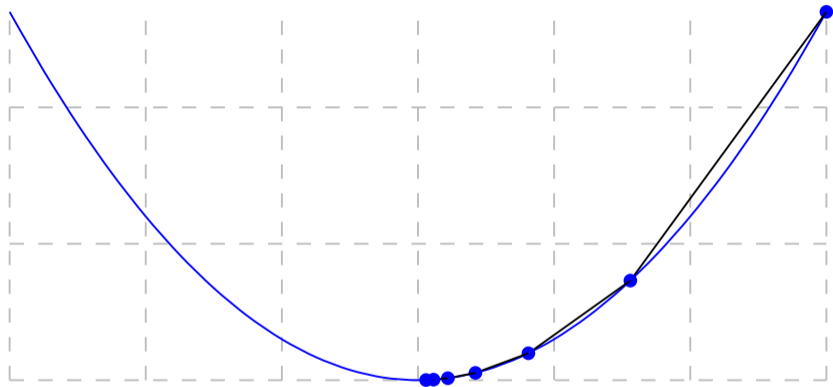


$$y = 0.3x^2, x_0 = 3, \alpha = 0.8$$

$$\text{gradient} = 0.0356$$

$$x_{new} = 0.05934$$

Gradient descent



$$y = 0.3x^2, x_0 = 3, \alpha = 0.8$$

$$\text{gradient} = 0.0356$$

$$x_{\text{new}} = 0.03087$$

Minimization of MSE: Gradient descent

- Assuming $\text{MSE}_{(\text{train})} = J(w_1, w_2)$
- Target is to $\min_{w_1, w_2} J(w_1, w_2)$
- Approach
 - Start with some w_1, w_2
 - Keep modifying w_1, w_2 so that $J(w_1, w_2)$ reduces till the desired accuracy is achieved

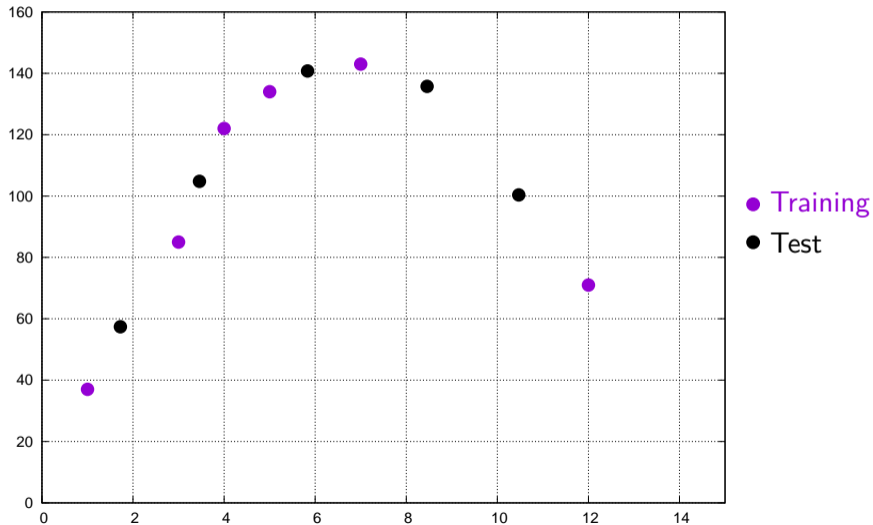
Minimization of MSE: Gradient descent

- Assuming $\text{MSE}_{(\text{train})} = J(w_1, w_2)$
- Target is to $\min_{w_1, w_2} J(w_1, w_2)$
- Approach
 - Start with some w_1, w_2
 - Keep modifying w_1, w_2 so that $J(w_1, w_2)$ reduces till the desired accuracy is achieved
- Algorithm
 - Repeat the following until convergence $w_j = w_j - \frac{\partial}{\partial w_j} J(w_1, w_2)$
- Gradient descent proposes a new point as $w' = w - \epsilon \nabla_w f(w)$ where ϵ is the learning rate

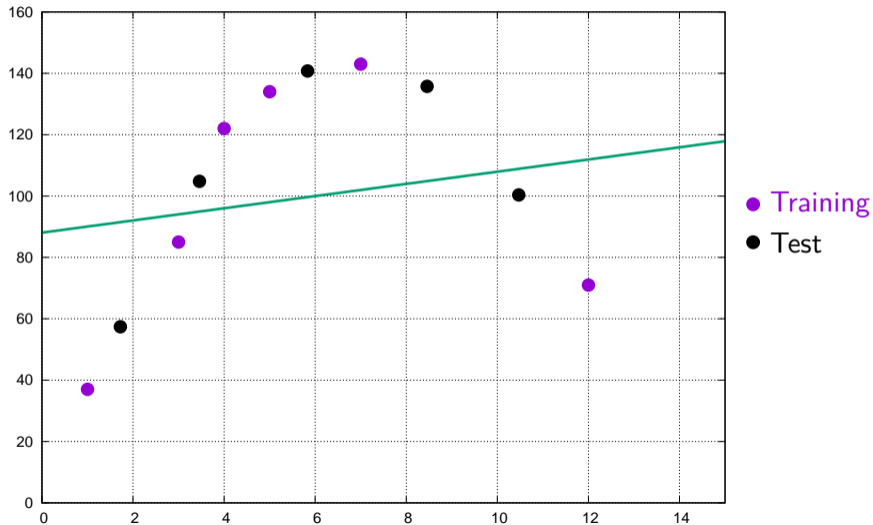
Error

- Training error - Error obtained on a training set
- Generalization error - Error on unseen data
- Data assumed to be independent and identically distributed (iid)
 - Each data set are independent of each other
 - Train and test data are identically distributed
- **Expected** training and test error will be the same
- It is more likely that the test error is greater than or equal to the expected value of training error
- Target is to make the training error is small. Also, to make the gap between training and test error smaller

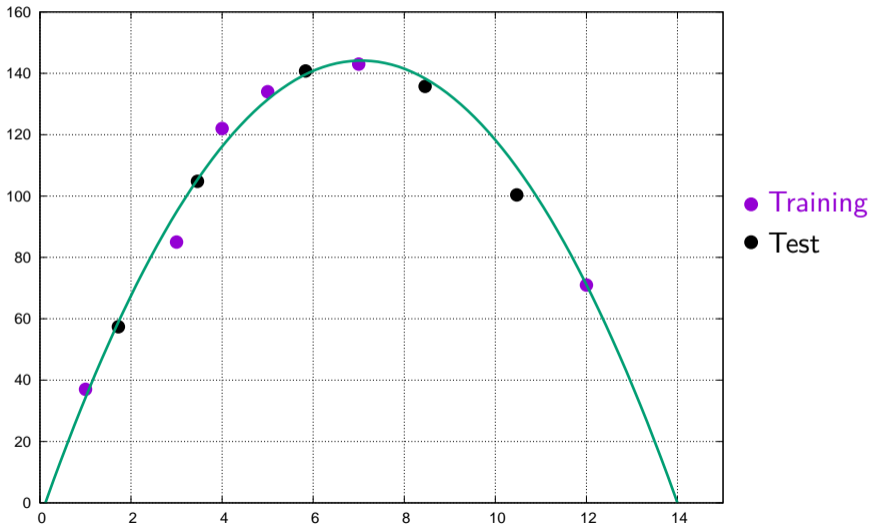
Regression example



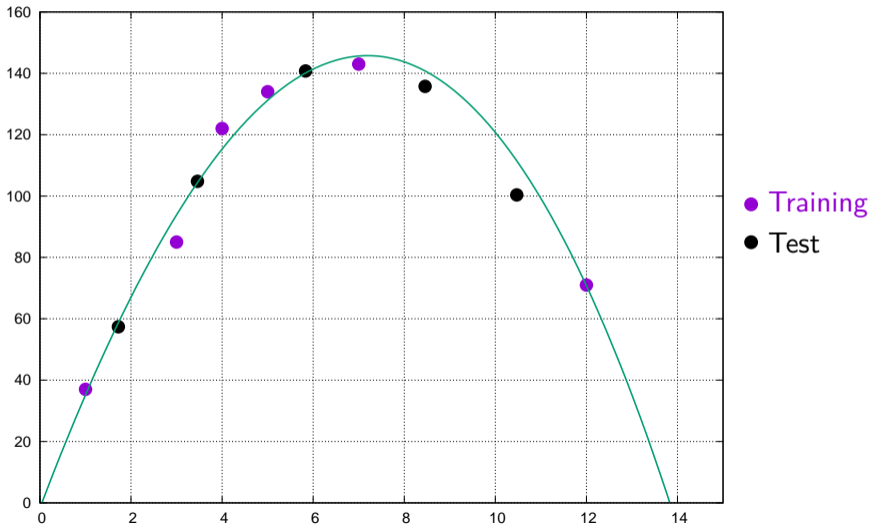
Regression example: degree 1



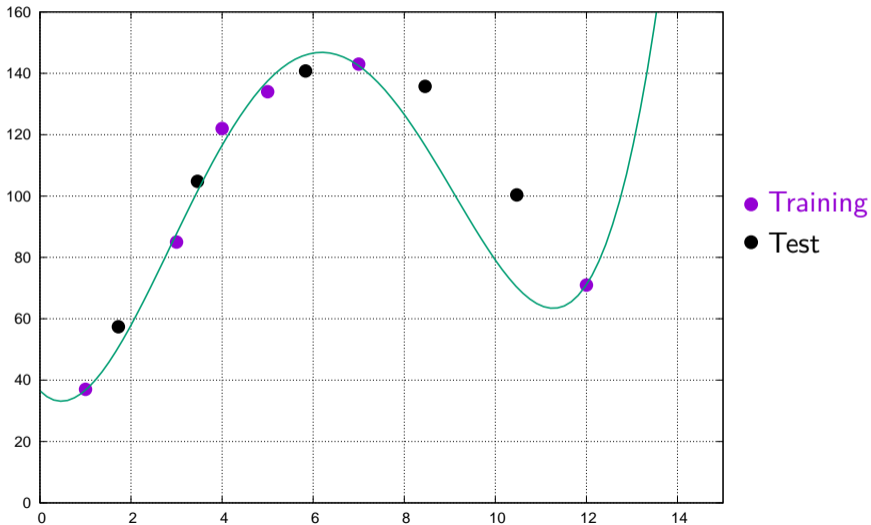
Regression example: degree 2



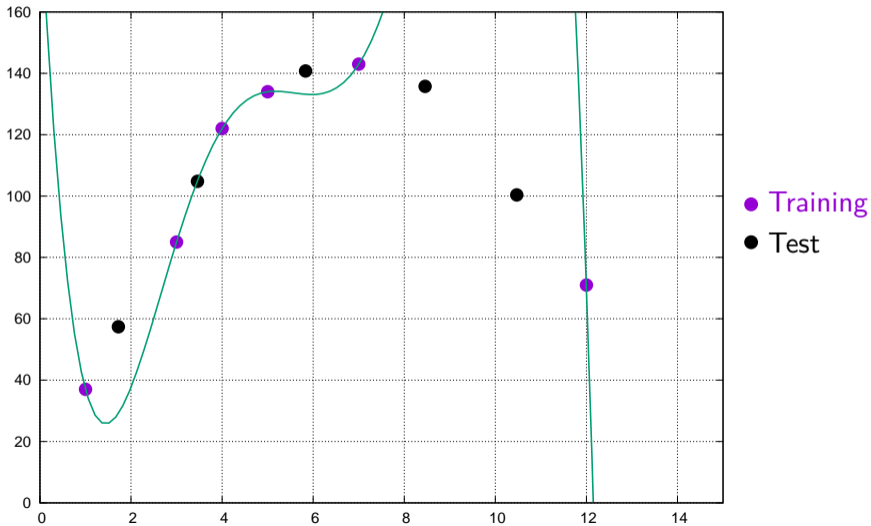
Regression example: degree 3



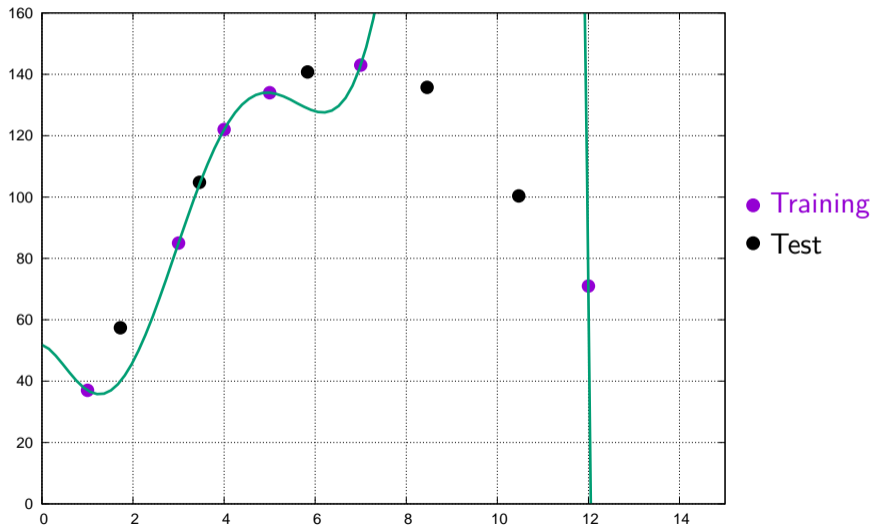
Regression example: degree 4



Regression example: degree 5



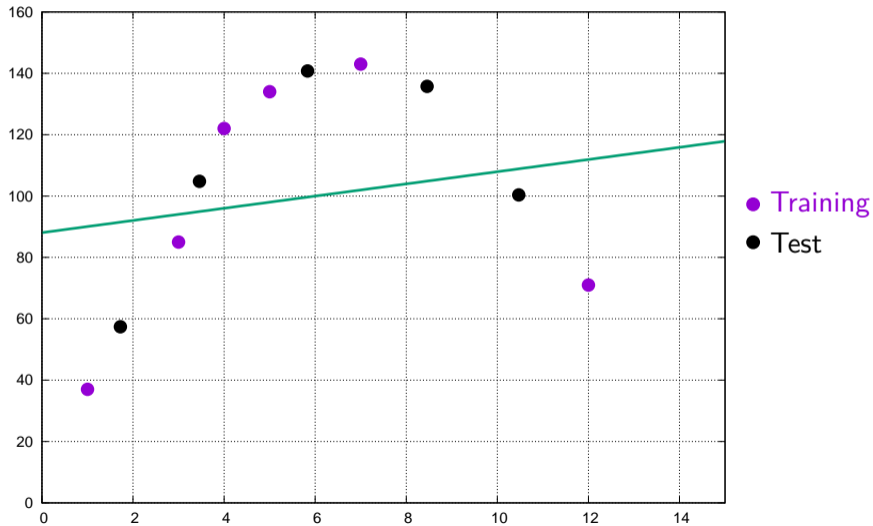
Regression example: degree 6



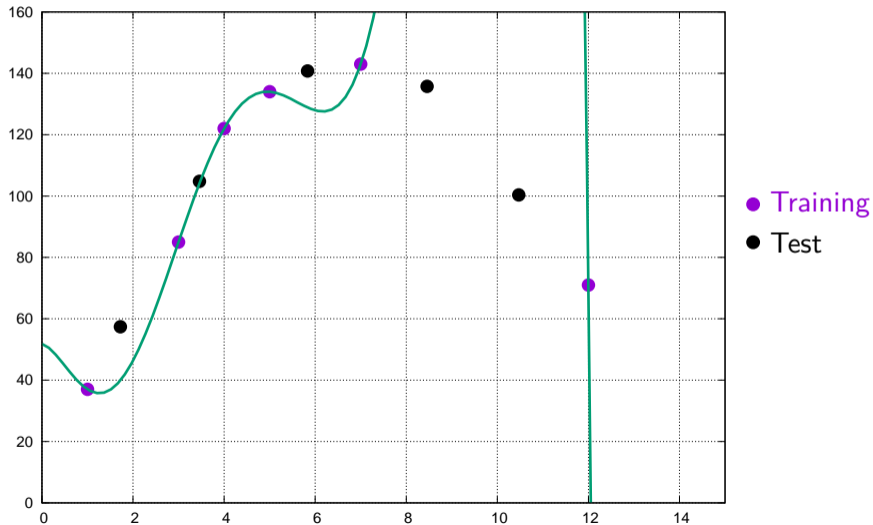
Underfitting & Overfitting

- Underfitting
 - When the model is not able to obtain sufficiently low error value on the training set
- Overfitting
 - When the gap between training set and test set error is too large

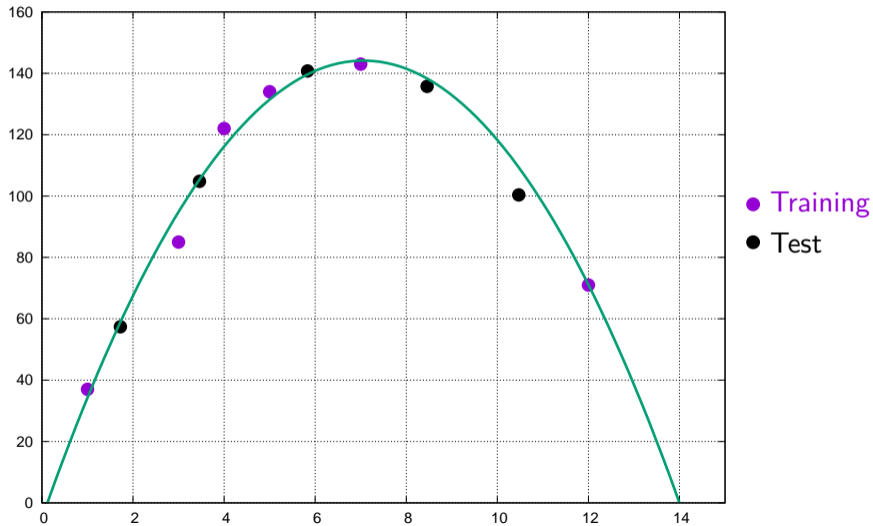
Underfitting example



Overfitting example



Better fit



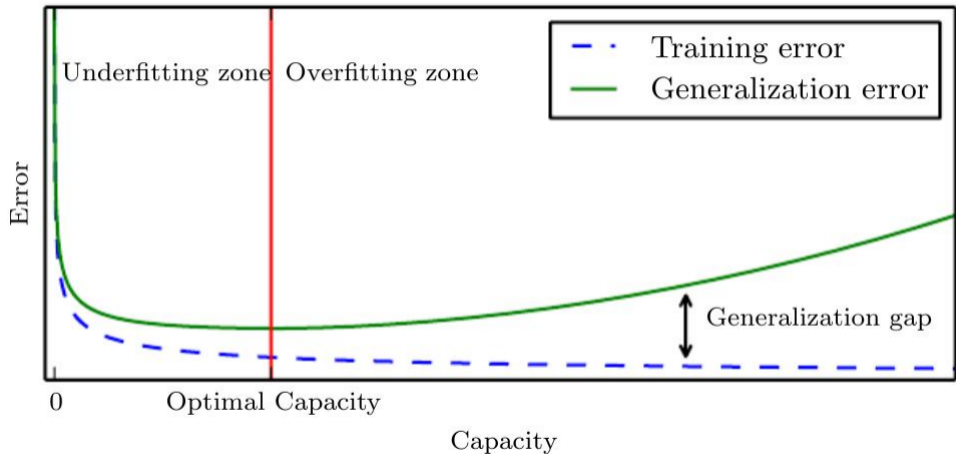
Capacity

- Ability to fit wide variety of functions
 - Low capacity will struggle to fit the training set
 - High capacity will can overfit by memorizing the training set
- Capacity can be controlled by choosing hypothesis space
 - A polynomial of degree 1 gives linear regression $\hat{y} = b + wx$
 - By adding x^2 term, it can learn quadratic curve $\hat{y} = b + w_1x + w_2x^2$
 - Output is still a linear function of parameters
- Capacity is determined by the choice of model (Representational capacity)
- Finding best function is very difficult optimization problem
 - Learning algorithm does not find the best function but reduces the training error
 - Imperfection in optimization algorithm can further reduce the capacity of model (effective capacity)

Capacity (contd.)

- Occam's razor
 - Among equally well hypotheses, choose the simplest one
- Vapnik-Chervonenski dimension - Capacity for binary classifier
 - Largest possible value of m for which a training set of m different x points that the classifier can label arbitrarily
- Training and test error is bounded from above by a quantity that grows as model capacity grows but shrinks as the number of training example increases
 - Bounds are usually provided for ML algorithm and rarely provided for DL
 - Capacity of deep learning model is difficult as the effective capacity is limited by optimization algorithm
 - Little knowledge on non-convex optimization

Error vs Capacity



Non-parametric model

- Parametric model learns a function described by a parameter vector
 - Size of vector is finite and fixed
- Nearest neighbor regression
 - Finds out the nearest entry in training set and returns the associated value as the predicted one
 - Mathematically, for a given point x , $\hat{y} = y_i$ where $i = \arg \min \|X_{i,:} - x\|_2^2$
- Wrapping parametric algorithm inside another algorithm

Bayes error

- Ideal model is an oracle that knows the true probability distribution for data generation
- Such model can make error because of noise
 - Supervised learning
 - Mapping of x to y may be stochastic
 - y may be deterministic but x does not have all variables
- Error by an oracle in predicting from the true distribution is known as Bayes error

Note

- Training and generalization error varies as the size of training set varies
- Expected generalization error can never increase as the number of training example increases
- Any fixed parametric model with less than the optimal capacity will asymptote to an error value that exceeds the Bayes error
- It is possible to have optimal capacity but have large gap between training and generalization error
 - Need more training examples

No free lunch

- Averaged over all possible data generating distribution, every classification algorithm has same error rate when classifying unseen points
- No machine learning algorithm is universally any better than any other

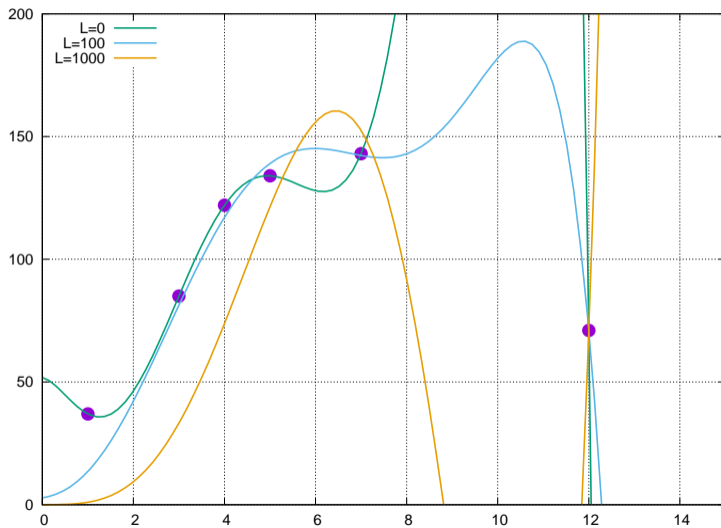
Regularization

- A set of preferences is applied to learning algorithm so that it performs well on a specific task
- Weight decay - In linear regression, preference on the weights is introduced
 - Sum of MSE and squared L^2 norms of the weight is minimized ie.

$$J(w) = \text{MSE}_{train} + \lambda w^T w$$

- $\lambda = 0$ - No preference
- λ becomes large - weight becomes smaller
- Regularization is intended to reduce test error not training error

Example: Weight decay



Hyperparameters

- Settings that are used to control the behavior of learning algorithm
 - Degree of polynomial
 - λ for decay weight
- Hyperparameters are usually not adapted or learned on the training set

Validation set

- Test data should not be used to choose the model as well as hyperparameters
- Validation set is constructed from training set
 - Typically 80% will be used for training and rest for validation
- Validation set may be used to train hyperparameters

Cross validation

- Dividing data set into training and fixed test may result into small test set
 - For large data this is not an issue
- For small data set use k-fold cross validation
 - Partition the data in k disjoint subsets
 - On i-th trial, i-th set used as the test set and rest are treated as training set
 - Test error can be determined by averaging the test error across the k trials

Point estimation

- To provide single best prediction of some quantity of interest
- Estimation of the relationship between input and output variables
- It can be single parameter or a vector of parameters
 - Weights in linear regression
- Notation: true parameter — θ and estimate — $\hat{\theta}$
- Let $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$ be set of m independent and identically distributed point.
- A point estimator is a function $\hat{\theta}_m = g(x^{(1)}, x^{(2)}, \dots, x^{(m)})$
 - Good estimator is a function whose output is close to θ
 - θ is unknown but fixed
 - $\hat{\theta}$ depends on data

Bias

- Difference between this estimator's expected value and the true value of the parameter being estimated
 - $\text{bias}(\hat{\theta}_m) = \mathbb{E}(\hat{\theta}_m) - \theta$
- An estimator will be said unbiased if $\text{bias}(\hat{\theta}_m) = 0$
 - $\mathbb{E}(\hat{\theta}_m) = \theta$
- An estimator will be asymptotically unbiased if $\lim_{m \rightarrow \infty} \text{bias}(\hat{\theta}_m) = 0$

Estimator for Gaussian distribution

- Let us consider a set of samples $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$ that are independently and identically distributed according to

$$p(x^{(i)}) = \mathcal{N}(x^{(i)}; \mu, \sigma^2) \quad \forall i = 1, 2, \dots, m$$

Estimator for Gaussian distribution

- Let us consider a set of samples $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$ that are independently and identically distributed according to

$$p(x^{(i)}) = \mathcal{N}(x^{(i)}; \mu, \sigma^2) \quad \forall i = 1, 2, \dots, m$$

- Gaussian mean estimator (aka sample mean) — $\hat{\mu}_m = \frac{1}{m} \sum_{i=1}^m x^{(i)}$

Estimator for Gaussian distribution

- Let us consider a set of samples $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$ that are independently and identically distributed according to

$$p(x^{(i)}) = \mathcal{N}(x^{(i)}; \mu, \sigma^2) \quad \forall i = 1, 2, \dots, m$$

- Gaussian mean estimator (aka sample mean) — $\hat{\mu}_m = \frac{1}{m} \sum_{i=1}^m x^{(i)}$
- Bias of sample mean

$$\text{bias}(\hat{\mu}_m) = \mathbb{E}(\hat{\mu}_m) - \mu$$

Estimator for Gaussian distribution

- Let us consider a set of samples $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$ that are independently and identically distributed according to

$$p(x^{(i)}) = \mathcal{N}(x^{(i)}; \mu, \sigma^2) \quad \forall i = 1, 2, \dots, m$$

- Gaussian mean estimator (aka sample mean) — $\hat{\mu}_m = \frac{1}{m} \sum_{i=1}^m x^{(i)}$

- Bias of sample mean

$$\text{bias}(\hat{\mu}_m) = \mathbb{E}(\hat{\mu}_m) - \mu = \mathbb{E}\left(\frac{1}{m} \sum_{i=1}^m x^{(i)}\right) - \mu$$

Estimator for Gaussian distribution

- Let us consider a set of samples $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$ that are independently and identically distributed according to

$$p(x^{(i)}) = \mathcal{N}(x^{(i)}; \mu, \sigma^2) \quad \forall i = 1, 2, \dots, m$$

- Gaussian mean estimator (aka sample mean) — $\hat{\mu}_m = \frac{1}{m} \sum_{i=1}^m x^{(i)}$

- Bias of sample mean

$$\begin{aligned} \text{bias}(\hat{\mu}_m) &= \mathbb{E}(\hat{\mu}_m) - \mu = \mathbb{E}\left(\frac{1}{m} \sum_{i=1}^m x^{(i)}\right) - \mu \\ &= \left(\frac{1}{m} \sum_{i=1}^m \mathbb{E}(x^{(i)})\right) - \mu \end{aligned}$$

Estimator for Gaussian distribution

- Let us consider a set of samples $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$ that are independently and identically distributed according to

$$p(x^{(i)}) = \mathcal{N}(x^{(i)}; \mu, \sigma^2) \quad \forall i = 1, 2, \dots, m$$

- Gaussian mean estimator (aka sample mean) — $\hat{\mu}_m = \frac{1}{m} \sum_{i=1}^m x^{(i)}$

- Bias of sample mean

$$\begin{aligned} \text{bias}(\hat{\mu}_m) &= \mathbb{E}(\hat{\mu}_m) - \mu = \mathbb{E}\left(\frac{1}{m} \sum_{i=1}^m x^{(i)}\right) - \mu \\ &= \left(\frac{1}{m} \sum_{i=1}^m \mathbb{E}(x^{(i)})\right) - \mu = \left(\frac{1}{m} \sum_{i=1}^m \mu\right) - \mu \end{aligned}$$

Estimator for Gaussian distribution

- Let us consider a set of samples $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$ that are independently and identically distributed according to

$$p(x^{(i)}) = \mathcal{N}(x^{(i)}; \mu, \sigma^2) \quad \forall i = 1, 2, \dots, m$$

- Gaussian mean estimator (aka sample mean) — $\hat{\mu}_m = \frac{1}{m} \sum_{i=1}^m x^{(i)}$
- Bias of sample mean

$$\begin{aligned} \text{bias}(\hat{\mu}_m) &= \mathbb{E}(\hat{\mu}_m) - \mu = \mathbb{E}\left(\frac{1}{m} \sum_{i=1}^m x^{(i)}\right) - \mu \\ &= \left(\frac{1}{m} \sum_{i=1}^m \mathbb{E}(x^{(i)})\right) - \mu = \left(\frac{1}{m} \sum_{i=1}^m \mu\right) - \mu = \mu - \mu = 0 \end{aligned}$$

Estimator for Gaussian distribution (cont)

- Sample variance

- $\hat{\sigma}_m^2 = \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \hat{\mu}_m)^2$

Estimator for Gaussian distribution (cont)

- Sample variance

- $\hat{\sigma}_m^2 = \frac{1}{m} \sum_{i=1}^m (x^{(i)} - \hat{\mu}_m)^2$

- Bias of sample variance $\text{bias}(\hat{\sigma}_m^2) = \mathbb{E}(\hat{\sigma}_m^2) - \sigma^2$

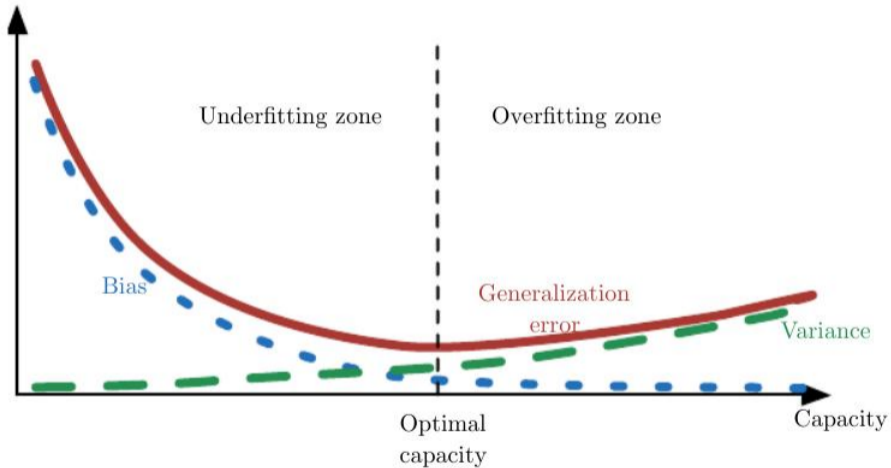
- It can be shown that, $\mathbb{E}(\hat{\sigma}_m^2) = \frac{m-1}{m} \sigma^2$

Trade off Bias and Variance

- Bias — Expected deviation from the true value of the function parameter
- Variance — Measure of deviation from the expected estimator value
- Choice of estimator — large bias or large variance?
 - Use cross-validation
 - Compare Mean Square Error

$$\text{MSE} = \mathbb{E}(\hat{\theta}_m - \theta)^2 = \text{bias}(\hat{\theta}_m)^2 + \text{Var}(\hat{\theta}_m)$$

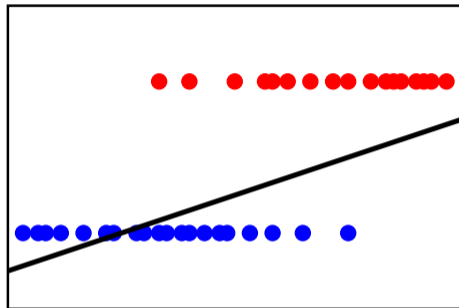
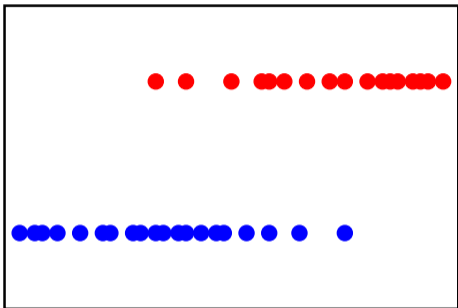
Trade off Bias and Variance (cont)



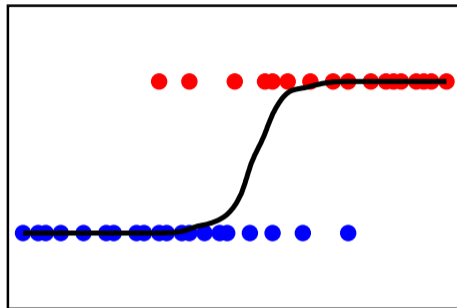
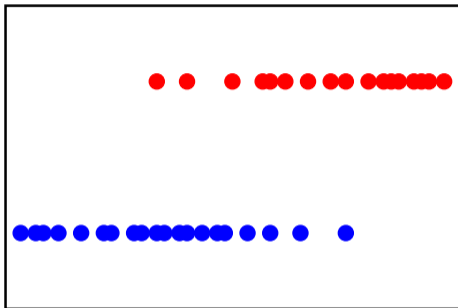
Logistic regression

- Responses may be qualitative (categorical)
 - Example: ⟨Hours of study, pass/fail⟩, ⟨MRI scan, benign/malignant⟩
 - Output should be 0 or 1
- Predicting qualitative response is known as classification
- Linear regression does not help

Issues with linear regression



Logistic regression



Logistic model

- Linear regression model to represent probability $p(x) = w_0 + w_1x$
- To avoid problem, we use function $p(x) = \frac{e^{w_0+w_1x}}{1 + e^{w_0+w_1x}}$
- Quantity $\frac{p(x)}{1-p(x)} = e^{w_0+w_1x}$ is known as odds
- Taking \log on both the sides, we get $\log\left(\frac{p(x)}{1-p(x)}\right) = w_0 + w_1x$
- Coefficient can be determined using maximum likelihood
 - $l(w_0, w_1) = \prod_{i:y_i=1} p(x_i) \prod_{j:y_j=0} p(x_j)$

Logistic model (contd.)

- Similar to linear regression except the output is mapped between 0 and 1 ie.

$$p(y|x, \theta) = \sigma(\theta^T x)$$

where $\sigma(x) = \frac{1}{1 + \exp(-x)}$ (Sigmoid function)

Support Vector Machine

- An approach for classification
- Developed in 1990s
- Generalization of maximum margin classifier
 - Mostly limited to linear boundary
- Support vector classifier — broad range of classes
- SVM — Non-linear class boundary

Hyperplane

- In n dimensional space a hyperplane is a flat affine subspace of dimension $n - 1$
- Mathematically it is defined as
 - For 2 dimensions — $w_0 + w_1x_1 + w_2x_2 = 0$
 - For n dimensions — $w_0 + w_1x_1 + \dots + w_nx_n = 0$

Classification using Hyperplane

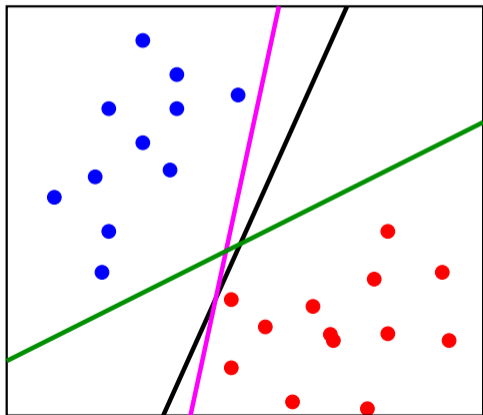
- Assume, m training observation in n dimensional space

Classification using Hyperplane

- Assume, m training observation in n dimensional space
- Separating hyperplane has the property
 - $w_0 + w_1x_1 + \dots + w_nx_n > 0$ if $y_i = 1$
 - $w_0 + w_1x_1 + \dots + w_nx_n < 0$ if $y_i = -1$

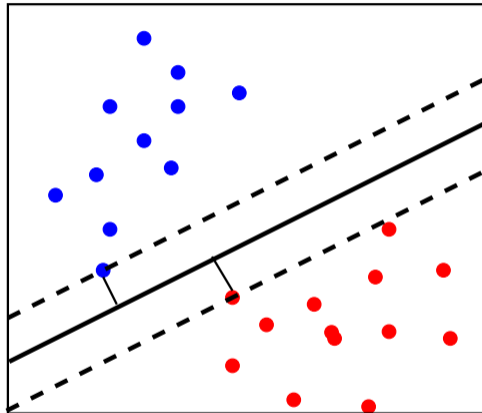
Classification using Hyperplane

- Assume, m training observation in n dimensional space
- Separating hyperplane has the property
 - $w_0 + w_1x_1 + \dots + w_nx_n > 0$ if $y_i = 1$
 - $w_0 + w_1x_1 + \dots + w_nx_n < 0$ if $y_i = -1$
- Hence, $y_i(w_0 + w_1x_1 + \dots + w_nx_n) > 0$
- Classification of test observation x^* is done based on the sign of
$$f(x^*) = w_0 + w_1x_1^* + \dots + w_nx_n^*$$
- Magnitude of $f(x^*)$
 - Far from 0 — Confident about prediction
 - Close to 0 — Less certain



Maximal margin classifier

- Also known as optimal separating hyperplane
- Separating hyperplane farthest from training observation
 - Compute perpendicular distance from training point to the hyperplane
 - Smallest of these distances represents the margin
- Target is to find the hyperplane for which the margin is the largest



Construction of maximal margin classifier

- Input — m points in n dimension space ie. x_1, x_2, \dots, x_m
- Input — labels y_1, y_2, \dots, y_m for each point x_i where $y_i \in \{-1, 1\}$
- Need to solve the following optimization problem

$$\max_{w_0, w_1, \dots, w_n, M} M$$

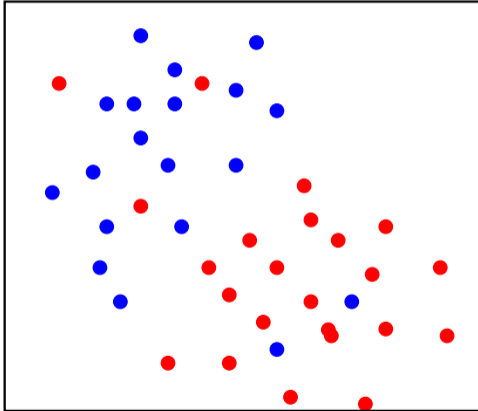
subject to

$$y_i(w_0 + w_1x_{i1} + w_2 + \dots + w_nx_{in}) \geq M \quad \forall i = 1, \dots, m$$

$$\sum_{i=1}^n w_i^2 = 1$$

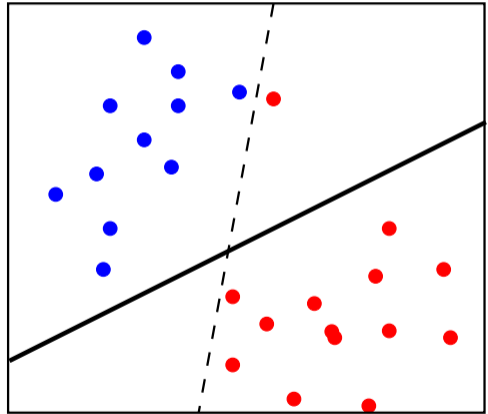
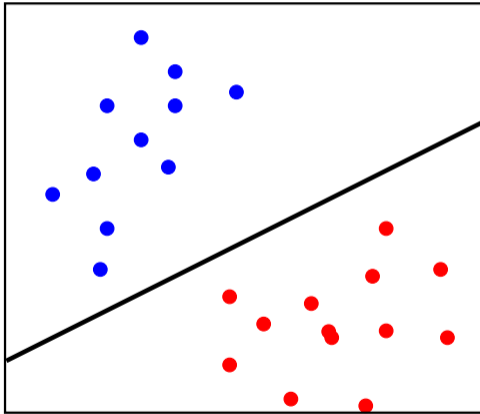
Issues

- Maximal margin classifier fails to provide classification in case of overlap



Issues

- Single observation point can change the hyperplane drastically

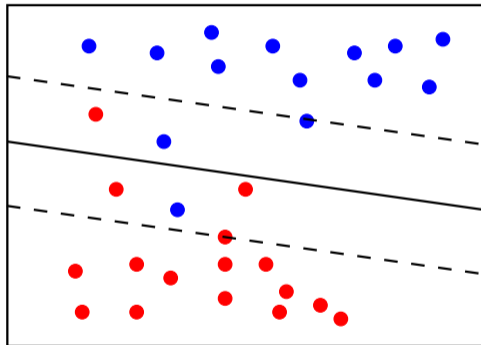
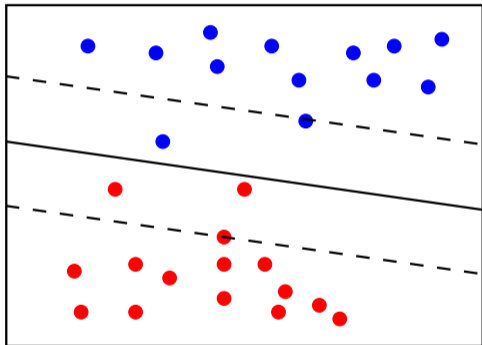


Support Vector Classifier

- Provides greater robustness to individual observations
- Better classification of most of the training observations
- Worthwhile to misclassify a few training observations
- Also known as soft margin classifier

Support Vector Classifier

- Points can lie within the margin or wrong side of hyperplane



Optimization with misclassification

- Input — x_1, x_2, \dots, x_m and y_1, y_2, \dots, y_m
- Need to solve the following optimization problem

$$\max_{w_0, w_1, \dots, w_n, M} M$$

subject to

$$y_i(w_0 + w_1x_{i1} + \dots + w_nx_{in}) \geq M(1 - \epsilon_i) \quad \forall i = 1, \dots, m$$

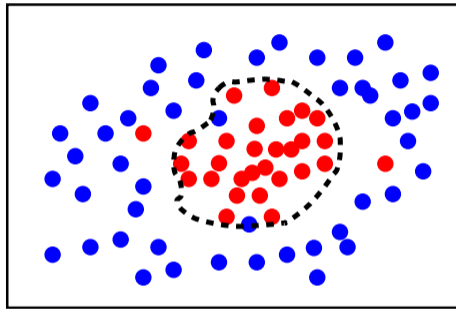
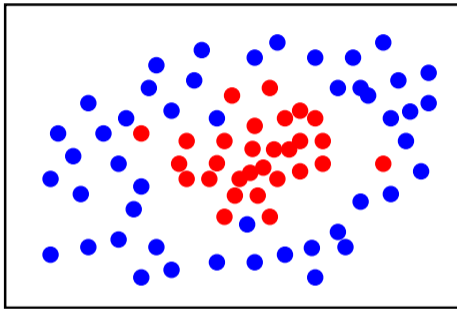
$$\sum_{i=1}^n w_i^2 = 1, \quad \sum_{i=1}^m \epsilon_i = C$$

- C is non-negative tuning parameter, ϵ_i - slack variable
- Classification of test observation remains the same

Observations

- $\epsilon_j = 0$ — j th observation is on the correct side of margin
- $\epsilon_j > 0$ — j th observation is on the wrong side of margin
- $\epsilon_j > 1$ — j th observation is on the wrong side of hyperplane
- C — budget for the amount that the margin can be violated by m observations
 - $C = 0$ — No violation, ie. maximal margin classifier
 - $C > 0$ — No more than C observation can be on the wrong side of hyperplane
 - C is small — Narrow margin, highly fit to data, low bias and high variance
 - C is large — Fitting data is less hard, more bias and may have less variance

Classification with non-linear boundaries



Classification with non-linear boundaries

- Performance of linear regression can suffer for non-linear data
- Feature space can be enlarged using function of predictors
 - For example, instead of fitting with x_1, x_2, \dots, x_n features we could use $x_1, x_1^2, x_2, x_2^2, \dots, x_n, x_n^2$ as features
- Optimization problem becomes

$$\max_{w_0, w_{11}, w_{12}, \dots, w_{n1}, w_{n2}, \epsilon_i, M} M$$

subject to

$$y_i \left(w_0 + \sum_{j=1}^n w_{j1} x_{ij} + \sum_{j=1}^n w_{j2} x_{ij}^2 \right) \geq M(1 - \epsilon_i) \quad \forall i = 1, \dots, m$$

$$\sum_{i=1}^n \sum_{j=1}^2 w_{ij}^2 = 1, \quad \sum_{i=1}^m \epsilon_i \leq C, \quad \epsilon_i \geq 0$$

Support Vector Machine

- Extension of support vector classifier that results from enlarging feature space
 - It involves inner product of the observations $f(x) = w_0 + \sum_{i=1}^m \alpha_i \langle x, x_i \rangle$ where α_i - one per training example
 - To estimate α_i and w_0 , we need $m(m-1)/2$ inner products, $\langle x_i, x_{i'} \rangle$
 - It turns out that $\alpha_i \neq 0$ for support vectors
- $f(x) = w_0 + \sum_{i \in S} \alpha_i \langle x, x_i \rangle$ where S - set of support vectors

Support Vector Machine

- Inner product is replaced with kernel, K or $K(x_i, x_{i'})$
- Kernel quantifies similarity between observations $K(x_i, x_{i'}) = \sum_{j=1}^n x_{ij}x_{i'j}$
 - Above one is Linear kernel ie. Pearson correlation
- Polynomial kernel $K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^n x_{ij}x_{i'j}\right)^d$ where d is positive integer > 1
- Support vector classifier with non-linear kernel is known as support vector machine and the function will look

$$f(x) = w_0 + \sum_{i \in S} \alpha_i K(x, x_i)$$

- Radial kernel: $K(x_i, x_{i'}) = \exp\left(-\gamma \sum_{j=1}^n (x_{ij} - x_{i'j})^2\right)$ where $\gamma > 0$

Challenges for Deep Learning

- Curse of dimensionality
- Local constancy and smoothness regularization
- Manifold learning