VSB TECHNICAL UNIVERSITY OF OSTRAVA | FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE | DEPARTMENT OF COMPUTER SCIENCE

Fundamentals of Machine Learning

Regression

Jan Platos

November 15, 2023

Regression

Regression - Linear models

- Class of algorithms that is focused on a numerical data.
- Models allow:
    - prediction of the numeric values,
    - classification.
- Elementary model behind the neural network.

Regression - Linear models

Linear regression:

- The class is expressed using linear coefficient.

$$x = w_0 + w_1 a_1 + w_2 a_2 + \cdots + w_k a_k$$

- $a_1, a_2, \ldots, a_k$ are the attribute values,
- $w_0, w_1, \ldots, w_k$ are the weights.

Regression - Linear models

Linear regression:

- The weights are calculated from the training data.
- The prediction for the *i*-th instance is calculated as:

$$w_0 a_0^{(i)} + w_1 a_1^{(i)} + w_2 a_2^{(i)} + \cdots + w_k a_k^{(i)} = \sum_{j=0}^{k} w_j a_j^{(i)}$$

- The important is the difference between the true value y and the predicted one.

Regression - Linear models

Linear regression:

- The least-squares linear regression is to choose the weights w_j to minimize the sum of squares of the differences.

$$\sum_{i=0}^{n}\left(y^{(i)} - \sum_{j=0}^{k} w_j a_j^{(i)}\right)^2$$

Regression - Linear models

Linear regression:

- The classification version may be modified from regression using replacement of the class.
- The first class has assigned 0 and the second has 1.
- The predicted value may be understand as a probability or a membership.

Regression - Linear models

- The goal is to find a linear model that is able to predict the true value y from the input vector x.
- The expected value $\overline{y}$ is expressed using linear coefficient.

$$\overline{y} = w_0 x_0 + w_1 x_1 + w_2 x_2 + \cdots + w_k x_k$$

- x_0 is always 1 and represents the bias.
- $x_1, x_2, \ldots, x_k$ are the attribute values,
- $w_0, w_1, \ldots, w_k$ are the weights.

Regression - Linear models

- The error function is defined as:

$$\sum_{i=0}^{n} \left(y^{(i)} - \sum_{j=0}^{k} w_j x_j^{(i)} \right)^2$$

- The goal is to find the weights to minimize the error.

$$\min_{w} \left\{ \sum_{i=0}^{n} \left(y^{(i)} - \sum_{j=0}^{k} w_j x_j^{(i)} \right)^2 \right\}$$

Regression - Linear models

$$\min_{w}\left\{\sum_{i=0}^{n}\left(y^{(i)}-\sum_{j=0}^{k}w_jx_j^{(i)}\right)^2\right\}$$

- The solution may be find using:
    - Ordinary Least Squares algorithm.
    - Gradient Descent (a learning rate need to be set and iterative approach is processed).

Regression - Linear models - Regularization

- The weights computed by the optimization algorithm may exceeds some limits and/or may contains many small numbers.
- Such weights means over-fitting - too big specialization to the training data.

Regression - Linear models - Regularization

- Lasso regression
  - Minimizes the sum of weights.
  - Eliminates small weight in favor to more important ones.

$$\min_{w}\left\{\sum_{i=0}^{n}\left(y^{(i)}-\sum_{j=0}^{k}w_jx_j^{(i)}\right)^2+\alpha\sum_{j=0}^{k}|w_j|\right\}$$

Regression - Linear models - Regularization

- Ridge regression
  - Minimizes the sum of squares of the weights (a norm o the weight vector).
  - Suppress large values in favor of smaller and more universal ones.

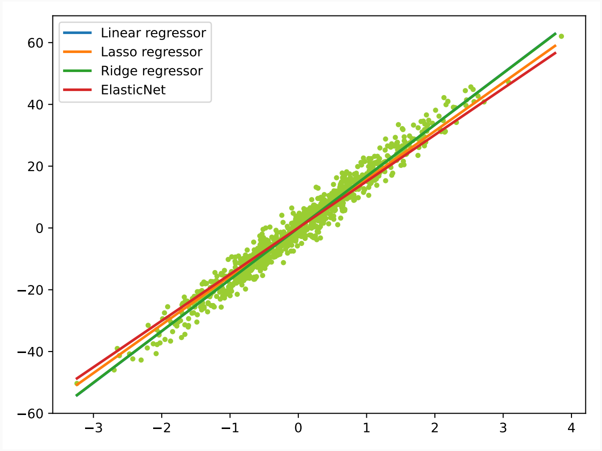
$$\min_{w}\left\{\sum_{i=0}^{n}\left(y^{(i)}-\sum_{j=0}^{k}w_jx_j^{(i)}\right)^2+\beta\sum_{j=0}^{k}|w_j|^2\right\}$$

Regression - Linear models - Regularization

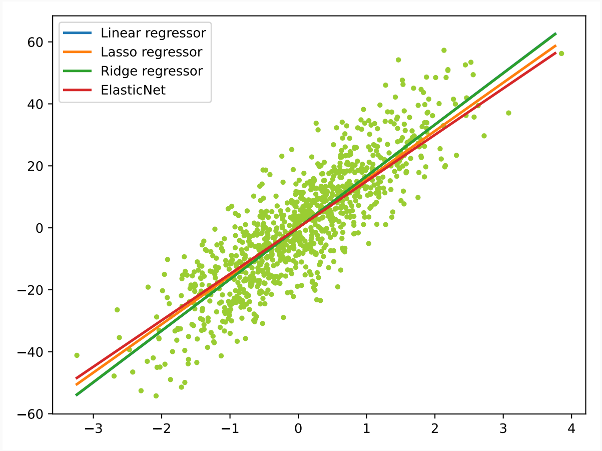
- Elastic Net
  - Combines both regularization to gain benefit from them.

$$\min_{w} \left\{ \sum_{i=0}^{n} \left(y^{(i)} - \sum_{j=0}^{k} w_j x_j^{(i)} \right)^2 + \alpha \sum_{j=0}^{k} |w_j| + \beta \sum_{j=0}^{k} |w_j|^2 \right\}$$

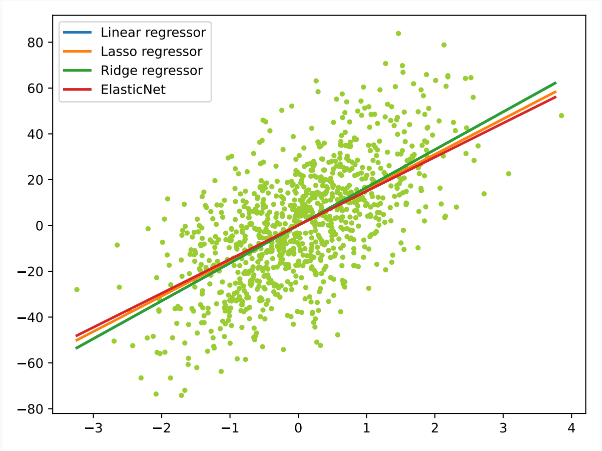
Regression - Linear models



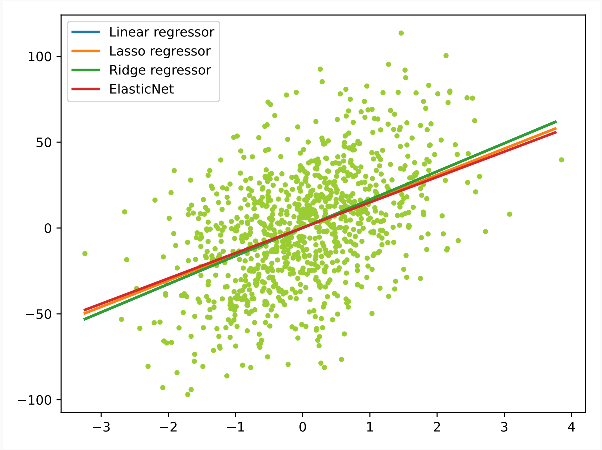
Regression - Linear models



Regression - Linear models



Regression - Linear models



Regression - Single-layer Neural Network (Perceptron)

- The structure has two layers.
  - The input layer has one node for each input attribute.
  - The input node only transmit the input value to the output node.
  - The connection between input and output nodes are weighted.
  - The output layer consist of one output neuron.
  - The output neuron computes the output value.
- The class labels are from the set of $\{-1, +1\}$.

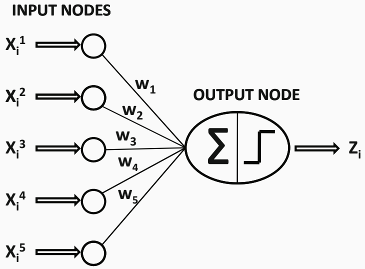


Figure 1: The Perceptron

Regression - Single-layer Neural Network (Perceptron)

- The weighted inputs are transformed into output value.
- The value in drawn from the set $\{-1, +1\}$.
- The value may be interpreted as the perceptron prediction of the class variable.
- The weights $W = \{w_1, \ldots, w_d\}$ are modified when the predicted output does not match expected value.

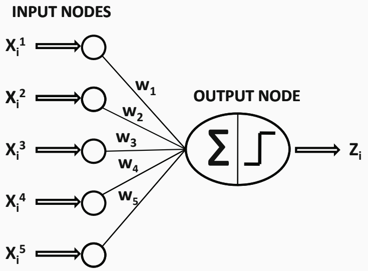


Figure 2: The Perceptron

Regression - Single-layer Neural Network (Perceptron)

- The function learned by the perceptron is referred as *activation function*.
- The function is usually signed linear function (e.g. weighted sum).
- The $W = \{w_1, \ldots, w_d\}$ are the weights for the connections of d different inputs to the output neuron.
- The d is also the dimensionality of the data.
- The b is the bias associated with the activation function.
- The output $z_i \in \{-1, +1\}$ is for the data record $\overline{X_i} = (x_i^1, \ldots, x_i^d)$ computed as follows:

$$z_i = sign\left\{\sum_{j=1}^{d} w_j x_i^j + b\right\} = sign\left\{\overline{W} \cdot \overline{X_i} + b\right\}$$

Regression - Single-layer Neural Network (Perceptron)

- The difference between the prediction of the class value z_i and the real class value y_i is $(y_i - z_i) \in \{-2, 0, 2\}$.
- The result is 0 when the prediction and reality is the same.
- The weight vector $\overline{W}$ and bias b need to be updated, based on the error $(y_i - z_i)$.
- The learning process is iterative.
- The weight update rule for i-th input point $\overline{X_i}$ in t-th iteration is as follows:

$$\overline{W}^{t+1} = \overline{W}^{t} + \eta(y_i - z_i)\overline{X_i}$$

- The η is the learning rate that regulate the learning speed.
- Each cycle per input points in the learning phase is referred as an *epoch*.

Regression - Single-layer Neural Network (Perceptron)

$$\overline{W}^{t+1} = \overline{W}^{t} + \eta(y_i - z_i)\overline{X_i}$$

- The incremental term $(y_i - z_i)\overline{X_i}$ is the approximation of the negative of the gradient of the least=squares prediction error $(y_i - z_i)^2 = \left(y_i - sign\left(\overline{W} \cdot \overline{X_i} - b\right)\right)^2$
- The update is performed on a tuple-by-tuple basis not a global over whole dataset.
- The perceptron may be considered a modified version of a gradient descent method that minimizes the squared error of prediction.

Regression - Single-layer Neural Network (Perceptron)

- The size of the η affect the speed of the convergence and the quality of the solution.
  - The higher value of η means faster convergence, but suboptimal solution may be found.
  - Lower values of η results in higher-quality solutions with slow convergence.
- In practice, η is decreased systematically with increasing number of epochs performed.
- Higher values at the beginning allows bigger jumps in weight space and lower values later allows precise setting of the weights.

Regression - Multi-layer Neural Network

- The perceptron, with only one computational neuron produces only a linear model.
- Multi-layer perceptron adds a hidden layer beside the input and output layer.
- The hidden layer itself may consist of different type of topology (e.g. several layers).

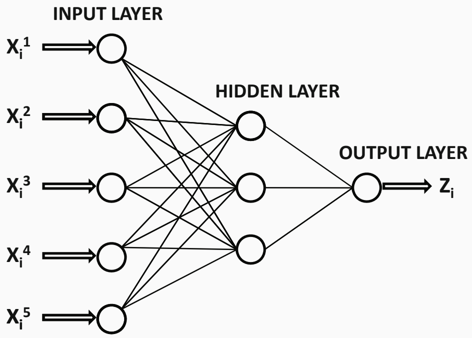


Figure 3: Multi-layer neural network

Regression - Multi-layer Neural Network

- The output of nodes in one layer feed the inputs of the nodes in the next layer - this behavior is called *feed-forward network*.
- The nodes in one layer are fully connected to the neurons in the previous layer.

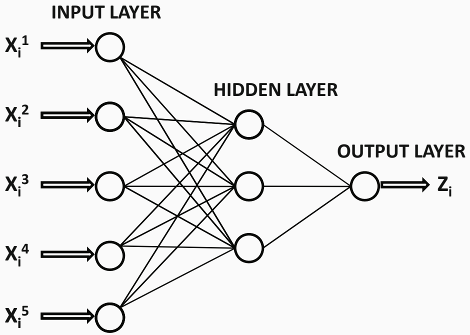


Figure 4: Multi-layer neural network

Regression - Multi-layer Neural Network

- The topology of the multi-layer feed-forward network is determined automatically.
- The perceptron may be considered as a single-layer feed-forward neural network.
- The number of layers and the number of nodes in each layer have to be determined manually.
- Standard multi-layer network uses only one hidden layer, i.e. this is considered as a two-layer feed forward neural network.
- The activation function is not limited to linear signed weighted sum, other functions such as logistic, sigmoid or hyperbolic tangents are allowed.

Regression - Multi-layer Neural Network

Sigmoid/Logistic function	$\sigma(x) = \frac{1}{1+e^{-x}}$
TanH	$\mathbf{tanh}(x) = \frac{(e^{x}-e^{-x})}{(e^{x}+e^{-x})}$
ReLU (Rectified linear unit)	$f(x) = \begin{cases} 0 & for x \leq 0 \\ x & for x \geq 0 \end{cases}$
Sinc	$f(x) = \begin{cases} 1 & for x = 0 \\ \frac{sin(x)}{x} & for x \neq 0 \end{cases}$
Gaussian	$f(x) = e^{x^2}$
Softmax	$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$

Regression - Multi-layer Neural Network

- The learning phase is more complicated than the one in perceptron.
- The biggest problem is the get the error in the hidden layer, because the direct class label is not defined on this level.
- Some kind of *feedback* is required from the nodes in the forward layer to the nodes in earlier layers about the *expected* outputs and corresponding errors.
- This principle is realized in the *back-propagation* algorithm.

Regression - Multi-layer Neural Network

Back-propagation algorithm

- *Forward phase:*
    - The input is fed into input neurons.
    - The computed values are propagated using the current weights to the next layers.
    - The final predicted output is compared with the class label and the error is determined.

Regression - Multi-layer Neural Network

Back-propagation algorithm

- *Backward phase:*
    - The main goal is to learn weights in the backward direction by providing the error estimation from later layers to the earlier layers.
    - The estimation in the hidden layer is computed as a function of the error estimate and weight is the layers ahead.
    - The error is estimated again using the gradient method.
    - The process is complicated by the using of non-linear functions n the inner nodes.

Regression - Multi-layer Neural Network - Back-propagation alg.

- Lets have an example multi-layer neural network with single output neuron.
- In each iteration do take the i-th input vector.
- Pass it through the networks using the forward pass.
- Compare the i-th output o_i to the expected value y_i.
- Compute the error and update the weight using the learning rate η.
- The goal is to optimize the weights w_i to minimize the error function of the differences between y_i and o_i.

Regression - Multi-layer Neural Network - Back-propagation alg.

- The error function E over whole dataset of size n may be defined as follows:

$$E = \frac{1}{2}\sum_{i=0}^{n}(y_i - o_i)^2$$

- The weights of the neurons must be adapted according to the error produced by the neuron weight.

$$w_{i+1} = -\eta\frac{\partial E}{\partial w_i} + \mu w_i$$

Regression - Multi-layer Neural Network - Back-propagation alg.

- The partial derivation may be computed using so called chain rule.

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial z} \cdot \frac{\partial z}{\partial w_i}$$

- where

$$y = \frac{1}{1 + e^{-\lambda z}} \qquad z = \sum_{i=0}^{m} w_i x_i$$

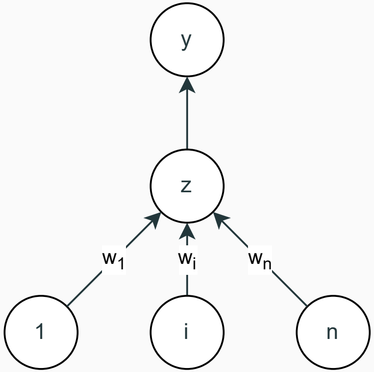
- therefore

$$\frac{\partial z}{\partial w_i} = x_i \qquad \frac{\partial y}{\partial z} = y \cdot (1 - y)\lambda$$

Regression - Multi-layer Neural Network - Back-propagation alg.

- The first partial derivation computation differs for neuron from output and hidden layer.
- The solution for the output layer and *i*-th output is as follows:

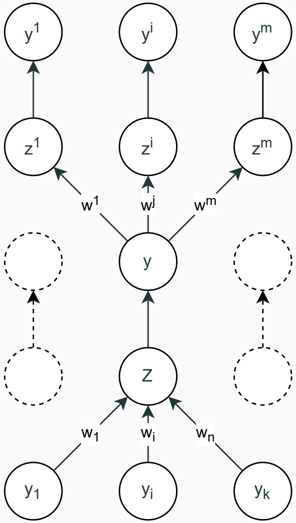
$$\frac{\partial E}{\partial y} = (y_i - o_i)$$



Regression - Multi-layer Neural Network - Back-propagation alg.

- The solution for the hidden layer and *i*-th output is as follows:

$$\frac{\partial E}{\partial y} = \sum_{j=0}^{m} \frac{\partial E}{\partial z^j} \cdot \frac{\partial z^j}{\partial y} = \sum_{j=0}^{m} \frac{\partial E}{\partial z^j} \cdot w^j$$



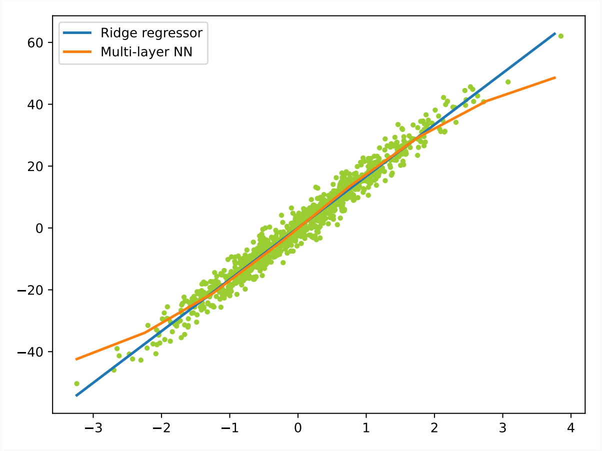
Regression - Multi-layer Neural Network

- It has ability not only to capture decision boundaries of arbitrary shapes, but also non-contiguous class distribution with different decision boundaries in different regions.
- With increasing number of nodes and layers, virtually any function may be approximated.
- **The neural networks are universal function approximate**.

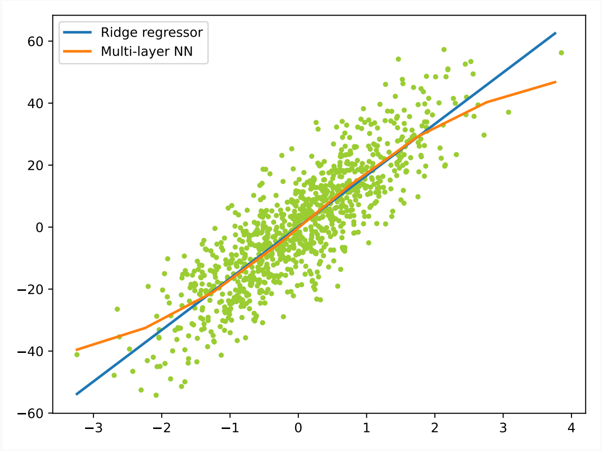
Regression - Multi-layer Neural Network

- This generality brings several challenges that have to be dealt with:
    - The design of the topology presents many trade=off challenges for the analyst.
    - Higher number of nodes and layers provides greater generality but also the risk of over-fitting.
    - There is very little guidance provided from the data.
    - The neural network has poor interpretability associated with the classification process.
    - The learning process is very slow and sensitive to the noise.
    - Larger networks has very slow learning process.

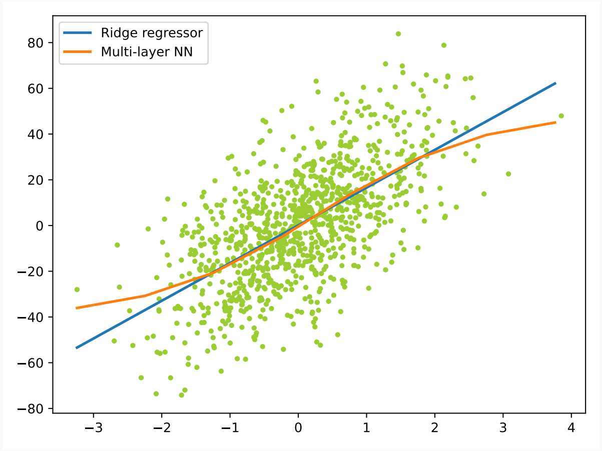
Regression - Multi-layer Neural Network



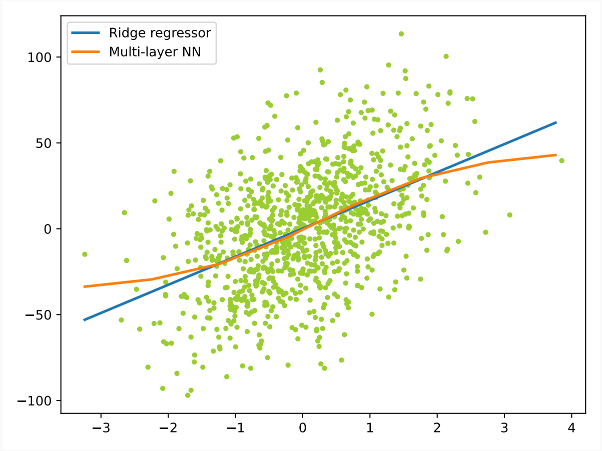
Regression - Multi-layer Neural Network



Regression - Multi-layer Neural Network



Regression - Multi-layer Neural Network



Regression - Regression Trees

- In reality, local linear regression may be quite effective even when the relationships is nonlinear.
- This is used in Regression Trees.
- Each test instance is classified with its locally optimized linear regression by determining its appropriate partition.
- The partition is determined using split criteria in the internal nodes, i.e. the same as the Decision trees.

Regression - Regression Trees

- The general strategy of tree construction is the same as for Decision Trees.
- The splits are univariate (single variable/axis parallel).
- The changes are done in splitting criterion determination and in the pruning.
- The number of points used for training need to be high to avoid over-fitting

Regression - Regression Trees

Splitting criterion

- Due to numeric nature of the class variable, error-based measure have to be used instead of entropy or Gini index.
- The regression modeling is applied on each child resulting from potential split.
- The aggregated squared error of prediction of all training points is computed.

Regression - Regression Trees

Splitting criterion

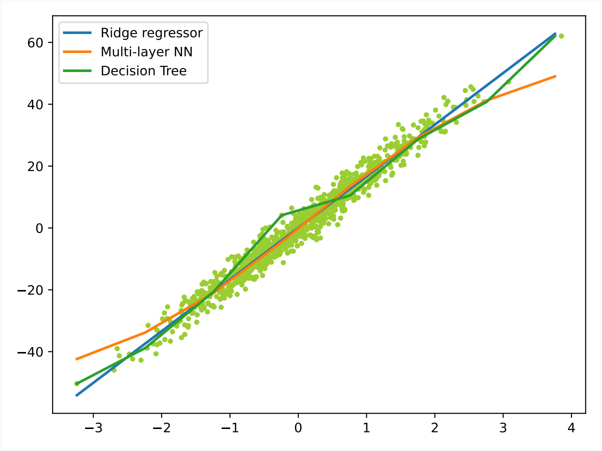
- The split point with the minimum aggregated error is selected.
- The complete regression modeling is computationally very expensive.
- An average variance of the numeric class variable may be used instead.
- The linear regression models are constructed at the leaf nodes after the tree is created.
- This results in larger trees but it its computational expensiveness is much lower.

Regression - Regression Trees

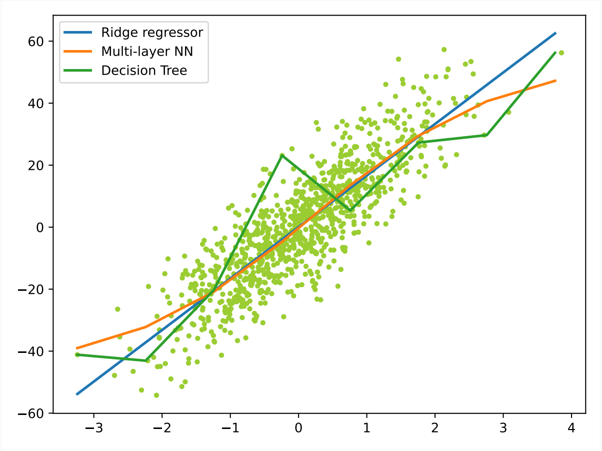
Pruning criterion

- A portion of the training data is not used during construction phase.
- This set is used for evaluation of the squared error of the prediction.
- Leaf nodes are iteratively removed if the accuracy not decreases.

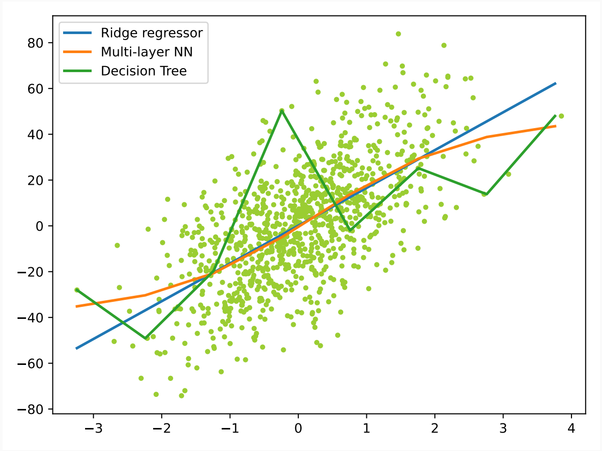
Regression - Regression Trees



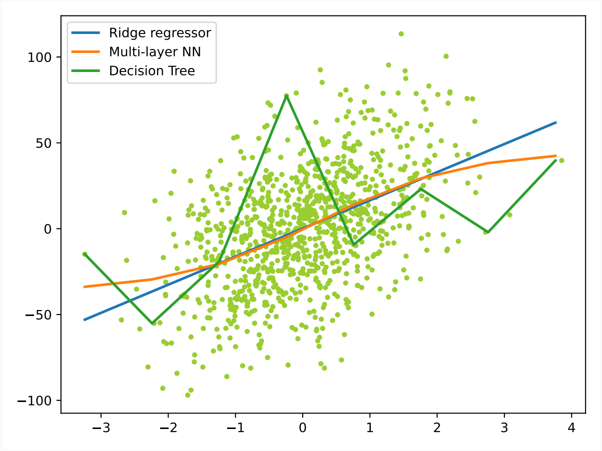
Regression - Regression Trees



Regression - Regression Trees



Regression - Regression Trees



Regression- Assessing Model Effectiveness

- **Mean Absolute Error (MAE)** - is the average of the absolute difference between the predicted and actual value. It is highly affected by outliers.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - g(\overline{X_i})|$$

Regression- Assessing Model Effectiveness

- **Mean Squared Error (MSE)** - is the average of the squared difference between the predicted and actual value. It is differentiable and may be used for optimization.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - g(\overline{X}_i)\right)^2$$

- **Root Mean Squared Error (RMSE)** - is the square root of the average of the squared difference of the predicted and actual value. The root mean is able penalize large errors.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - g\left(\overline{X}_i\right)\right)^2}$$

Regression- Assessing Model Effectiveness

- The effectiveness of the linear regression models can be evaluated with a measure known as **R²-*statistics*** or *coefficient of determination*.
- The standard Sum of Squared Error is defined for a model $g(\overline{X})$ as:

$$SSE = \sum_{i=1}^{n} \left(y_i - g(\overline{X_i})\right)^2$$

- The Squared Error of the response variable about its mean is defined as:

$$SST = \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{n} \frac{y_j}{n}\right)^2 = \sum_{i=1}^{n} (y_i - \overline{y})^2$$

Regression- Assessing Model Effectiveness

- The R^2-*statistics* is then defined as:

$$R^2 = 1 - \frac{SSE}{SST}$$

- The value is always between 0 and 1 and higher are more desirable.
- For high dimension data, **adjusted** version is more accurate:

$$R^2 = 1 - \frac{(n-d)SSE}{(n-1)SST}$$

- The R^2-*statistics* is not applicable on the nonlinear models.
- The nonlinear regression may be evaluated using pure SSE.

Regression- Assessing Model Effectiveness

- **Mean Average Percentage Error (MAPE)** - is the average percentage error between the predicted and actual value.

$$MAPE = \frac{100}{n} \sum_{i=1}^{n} \left| \frac{y_i - g(\overline{X}_i)}{y_i} \right|$$

- **Symmetric Mean Average Percentage Error (SMAPE)** - is the symmetric average percentage error between the predicted and actual value.

$$SMAPE = \frac{100}{n} \sum_{i=1}^{n} \frac{|y_i - g(\overline{X}_i)|}{\frac{|y_i| + |g(\overline{X}_i)|}{2}}$$

Questions