CS365: Deep Learning

Neural Networks-II

Arijit Mondal
Dept. of Computer Science & Engineering
Indian Institute of Technology Patna
arijit@iitp.ac.in

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
- A ML/DL algorithm is an algorithm that is able to learn from data
- Mitchelle (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/DL task is usually described in terms of how the 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 $\mathbf{x} \in \mathbb{R}^n$ where x_i is a feature
    - Feature of an image — pixel values

Typical tasks

- Classification
  - Need to predict which of the k categories some input belongs to
  - Need to have a function $f : \mathbb{R}^n \to \{1, 2, \ldots, k\}$
  - $y = f(\mathrm{x})$ 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 $f : \mathbb{R}^n \to \mathbb{R}$
  - Examples
    - Energy consumption
    - Amount of insurance claim

Typical tasks (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)

Typical tasks (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

Typical tasks (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)

Typical tasks (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)

Typical tasks (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

Typical tasks (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 DL 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

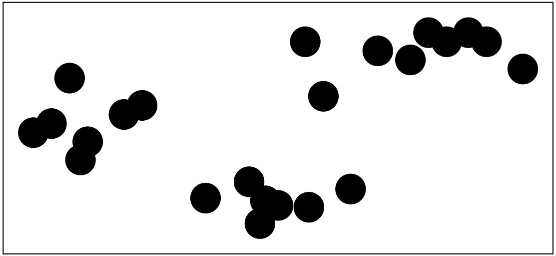
- Allowed to use labeled dataset
- Example — Iris
  - Collection of measurements of different parts of Iris plant
  - Each plant means each example
  - Features
    - Sepal length/width, petal length/width
    - Also record which species the plant belong to

Supervised learning (contd.)

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

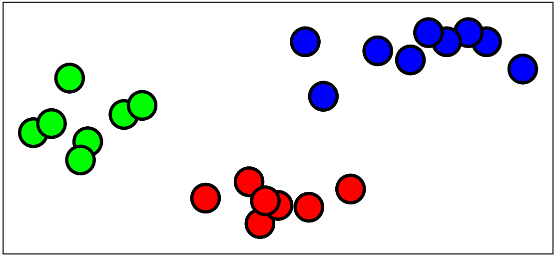
Unsupervised learning

- 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



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 $p(\mathrm{x})$
- Supervised tries to predict y from x ie. $p(y|\mathrm{x})$

Supervised vs Unsupervised learning

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

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

Supervised vs Unsupervised learning

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

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

- Solving supervised learning using traditional unsupervised learning

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

Multi layer neural network

- Pre-activation in layer

  $k > 0$ $(\mathrm{h}^{(0)}(\mathrm{x}) = x)$

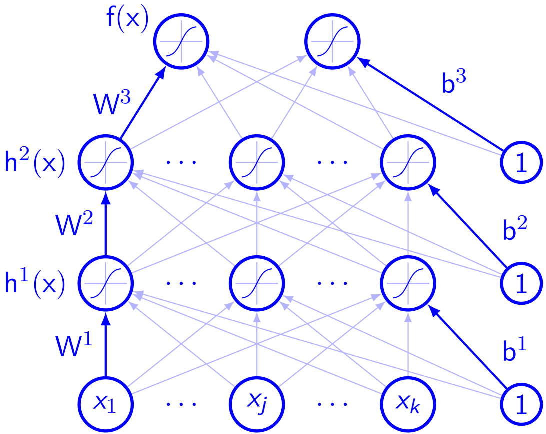
$$\mathrm{a}^{(k)}(\mathrm{x}) = \mathrm{b}^{(k)} + \mathrm{W}^{(k)}\mathrm{h}^{(k-1)}\mathrm{x}$$

- Hidden layer activation

$$\mathrm{h}^{(k)}(\mathrm{x}) = \mathrm{g}(\mathrm{a}^{(k)}(\mathrm{x}))$$

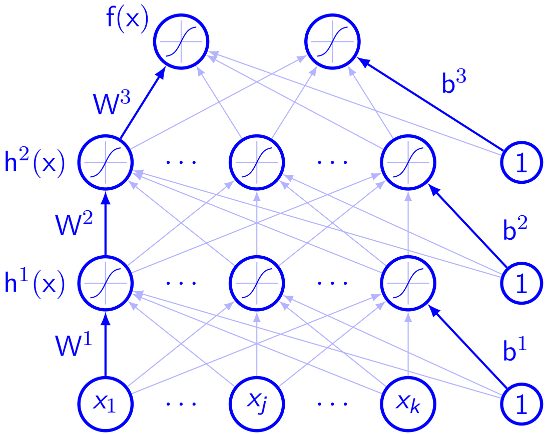
- Output layer activation

$$\mathrm{h}^{(L+1)}(\mathrm{x}) = o(\mathrm{a}^{(L+1)}(\mathrm{x})) = \mathrm{f}(\mathrm{x})$$



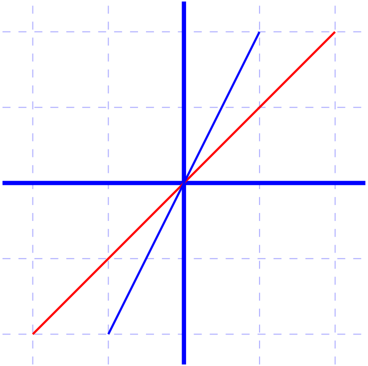
Multi layer neural network

- Design issues
  - Number of layers
  - Number of neurons in each layer
  - Activation function
  - Output function
  - Loss function
  - Optimizer



Activation function

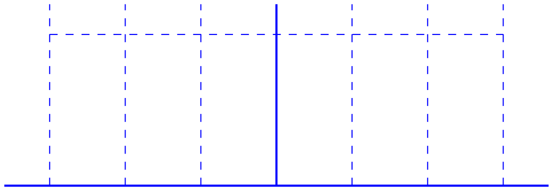
- Linear activation function
  - Not very interesting
  - No change in values
  - Huge range



$g(x) = x$, $g(x) = 2x$

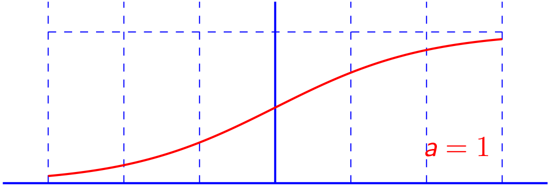
Activation function

- Sigmoid function
  - Values lie between 0 and 1
  - Strictly increasing function
  - Bounded



Activation function

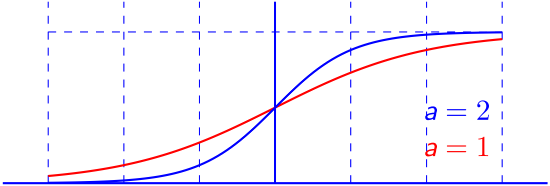
- Sigmoid function
  - Values lie between 0 and 1
  - Strictly increasing function
  - Bounded



$$\sigma(ax) = \frac{1}{1 + \exp(-a \times x)}$$

Activation function

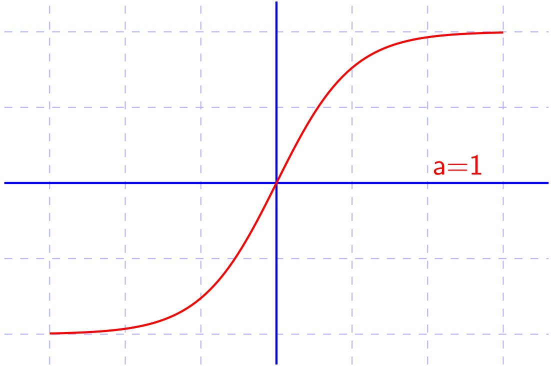
- Sigmoid function
  - Values lie between 0 and 1
  - Strictly increasing function
  - Bounded



$$\sigma(ax) = \frac{1}{1 + \exp(-a \times x)}$$

Activation function

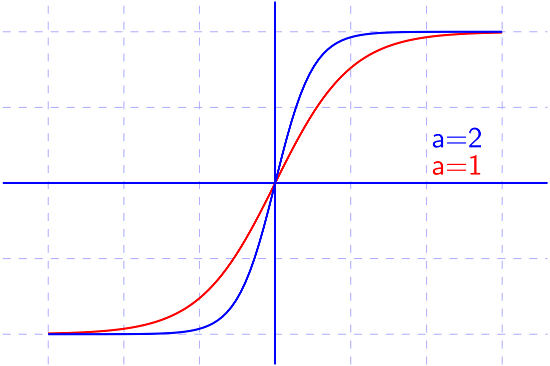
- Hyperbolic Tangent (Tanh) function
  - Can be positive or negative
  - Values lie between -1 and 1
  - Strictly increasing function
  - Bounded



$$g(ax) = \tanh(ax) = \frac{\exp(ax) - \exp(-ax)}{\exp(ax) + \exp(-ax)}$$

Activation function

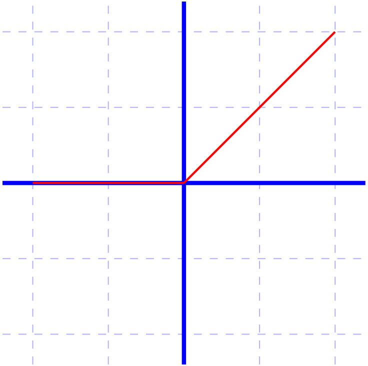
- Hyperbolic Tangent (Tanh) function
  - Can be positive or negative
  - Values lie between -1 and 1
  - Strictly increasing function
  - Bounded



$$g(ax) = \tanh(ax) = \frac{\exp(ax) - \exp(-ax)}{\exp(ax) + \exp(-ax)}$$

Activation function

- Rectified linear activation function (ReLU)
  - Bounded below by 0
  - Strictly increasing function
  - Not upper bounded



$g(x) = \mathrm{reclin}(x) = \max(0, x)$

Generalization of ReLU

- ReLU is defined as $g(z) = \max\{0, z\}$
- Using non-zero slope, $h_i = g(\mathrm{z}, \boldsymbol{\alpha})_i = \max(0, z_i) + \alpha_i \min(0, z_i)$
  - Absolute value rectification will make $\alpha_i = -1$ and $g(z) = |z|$
- Leaky ReLU assumes very small values for α_i
- Parametric ReLU tries to learn α_i parameters
- Maxout unit $g(\mathrm{z})_i = \max_{j \in \mathbb{G}^{(i)}} z_j$
  - Suitable for learning piecewise linear function

Logistic sigmoid & hyperbolic tangent

- Logistic sigmoid $g(z) = \sigma(z)$
- Hyperbolic tangent $g(z) = tanh(z)$
  - $tanh(z) = 2\sigma(2z) - 1$
- Widespread saturation of sigmoidal unit is an issue for gradient based learning
  - Usually discouraged to use as hidden units
- Usually, hyperbolic tangent function performs better where sigmoidal function must be used
  - Behaves linearly at 0
  - Sigmoidal activation function are more common in settings other than feedforward network

Other activation functions

- Differentiable functions are usually preferred
- Activation function $h = \cos(\mathsf{W}\mathsf{x} + \mathsf{b})$ performs well for MNIST data set
- Sometimes no activation function helps in reducing the number of parameters
- Radial Basis Function - $\phi(\mathsf{x}, \mathsf{c}) = \phi(\|\mathsf{x} - \mathsf{c}\|)$
  - Gaussian - $\exp(-(\varepsilon r)^2)$
- Softplus - $g(x) = \zeta(x) = \log(1 + exp(x))$
- Hard tanh - $g(x) = \max(-1, \min(1, x))$
- Hidden unit design is an active area of research

Hidden units

- Active area of research and does not have good guiding theoretical principle
- Usually rectified linear unit (ReLU) is chosen in most of the cases
- Design process consists of trial and error, then the suitable one is chosen
- Some of the activation functions are not differentiable (eg. ReLU)
  - Still gradient descent performs well
  - Neural network does not converge to local minima but reduces the value of cost function to a very small value

Output units

- Choice of output function usually depends on the type of problem being solved
- Usually linear function is chosen for regression and sigmoid for classification problems
- Any kind of output unit can be used as hidden unit

Linear units

- Suited for Gaussian output distribution
- Given features h, linear output unit produces $\hat{\mathrm{y}} = \mathrm{W}^T\mathrm{h} + \mathrm{b}$
- This can be treated as conditional probability $p(\mathrm{y}|\mathrm{x}) = \mathcal{N}(\mathrm{y}; \hat{\mathrm{y}}, \mathrm{I})$
- Maximizing log-likelihood is equivalent to minimizing mean square error

Sigmoid unit

- Mostly suited for binary classification problem that is Bernoulli output distribution
- The neural networks need to predict $p(y = 1|\mathrm{x})$
  - If linear unit has been chosen, $p(y = 1|\mathrm{x}) = \max\left\{0, \min\{1, \mathrm{W}^T\mathrm{h} + \mathrm{b}\}\right\}$
  - Gradient?
- Model should have strong gradient whenever the answer is wrong
- Let us assume unnormalized log probability is linear with $z = \mathrm{W}^T\mathrm{h} + \mathrm{b}$
- Therefore, $\log \tilde{P}(y) = yz \Rightarrow \tilde{P}(y) = \exp(yz) \Rightarrow P(y) = \frac{\exp(yz)}{\sum_{y' \in \{0,1\}} \exp(y'z)}$
  - It can be written as $P(y) = \sigma((2y-1)z)$
- The loss function for maximum likelihood is $J(\boldsymbol{\theta}) = -\log P(y|\mathrm{x}) = -\log \sigma((2y-1)z) = \zeta((1-2y)z)$

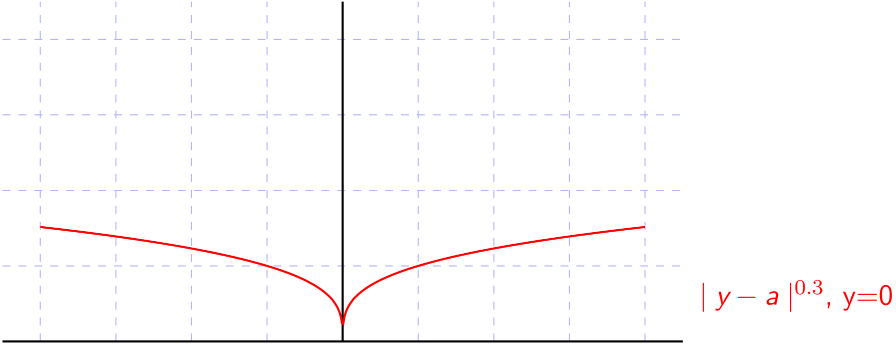
Softmax unit

- Similar to sigmoid. Mostly suited for multinoulli distribution
- We need to predict a vector $\hat{\mathrm{y}}$ such that $\hat{y}_i = P(Y = i|\mathrm{x})$
- A linear layer predicts unnormalized probabilities $\mathrm{z} = \mathrm{W}^T\mathrm{h} + \mathrm{b}$ that is $z_i = \log \tilde{P}(y = i|\mathrm{x})$
- Formally, $\mathrm{softmax}(\mathrm{z})_i = \dfrac{\exp z_i}{\sum_j \exp(z_j)}$
- Log in log-likelihood can undo exp $\log\ \mathrm{softmax}(\mathrm{z})_i = z_i - \log \sum_j \exp(z_j)$
  - Does it saturate?
  - What about incorrect prediction?
- Invariant to addition of some scalar to all input variables ie.
  $\mathrm{softmax}(\mathrm{z}) = \mathrm{softmax}(\mathrm{z} + \mathrm{c})$

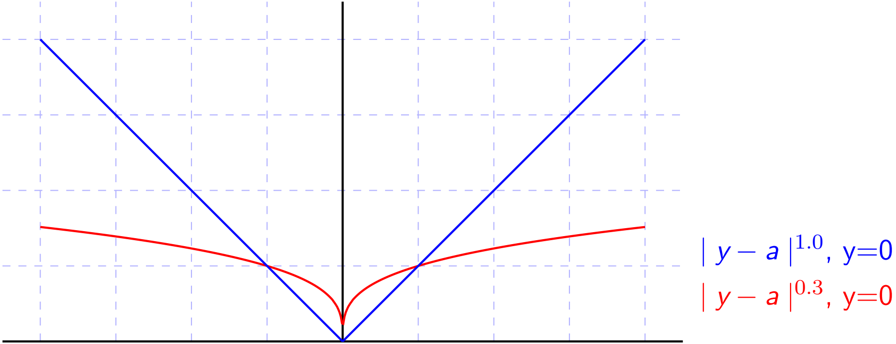
Loss function

- Need to compare $\hat{y} = f(x)$ with the true label y for an input x
- For a single input example loss will be measured as $\mathcal{L}(y, f(x))$
- Average loss over a set of examples will be $\frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(y_i, \hat{y}_i)$
- Target is to minimize the loss function
- Given the weights of the network W, the forward propagation yields $\hat{y}_i = f(x, W)$
- Our goal is as follows: $\underset{W}{\text{minimize}}\ \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(y_i, f(x_i, W))$
- Generic loss function can have the following form $\mid y - a \mid^p$
- Euclidean norm $p = 2$

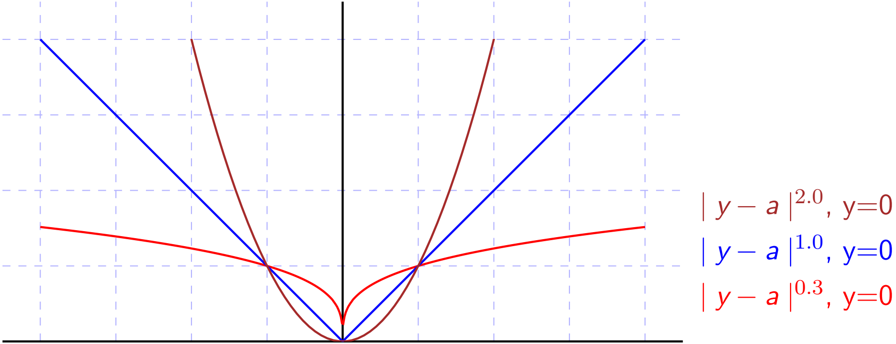
Loss curve



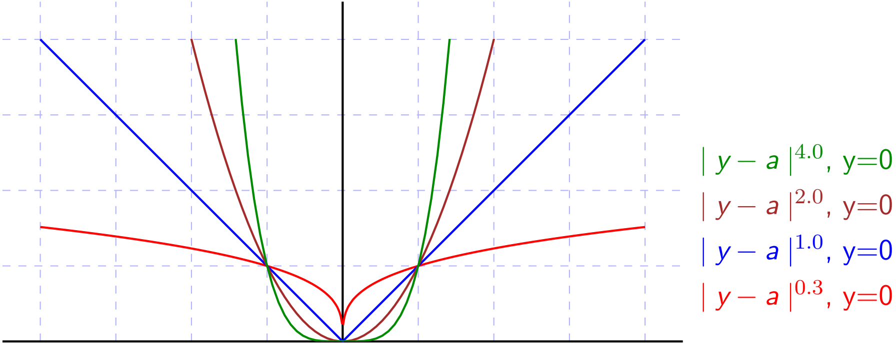
Loss curve



Loss curve



Loss curve



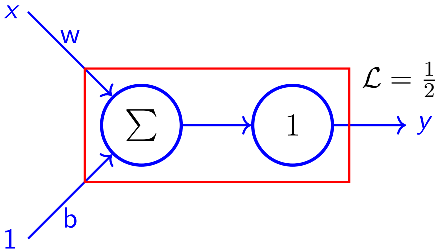
Linear regression

- 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 $\mathrm{x} \in \mathbb{R}^n$ and predict scalar $y \in \mathbb{R}$
  - Predicted value will be represented as $\hat{y} = \mathrm{w}^T\mathrm{x}$ where 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

Linear regression using neural network



$$\mathcal{L} = \frac{1}{2}(y - wx - b)^2$$

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 $\mathrm{X}^{(\text{test})}$ and target output is a vector $\mathrm{y}^{(\text{test})}$
  - Performance is measured by Mean Square Error (MSE)

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

  - 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 $\mathrm{X}^{(\text{test})}$ and target output is a vector $\mathrm{y}^{(\text{test})}$
  - Performance is measured by Mean Square Error (MSE)

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

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

Minimization of MSE

- We have the following now

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

Minimization of MSE

- We have the following now

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

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

Minimization of MSE

- We have the following now

$$
\begin{aligned}
& \nabla_w \mathrm{MSE}_{(\mathrm{train})} = 0 \\
\Rightarrow \quad & \nabla_w \frac{1}{m} \|\hat{\mathrm{y}}^{(\mathrm{train})} - \mathrm{y}^{(\mathrm{train})}\|_2^2 = 0 \\
\Rightarrow \quad & \frac{1}{m} \nabla_w \|\mathrm{X}^{(\mathrm{train})}\mathrm{w} - \mathrm{y}^{(\mathrm{train})}\|_2^2 = 0
\end{aligned}
$$

Minimization of MSE

- We have the following now

$$
\begin{aligned}
& \nabla_w \mathrm{MSE}_{(\mathrm{train})} = 0 \\
\Rightarrow \quad & \nabla_w \frac{1}{m} \|\hat{\mathrm{y}}^{(\mathrm{train})} - \mathrm{y}^{(\mathrm{train})}\|_2^2 = 0 \\
\Rightarrow \quad & \frac{1}{m} \nabla_w \|\mathrm{X}^{(\mathrm{train})}\mathrm{w} - \mathrm{y}^{(\mathrm{train})}\|_2^2 = 0 \\
\Rightarrow \quad & \nabla_w (\mathrm{X}^{(\mathrm{train})}\mathrm{w} - \mathrm{y}^{(\mathrm{train})})^T (\mathrm{X}^{(\mathrm{train})}\mathrm{w} - \mathrm{y}^{(\mathrm{train})}) = 0
\end{aligned}
$$

Minimization of MSE

- We have the following now

$$
\begin{aligned}
& \nabla_w \mathrm{MSE}_{(\mathrm{train})} = 0 \\
\Rightarrow \quad & \nabla_w \frac{1}{m} \|\hat{\mathrm{y}}^{(\mathrm{train})} - \mathrm{y}^{(\mathrm{train})}\|_2^2 = 0 \\
\Rightarrow \quad & \frac{1}{m} \nabla_w \|\mathrm{X}^{(\mathrm{train})}\mathrm{w} - \mathrm{y}^{(\mathrm{train})}\|_2^2 = 0 \\
\Rightarrow \quad & \nabla_w (\mathrm{X}^{(\mathrm{train})}\mathrm{w} - \mathrm{y}^{(\mathrm{train})})^T (\mathrm{X}^{(\mathrm{train})}\mathrm{w} - \mathrm{y}^{(\mathrm{train})}) = 0 \\
\Rightarrow \quad & \nabla_w (\mathrm{w}^T \mathrm{X}^{(\mathrm{train})T} \mathrm{X}^{(\mathrm{train})}\mathrm{w} - 2\mathrm{w}^T \mathrm{X}^{(\mathrm{train})T} \mathrm{y}^{(\mathrm{train})} + \mathrm{y}^{(\mathrm{train})T} \mathrm{y}^{(\mathrm{train})}) = 0
\end{aligned}
$$

Minimization of MSE

- We have the following now

$$
\begin{aligned}
& \nabla_w \text{MSE}_{(\text{train})} = 0 \\
\Rightarrow \quad & \nabla_w \frac{1}{m} \|\hat{\text{y}}^{(\text{train})} - \text{y}^{(\text{train})}\|_2^2 = 0 \\
\Rightarrow \quad & \frac{1}{m} \nabla_w \|\text{X}^{(\text{train})}\text{w} - \text{y}^{(\text{train})}\|_2^2 = 0 \\
\Rightarrow \quad & \nabla_w (\text{X}^{(\text{train})}\text{w} - \text{y}^{(\text{train})})^T (\text{X}^{(\text{train})}\text{w} - \text{y}^{(\text{train})}) = 0 \\
\Rightarrow \quad & \nabla_w (\text{w}^T \text{X}^{(\text{train})T} \text{X}^{(\text{train})}\text{w} - 2\text{w}^T \text{X}^{(\text{train})T} \text{y}^{(\text{train})} + \text{y}^{(\text{train})T} \text{y}^{(\text{train})}) = 0 \\
\Rightarrow \quad & 2\text{X}^{(\text{train})T} \text{X}^{(\text{train})}\text{w} - 2\text{X}^{(\text{train})T} \text{y}^{(\text{train})} = 0
\end{aligned}
$$

Minimization of MSE

- We have the following now

$$
\begin{aligned}
& \nabla_w \mathrm{MSE}_{(\mathrm{train})} = 0 \\
\Rightarrow \quad & \nabla_w \frac{1}{m} \|\hat{\mathrm{y}}^{(\mathrm{train})} - \mathrm{y}^{(\mathrm{train})}\|_2^2 = 0 \\
\Rightarrow \quad & \frac{1}{m} \nabla_w \|\mathrm{X}^{(\mathrm{train})}\mathrm{w} - \mathrm{y}^{(\mathrm{train})}\|_2^2 = 0 \\
\Rightarrow \quad & \nabla_w (\mathrm{X}^{(\mathrm{train})}\mathrm{w} - \mathrm{y}^{(\mathrm{train})})^T (\mathrm{X}^{(\mathrm{train})}\mathrm{w} - \mathrm{y}^{(\mathrm{train})}) = 0 \\
\Rightarrow \quad & \nabla_w (\mathrm{w}^T \mathrm{X}^{(\mathrm{train})T} \mathrm{X}^{(\mathrm{train})}\mathrm{w} - 2\mathrm{w}^T \mathrm{X}^{(\mathrm{train})T} \mathrm{y}^{(\mathrm{train})} + \mathrm{y}^{(\mathrm{train})T} \mathrm{y}^{(\mathrm{train})}) = 0 \\
\Rightarrow \quad & 2\mathrm{X}^{(\mathrm{train})T} \mathrm{X}^{(\mathrm{train})}\mathrm{w} - 2\mathrm{X}^{(\mathrm{train})T} \mathrm{y}^{(\mathrm{train})} = 0 \\
\Rightarrow \quad & \mathrm{w} = (\mathrm{X}^{(\mathrm{train})T} \mathrm{X}^{(\mathrm{train})})^{-1} \mathrm{X}^{(\mathrm{train})T} \mathrm{y}^{(\mathrm{train})}
\end{aligned}
$$

Minimization of MSE

- We have the following now

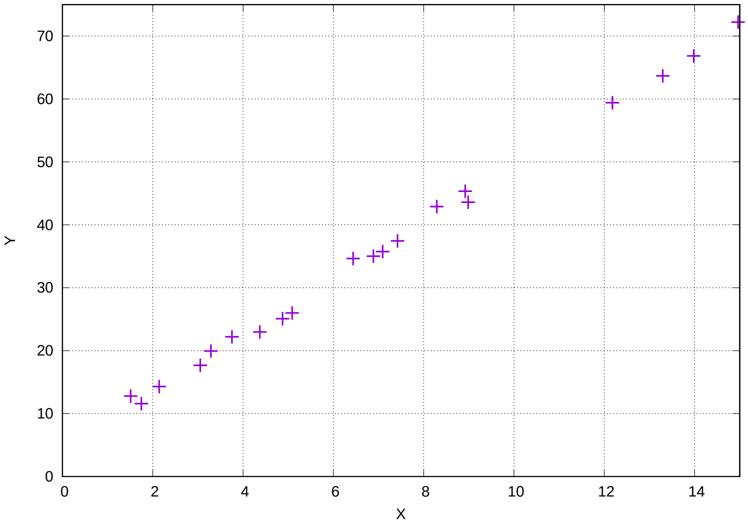
$$
\begin{aligned}
& \nabla_w \mathrm{MSE}_{(\mathrm{train})} = 0 \\
\Rightarrow\quad & \nabla_w \frac{1}{m} \|\hat{\mathrm{y}}^{(\mathrm{train})} - \mathrm{y}^{(\mathrm{train})}\|_2^2 = 0 \\
\Rightarrow\quad & \frac{1}{m} \nabla_w \|\mathrm{X}^{(\mathrm{train})}\mathrm{w} - \mathrm{y}^{(\mathrm{train})}\|_2^2 = 0 \\
\Rightarrow\quad & \nabla_w (\mathrm{X}^{(\mathrm{train})}\mathrm{w} - \mathrm{y}^{(\mathrm{train})})^T (\mathrm{X}^{(\mathrm{train})}\mathrm{w} - \mathrm{y}^{(\mathrm{train})}) = 0 \\
\Rightarrow\quad & \nabla_w (\mathrm{w}^T \mathrm{X}^{(\mathrm{train})T} \mathrm{X}^{(\mathrm{train})}\mathrm{w} - 2\mathrm{w}^T \mathrm{X}^{(\mathrm{train})T} \mathrm{y}^{(\mathrm{train})} + \mathrm{y}^{(\mathrm{train})T} \mathrm{y}^{(\mathrm{train})}) = 0 \\
\Rightarrow\quad & 2\mathrm{X}^{(\mathrm{train})T} \mathrm{X}^{(\mathrm{train})}\mathrm{w} - 2\mathrm{X}^{(\mathrm{train})T} \mathrm{y}^{(\mathrm{train})} = 0 \\
\Rightarrow\quad & \mathrm{w} = (\mathrm{X}^{(\mathrm{train})T} \mathrm{X}^{(\mathrm{train})})^{-1} \mathrm{X}^{(\mathrm{train})T} \mathrm{y}^{(\mathrm{train})}
\end{aligned}
$$

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

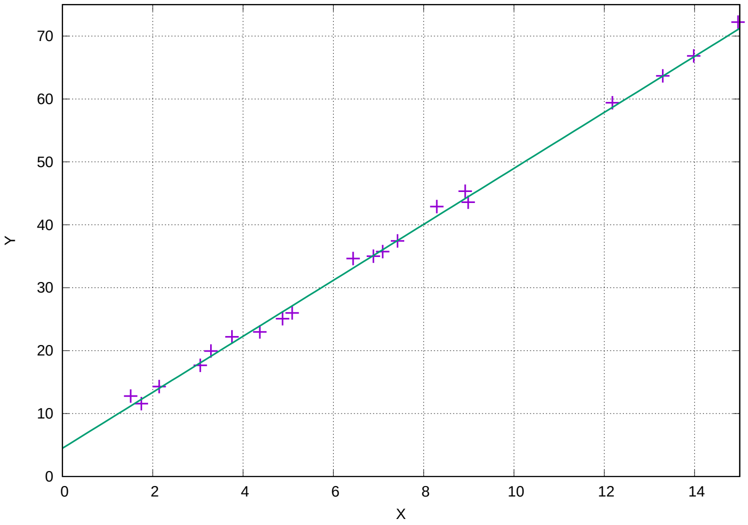
Moore-Penrose Pseudoinverse

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

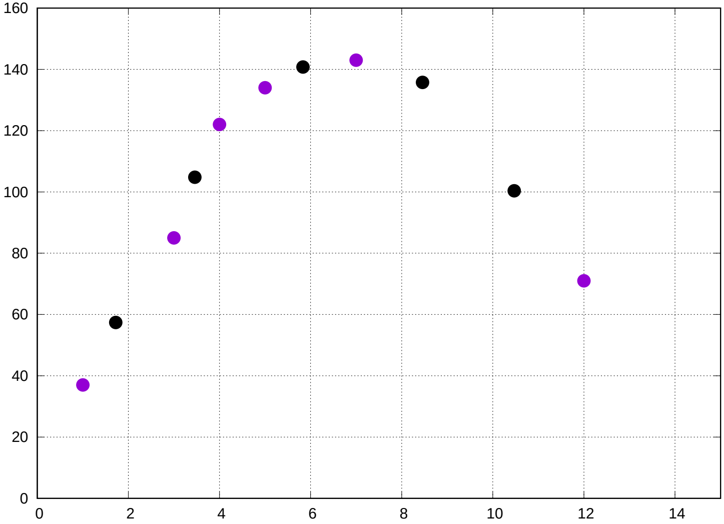
Regression example



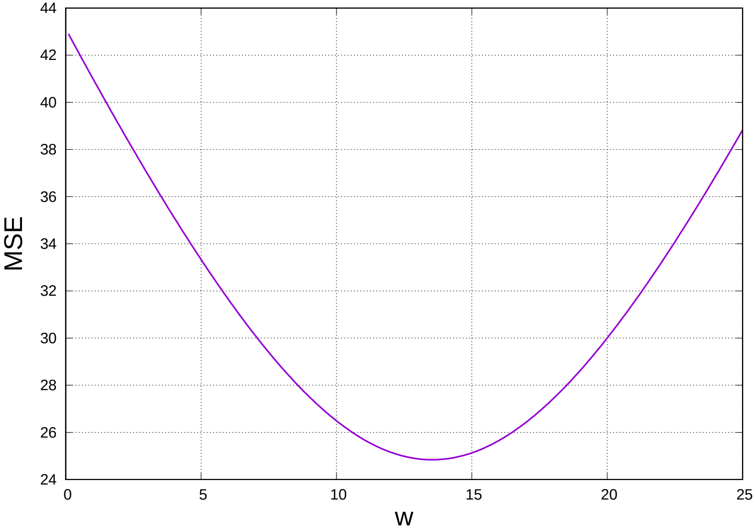
Regression example



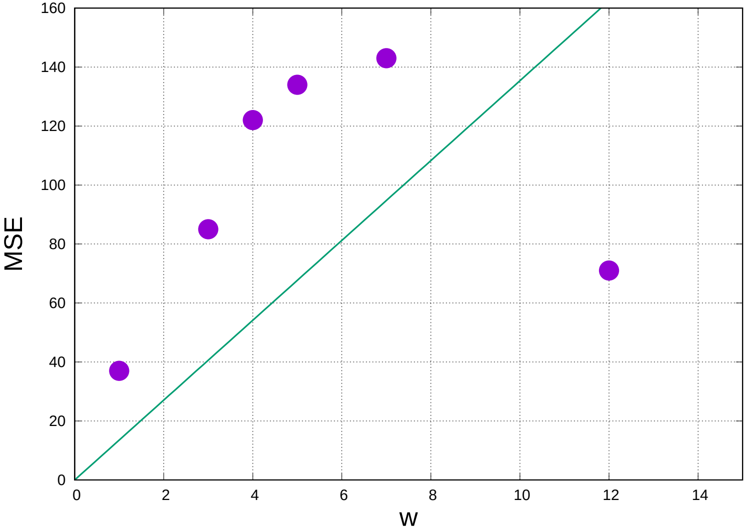
Example



Example: Variation of MSE wrt w



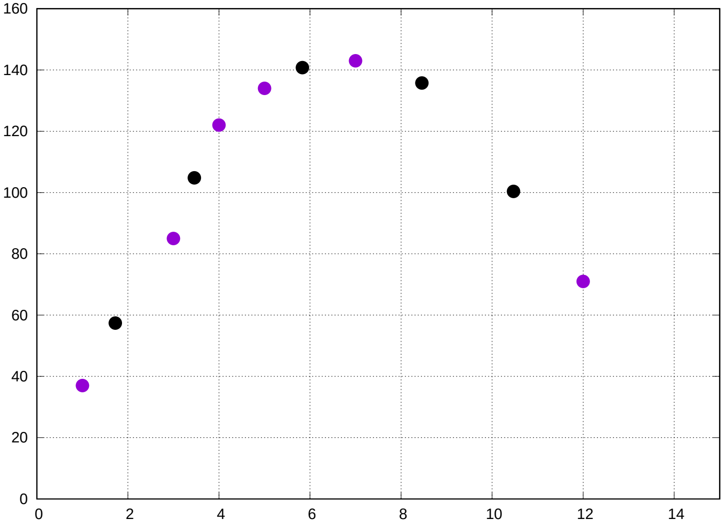
Example: Best fit



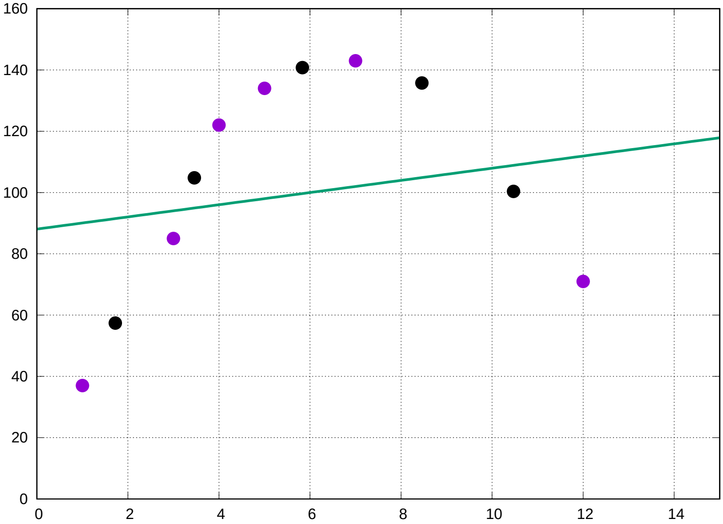
Error

- 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

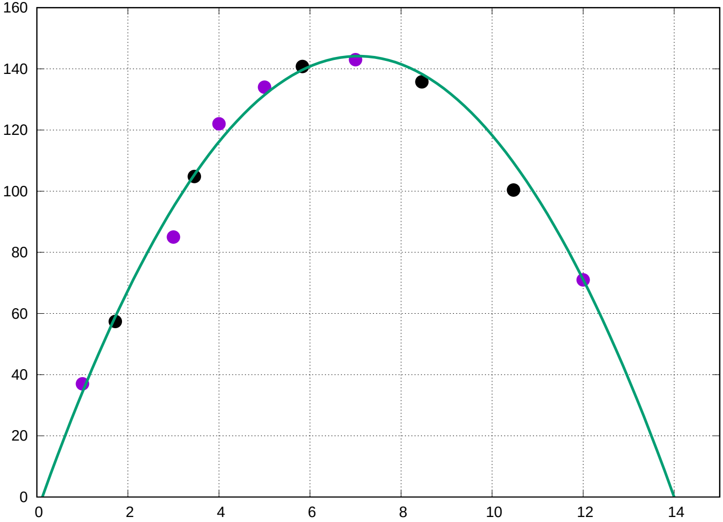
Regression example



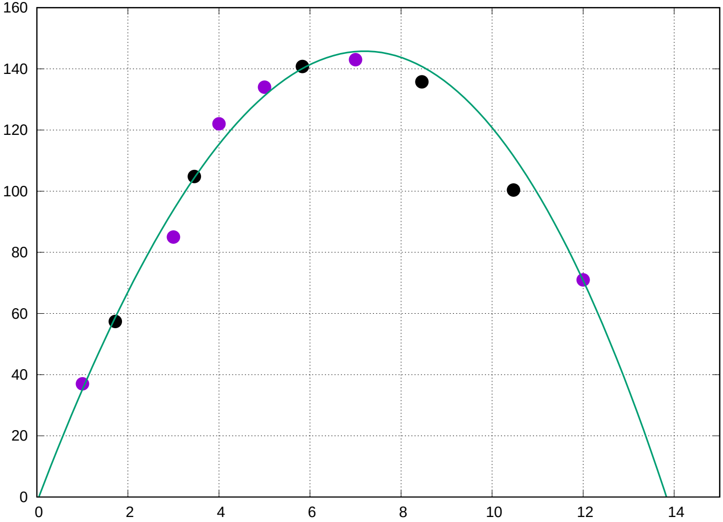
Regression example: degree 1



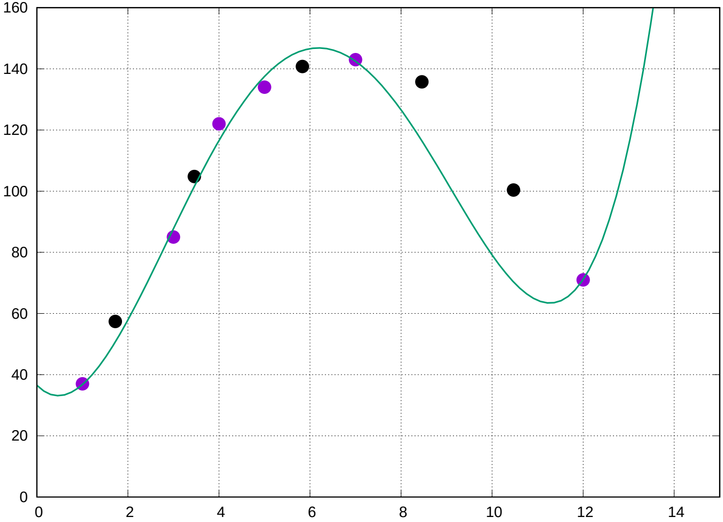
Regression example: degree 2



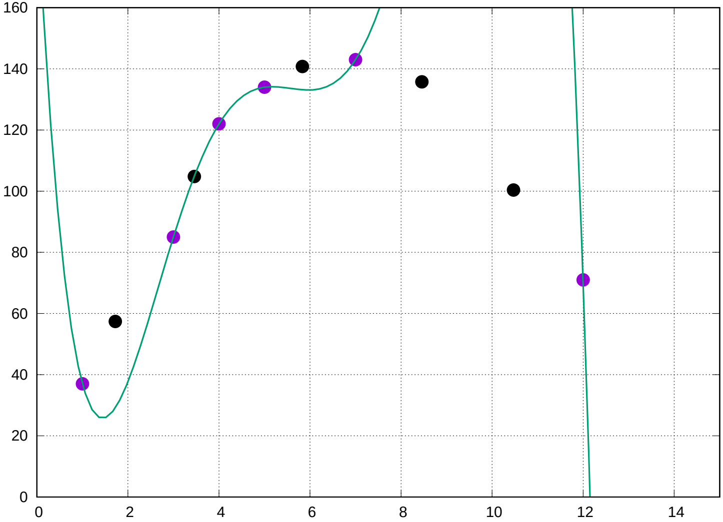
Regression example: degree 3



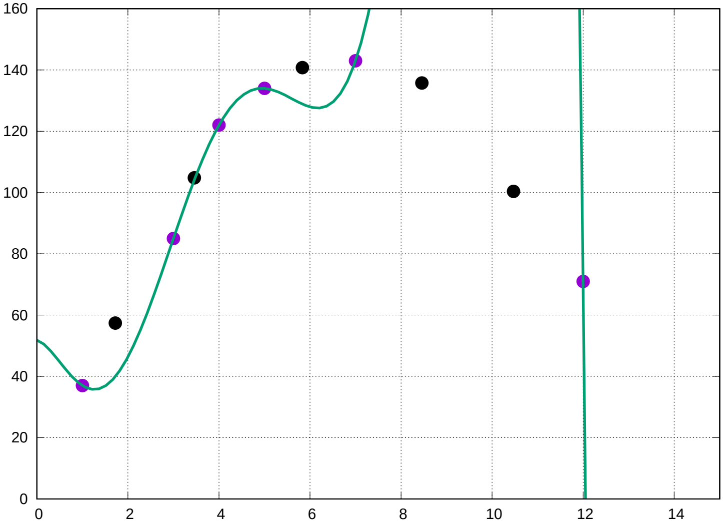
Regression example: degree 4



Regression example: degree 5



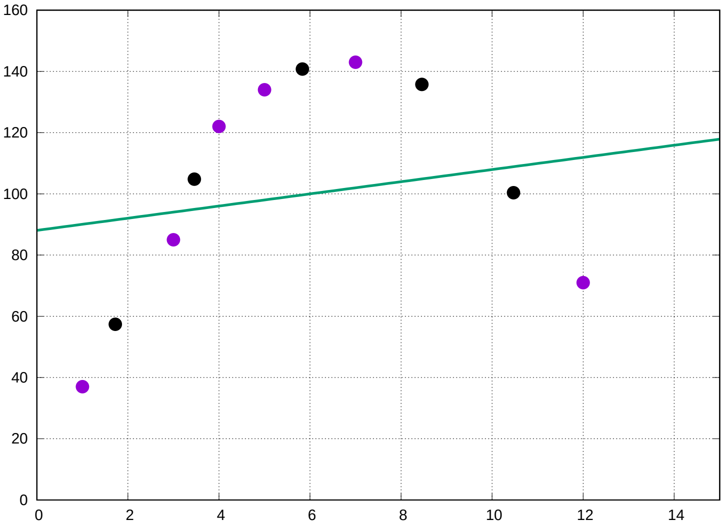
Regression example: degree 6



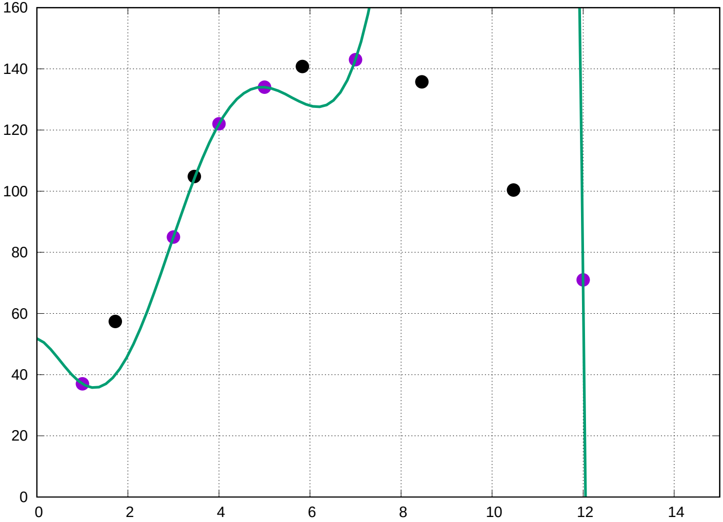
Underfitting & Overfitting

- 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

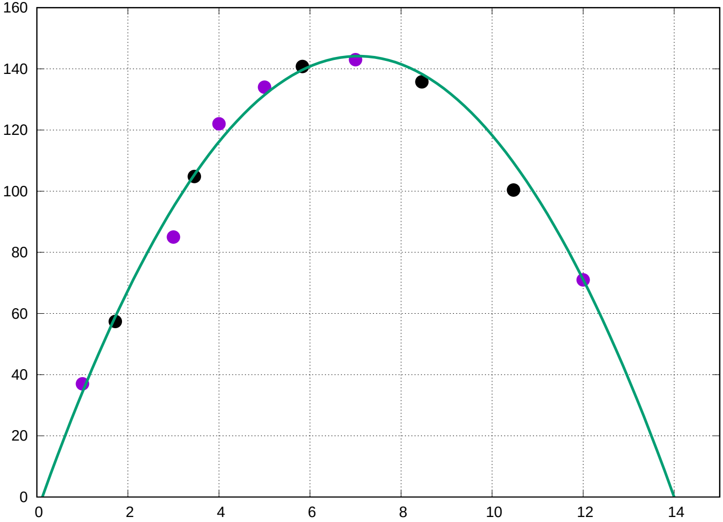
Underfitting example



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 $\hat{y} = b + w_1 x + w_2 x^2$
    - 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

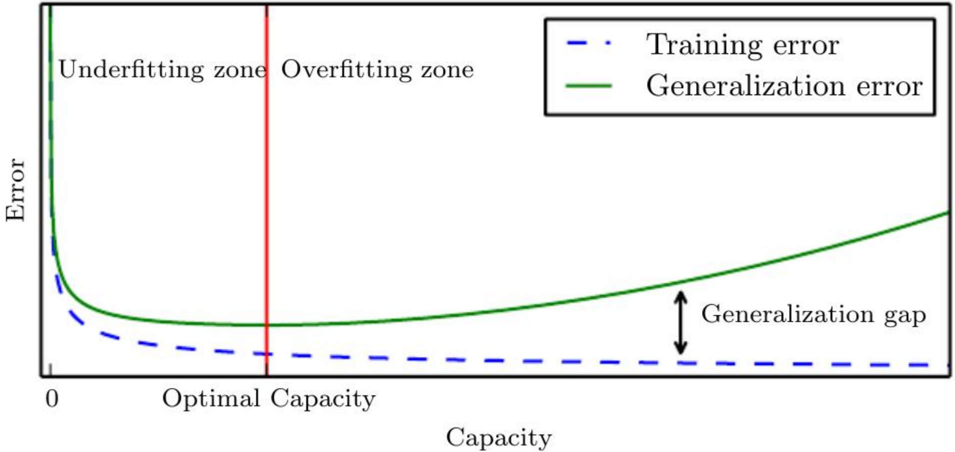


Image source: Deep Learning Book

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