Linear Regression

Univariate Linear Regression

$$J(\theta_0,\theta_1)=\frac{1}{2M}\sum_{m=1}^{M}\left(y^{(m)}-\hat{y}^{(m)}\right)^2$$

$$\hat{y}^{(m)}=\theta_0+\theta_1 x^{(m)}$$

Mulivariate Linear Regression

$M=$ number of samples

$N=$ number of features

$$x_0=1$$

$$X_m=\left[x_0,x_1^{(m)},x_2^{(m)},...,x_n^{(m)}\right]^T$$

$$\hat{y}^{(m)}=\theta^T X_m$$

$$J(\theta)=\frac{1}{2M}\sum_{m=1}^{M}\left(y^{(m)}-\theta^T x^{(m)}\right)^2$$

$$x=[1,x_1,x_2,...,x_n]^T$$

Normal Equation

$$\theta=(X^TX)^{-1}X^Ty \qquad \boldsymbol{X}^T\boldsymbol{X}\vec{\boldsymbol{\theta}}=\boldsymbol{X}^T\vec{\boldsymbol{y}} \quad \text{(LGS from HM2)}$$

- First column of X usually all 1s (bias term)
- Computationally expensive for large datasets (matrix inversion)
- Works for univariate and multivariate

Basic Assumptions

- Linearity: relationship between features and target is linear
- Independence: residuals don't correlate with each other
- Normality: residuals are normally distributed
- Homoscedasticity: constant variance of residuals

Evaluating Regression Models

$$MAE=\frac{\sum_{i=1}^{I}|y^{(i)}-\hat{y}^{(i)}|}{I}$$

$$MSE=\frac{\sum_{i=1}^{I}\left(y^{(i)}-\hat{y}^{(i)}\right)^2}{I}$$

Coefficient of Determination

$$SS_{\text{res}}=\sum_i\left(y^{(i)}-\hat{y}^{(i)}\right)^2$$

$$SS_{\text{tot}}=\sum_i\left(y^{(i)}-|(y)\right)^2$$

$$R^2=1-\frac{SS_{\text{res}}}{SS_{\text{tot}}}$$

- $R^2=0$: model predicts baseline mean
- $R^2=1$: perfect fit to regression line
- $R^2<0$: model worse than mean model

Gradient Descent

- Update $\theta_{j\ \text{updated}}=\theta_j-\alpha\frac{\partial}{\partial\theta_j}J(\theta) \quad \forall j=0,...n$
- α = learning rate (step size)
- Repeats until convergence

Polynomial Regression & Regularization

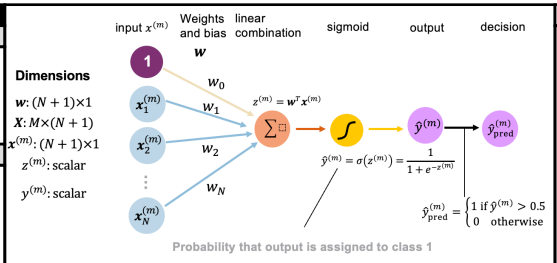
$$J(\theta)=\frac{1}{2M}\left[\sum_{m=1}^{M}\left(y^{(m)}-h_\theta\left(x^{(m)}\right)\right)^2+\lambda\sum_{j=1}^{n}\theta_j^2\right]$$

- λ = regularization parameter

Classification with Logistic Regression

Logistic Regression

$$\hat{y}=h_{\theta(x)}=g\left(\vec{\theta}^T\vec{x}\right)=\frac{1}{1+e^{-\left(\vec{\theta}^T\vec{x}\right)}} \qquad \text{with } x_0=1$$



Supervised Learning - Advanced Topics

Performance Metrics

$$\text{Accuracy}=\frac{\text{TP}+\text{TN}}{\text{M}\ (=\text{TP}+\text{TN}+\text{FP}+\text{FN})}$$

$$\text{Precision}=\frac{\text{TP}\ (=\text{N correct in class})}{\text{TP}+\text{FP}\ (=\text{Sum pred. in class})}$$

$$\text{Recall}=\frac{\text{TP}\ (=\text{N correct in class})}{\text{TP}+\text{FN}\ (=\text{Sum actual in class})}$$

$$\text{F1 Score}=2\cdot\frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}$$

Reading a table with "predicted" rows and "actual" columns:

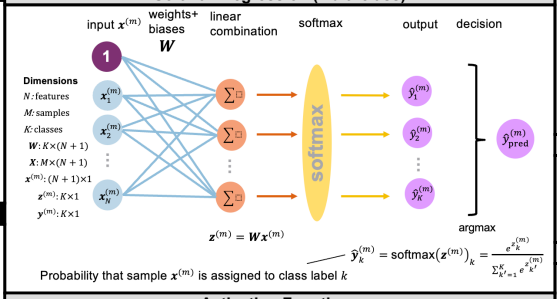
- Cell: **The model predicted the Row label, but actual was the Column label**
- Diagonal: correct predictions
- Row Sum: Total **predicted** of class (use for *precision*)
- Column Sum: Total **actual** of class (use for *recall*)
- Cell Sum (*all*): Total samples M

Bias-Variance Tradeoff

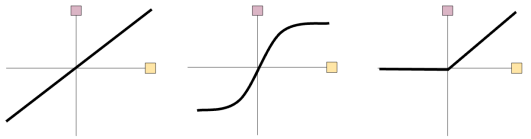
- **Variance**: High Variance means *sensitivity* to training data. Usually overfitting.
- **Bias**: High Bias means *systematic* errors. Usually underfitting.
- **Tradeoff**: Decreasing bias often increases variance and vice versa. Goal is to find balance for best generalization.

Neural Networks)

Softmax Regression (Multi-class)



Activation Functions

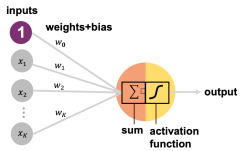


LINEAR
like linear regression
(only used in output layer)

LOGISTIC / SIGMOIDAL / TANH
Smooth, differentiable, saturating functions

RECTIFIED LINEAR (ReLU)
Cheap to compute, very popular

Single Neuron



Cost Functions

Regression task:

$$MSE=\frac{1}{M}\sum_{m=1}^{M}\left(y^{(m)}-\hat{y}^{(m)}\right)^2$$

Binary classification task:

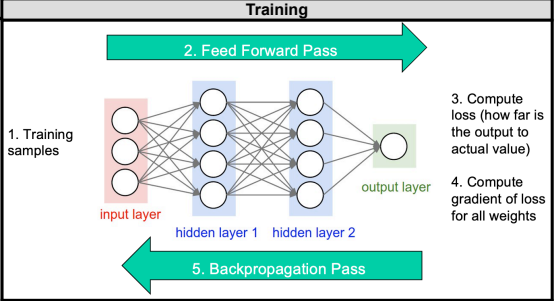
$$L_{\text{logistic}}=-\sum_{m=1}^{M}y^{(m)}\log\left(\hat{y}^{(m)}\right)+\left(1-y^{(m)}\right)\log\left(1-\hat{y}^{(m)}\right)$$

Multiclass (multinomial) classification:

$$L_{CE}==-\sum_{m=1}^{M}\log\frac{\exp\left(\boldsymbol{w}_{c_m}^T\boldsymbol{x}^{(m)}\right)}{\sum_{k=1}^{K}\exp\left(\boldsymbol{w}_k^T\boldsymbol{x}^{(m)}\right)}$$

where c_m the correct class (label) for the m-th sample.

Training



1. Initialize all network parameters (the weights) randomly
2. Compute the **gradient of the cost function** with respect to each of the model parameters
3. **Adjust the model parameters** by a small step α (the learning rate) in the opposite direction of the gradient:

$$\boldsymbol{w}_{\text{updated}}=\boldsymbol{w}_{\text{before}}-\alpha\frac{\partial L}{\partial \boldsymbol{w}}$$

Chain Rule & Backpropagation

$$\frac{\partial z}{\partial x}=\frac{\partial y}{\partial x}\frac{\partial z}{\partial y}$$

$$z=f(y),y=g(x)$$

NN Parameter Counting

- Example: 3 inputs → 25 hidden → 12 outputs
  - Input→Hidden: $(3\times25)+\mathbf{25}=100$
  - Hidden→Output: $(25\times12)+\mathbf{12}=312$
  - Total: 412

Regularization Techniques

Dropout
- Temporarily deactivates random nodes during each training iteration

Early Stopping (Early Termination)
- Stops training when validation error rises
- Prevents overfitting to training noise

Data Augmentation
- Artificially expands training dataset with label-preserving transformations

Universal Approximation Theorem

Universal Approximation Theorem
- Feedforward network with single hidden layer can approximate any continuous function
- Requires arbitrarily large (potentially infinite) neurons, not fixed small number
- Output layer: 1 node (regression/binary classification), multiple nodes (multi-class)
- Hidden layer node count ≠ input node count (flexible hyperparameter)

Support Vector Machines (SVM)

Maximum Margin Classifier

$$y^{(m)}\left(b+\vec{\boldsymbol{w}}^T\vec{\boldsymbol{x}}^{(m)}\right)>0$$

$$\min_{b,\vec{\boldsymbol{w}}}\frac{1}{2}\|\vec{\boldsymbol{w}}\|^2$$

- Min problem usually cast as lagrangian problem and solved with dedicated algorithms (quadratic optimization)

Soft Margin Classifier

$$\min_{b,\vec{\boldsymbol{w}},\varepsilon}\frac{1}{2}\|\vec{\boldsymbol{w}}\|^2+C\sum_{m=1}^{M}\varepsilon_m$$

subject to

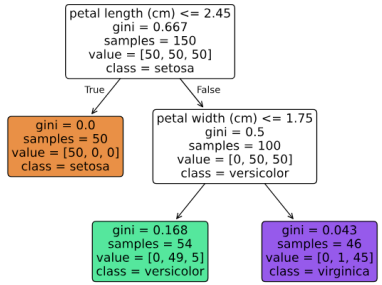
$$y^{(m)}\left(b+\vec{\boldsymbol{w}}^T\vec{\boldsymbol{x}}^{(m)}\right)\geq1-\varepsilon_m\forall m=1,...,M$$

- C tunable hyperparameter, chosen via cross-validation

Kernel Trick

$$\mathcal{K}\left(\boldsymbol{x}^{(m)},\boldsymbol{x}^{(m')}\right)=\left(1+\sum_{n=1}^{N}x_n^{(m)}x_n^{(m')}\right)^d$$

Decision Trees



Gini Impurity

$$G(Q_i)=1-\sum_{k=1}^{K}p_{i,k}^2$$

- **pure** $\text{gini}=0$ (all samples same class)
- root node is maximally impure

Regularization

- **max_depth**
- **min_samples_split**
- **min_samples_leaf**
- **min_samples_leaf**
- **min_weight_fraction_leaf**
- **max_leaf_nodes**
- **max_features**

Increasing `min_*` or reducing `max_*` hyperparameters will regularize the model.

Generative Models and Naive Bayes

$$P(B\mid A)=\frac{P(A\cap B)}{P(A)}$$

$$P(A\mid B)=\frac{P(A)\cdot P(B|A)}{P(B)}$$

Clustering

Types of Clustering

- **Soft**: data points can belong to multiple clusters with varying degrees of membership (e.g., Gaussian Mixture Models)
- **Hard**: data points belong to exactly one cluster (e.g., K-Means)

Elbow Method

- Plot within-cluster sum of squares (WCSS) vs number of clusters (k)
- Look for "elbow" point where adding more clusters doesn't significantly reduce WCSS
- Not always clear-cut, can be subjective

Silhouette Score

- a_m = avg dist to points in same cluster
- b_m = avg dist to points in nearest cluster
- $s_m\in[-1,1]$ defined as $s_m=\frac{b_m-a_m}{\max(b_m,a_m)}$

QUIZZES

Linear Regression

- A residual for a sample is the difference between the expected and predicted output value.
- Residual plots show the residual values on the vertical axis.
- Residuals are **not universally bounded** (e.g., they are not always between 0 and 1 or −1 and 1); their range depends entirely on the scale of the target variable.
- Residual plots **do not always** have the predicted value on the horizontal axis; they can also plot the independent features on the horizontal axis.
- The larger the coefficient of determination R^2, the closer the points scatter to the regression line.
- If a model always predicts the mean of all expected outcomes, then R^2 equals 0.
- A **larger** coefficient of determination R^2 indicates that the regression model fits the data **better**, not a smaller one.
- The values of R^2 are **always between 0 and 1** (for training data), not between −1 and 1.
- The goal of Linear Regression is to minimize the value of the **cost function** (e.g., Mean Squared Error).
- There exists a closed formula (the Normal Equation) for both univariate and multivariate linear regression to compute the optimal parameter values.
- The parameter θ is a parameter of the **hypothesis function**, **not** a parameter of the cost function (it is the variable the cost function optimizes).
- In Multivariate Linear Regression, the relationship between the features and the output is strictly modeled as a linear combination.

Gradient Descent

- In Multivariate Linear Regression, each sample is used **at least once** per iteration (typically exactly once per epoch in standard batch gradient descent).
- Running Gradient Descent multiple times with the same learning rate α and the same initial values of θ will **always result in the same solution** because it is a deterministic algorithm.
- The cost function **does not necessarily decrease in each iteration**; if the learning rate is too large, it might increase or diverge.
- If a learning curve is unstable or diverging, the learning rate is **probably too large**.
- The primary problem with overfitting is that the model might **not generalize well to new datapoints**.
- With hyperparameter tuning, we **cannot always explore all possible settings**, especially for continuous or unbounded hyperparameter spaces.
- For polynomial regression, we can reduce the risk of overfitting by **decreasing**, not increasing, the degree of the polynomials.
- Regularization in polynomial regression is used to **penalize large parameter weights**, not to control the size of the learning rate α.

Logistic Regression

- Classification problems involve predicting discrete categories, such as:
- Using hockey players' statistics to predict which of two teams will probably win a match.
- Analyzing a song to determine if it is classical, techno, pop, or rock.
- Analyzing a text to determine if the sentiment is happy or sad.

- Inspecting a patient's medical record to estimate how many hours of exercise they do per week is a **regression** problem, not classification.
- The logistic sigmoid function $g(z)$:
- Has an "s"-shape.
- Never intersects the x-axis (it asymptotes at 0 and 1).
- When used as part of logistic regression, its value can be interpreted as how **"confident"** the model is about its prediction.
- It is a **continuous, smooth curve**, it does not jump abruptly at the origin.
- It strictly **increases** from left to right; it does not decrease.

- If a logistic regression model produces an output of just above 0.5 for a given input, it means the model is **only slightly confident that the outcome is positive**.
- For the training process, the gradient descent update expressions are mathematically the **same as for linear regression**.
- Instead of the residual sum of squares (RSS), we typically use the **log loss (binary cross-entropy)** cost function to optimize the model parameters, not the Mean Absolute Error (MAE).
- We **cannot choose** to use the normal equation because the logistic loss function has no closed-form solution.
- Important hyperparameters for training a logistic regression model include the **number of gradient descent iterations** and the **learning rate**.

Neural Networks 1

- A neural network with 3 input nodes, 25 nodes in the hidden layer, and 12 output nodes has **412** trainable parameters.
- **Formula:** $(N_{\text{in}} \times N_{\text{hidden}} + N_{\text{hidden}}) + (N_{\text{hidden}} \times N_{\text{out}} + N_{\text{out}})$
- **Calculation:** $(3 \times 25 + 25) + (25 \times 12 + 12) = 100 + 312 = 412$

- The loss used in neural networks quantifies how much the predicted value differs from the true observed value.
- The nodes in a neural network **can** have different activation functions.
- A neural network with one hidden layer **cannot** approximate any continuous function if constrained to only ten neurons (Universal Approximation requires an arbitrary/sufficient number of neurons).
- A neural network can have **more than one node** in the output layer (e.g., one for each class in multi-class classification).
- The number of nodes in the first hidden layer **does not** have to be equal to the number of input nodes.
- Cross entropy is a cost function used primarily to address **classification problems**.
- When applied to a regression task, the output layer of a neural network typically uses a **linear activation function**, not a non-linear one.
- When using gradient descent to minimize the cost function of a neural network, we **do not guarantee finding a very close approximation to the global minimum**, as the loss landscape is highly non-convex and prone to local minima.

Neural Networks 2

- During the network training process (gradient descent), the **weights** and the **biases** are adapted.
- An epoch is completed when the forward and backward pass are completed for **all the samples in the entire dataset**, not just a single batch.
- The gradients in backpropagation are computed starting at the **output layer** and moving backwards, not starting at the input layer.
- In backpropagation, the final parameter updates are obtained by averaging the computed gradients over all training samples (in a batch).
- Backpropagation:
- Updates the parameters of the network.
- Reuses computations from earlier steps for efficiency (dynamic programming).
- Is affected by the vanishing gradients problem.
- It **does not** guarantee finding the global minimum of the cost function.

- Overfitting in neural networks can be addressed by **Dropout** and **Early termination** of the training process.
- Increasing the depth of the network generally **increases** the risk of overfitting, rather than addressing it.
- Decreasing the learning rate or increasing the iterations does not natively prevent overfitting.
- Dropout **temporarily** removes randomly chosen nodes during training iterations; it does not permanently remove them from the architecture.

Decision Trees

- Given a leaf node with Class 1: 15, Class 2: 0, and Class 3: 23 samples, the Gini impurity is **0.478**.
- **Formula:** $1 - \sum_{i=1}^{C} p_i^2$
- **Calculation:** Total samples = 38. $1 - \left(\left(\frac{15}{38}\right)^2 + \left(\frac{0}{38}\right)^2 + \left(\frac{23}{38}\right)^2\right) = 1 - (0.1558 + 0 + 0.3663) \approx 0.478$

- A node has maximal entropy if each class has the **same number of samples** (uniform distribution).
- A leaf node with Class 1: 42 and Class 2: 5 samples assigns a probability of **0.89** to a prediction of the majority class.
- **Formula:** $P(\text{Majority}) = \frac{N_{\text{majority}}}{N_{\text{total}}}$
- **Calculation:** $\frac{42}{42+5} = \frac{42}{47} \approx 0.89$

- A leaf node of a regression tree with target values 25, 32, 25, 42, 27 has a prediction of **30.20**.
- **Formula:** Mean of target values $\frac{\sum y_i}{N}$
- **Calculation:** $\frac{25+32+25+42+27}{5} = \frac{151}{5} = 30.2$

Support Vector Machines (SVM)

- The hyperplane in the context of SVMs is a subspace of the feature space.
- The distance between the hyperplane and every support vector in a maximal margin classifier is **the same**.
- In a maximal margin classifier, the margin is the **shortest distance**, not the average distance, between the separating hyperplane and the closest samples.
- SVMs directly output a decision boundary side or distance metric; they **do not natively return a probability function** $P(y|x)$.
- There are **always at least 2** support vectors.
- Slack variables are used to **avoid overfitting** by allowing for a soft margin.
- The slack variable for a misclassified sample will always have a value **> 1**.
- If there are m samples, there will be **one slack variable for each sample**, not one for each feature.
- The sum of all slack variables is **not always 0**; it is greater than zero if any samples violate the hard margin.
- If data is not linearly separable, you can resolve this by using an **RBF kernel** or a **polynomial kernel**.
- The removal of a single training sample **could alter** the position of the separating hyperplane (specifically, if that removed sample was a support vector).
- To find good values for the hyperparameters C and γ for a soft-margin SVM with an RBF kernel, you can use **Random search** evaluated on a validation set or **Grid search** utilizing the cross-validation performance score.
- The decision boundary derived from a soft margin classifier is always linear **in its respective (potentially higher-dimensional) mapped feature space**.

Naive Bayes & Generative Models

- Imagine an email dataset where 50% are spam. In spam, 70% contain "offer" and 90% contain "free". The joint probability of observing both words given the email is spam is **0.63**.
- **Formula:** Assuming conditional independence, $P(X_1, X_2 \mid Y) = P(X_1 \mid Y) \times P(X_2 \mid Y)$
- **Calculation:** $0.70 \times 0.90 = 0.63$

- **Discriminative models** focus on learning the decision boundary between classes.
- **Generative models**:
- Can be used to generate synthetic data by sampling from the learned probability distribution.
- Aim to understand the underlying probability distribution of the data within each class.

- Bayes' Theorem establishes that the probability of event A happening changes after observing event B. This updated belief relies on how likely we initially thought A was and how much evidence B provides about A.
- For a manufacturing company needing to classify part defects:
- A **Discriminative model** (e.g., Logistic Regression, NN, SVM) is an excellent choice because the goal is to accurately classify the defect type.
- A **Generative model** is also highly relevant because it allows the company to understand the underlying distributions of defects or generate synthetic images of defective parts to augment training data.

Unsupervised Learning

- The total number of clusters K in K-means is **not automatically determined**; it is a hyperparameter that must be set manually.
- Different runs of K-Means **can produce different clustering results** due to random initializations.
- The initial centroids in K-Means are selected as **randomly chosen points from the training set**.
- The **elbow rule** can be used to determine the optimal value for the total number of clusters K.
- DBSCAN is a **hard clustering algorithm**, not a soft clustering algorithm.
- Hyperparameters for DBSCAN include **minPts** and ε.
- The starting/core points in the DBSCAN algorithm are selected as a **random unprocessed point** in the set of data points.

Source Code

https://github.com/josias-r/zhaw-sem5-summaries