MachLe

Summary

Lukas Schöpf

24. Januar 2025

1 Introduction

1.1 Data Types

- Numerical/Quantitive, discret: Countable

  Example: Number of rejected Loans, classes taken this semster

- Numerical/Quantitive, Continuos: Interval data

  Example: Distance, mg of drug taken, size of a house

- Categorical, ordinal: distinct and can be ordered

  Example: Credit score can be {low, medium, high}

- Categorical, nominal: categories cannot be orderd

  Example: gender, eye color

1.2 Machine Learning Paradigms

- Unsupervised Learning: Discover and explore structure from unlabelled data
- Supervised Learning: Learn to predict/forecast an output of interest, we know what we want to predict and labelled data is available

1.2.1 Unsupervised Learning

Tasks:

- Dimensionality reduction
- Feature Learning
- Matrix compilation
- Anomaly detection
- Generating data

1.2.2 Supervised Learning

Given a set of features/attributes for some objects and also the ouput/target value of what we want to predict. The Supervised ML task: Given a new object and its features what would be the output value:

- Regression: Ouput is a numeric value
- Classification: Ouput is a categorical value

1.3 Data Preparation/Preprocessing

Data will rarly be in the format and quality needed for analytics and model training and several of these operations will be needed:

- Data integration/consolidation: Collects and merges data from multiple sources into coherent data store
- Data cleaning: removing or modifying incorrect data, identify and reduce noise in data
- Data transformations: normalize, discretize or aggregate the data
- Data reduction: reduce data size by reducing the number of samples or reducing the number of attributes, balance skewed data

Low quality data will result in low quality results

2 Supervised ML, Similarities, kNN & Performance measurments

2.1 Supervised ML

- Goal: Find the best Hypothesis

$$H^* = \arg\min_{H \in \mathcal{H}} \sum_{i=1}^{n} loss(x_i, y_i)$$

- Loss:

$$loss(x_i, y_i) = (H(x_i) - y_i)^2$$

- Model: $H(x) = a + bx + cx^2 + \cdot$

Task of the Learning Algorithem to find best parameters a, b, c(those that minimize the loss)

2.1.1 Overfitting

Model learns traning data but doesnt generalize well.

2.1.2 Training and test error

Dataset gets split in Training set and Test set (80%/20%)
$error_{train}$: Error form trained model on train set
$error_{test}$: Estimate of the true error (generalization error). Error from trained model on test set.

2.2 kNN

Pros:

- Simple and intuitive
- Multiclass
- Interpretable

Cons:

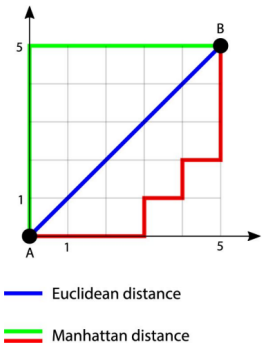
- Curse of Dimensionality
- Sensitive to noise
- Computationally expensive for large datasets

Given training set $X = (x_1, \text{class}_1)(x_2, \text{class}_2)$
Given a new instance $x_?$:

- Find the k-closest examples x_i to $x_?$ in the traning set
- Classify $x_?$ based on the majority vote of $\{x_{NN1}, x_{NN2}, \dots\}$

2.2.1 Distance and similarity measurments

Given 2 point $\mathbf{q} = (q_1, q_2, \dots, q_n)$ and $\mathbf{p} = (p_1, p_2, \dots, p_n)$. n is the number of dimensions.



Euclidean Distance

$$d(\mathbf{q},\mathbf{p}) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)}$$

Manhatten Distance

$$d(\mathbf{q},\mathbf{p}) = ||\mathbf{q}-\mathbf{p}||_1 = \sum_{i=1}^{n}|q_i - p_i|$$

2.3 Finding k

$k_{opt} \in \{1, 2, \ldots, N\}$ N = # of training samples
Extreme cases:

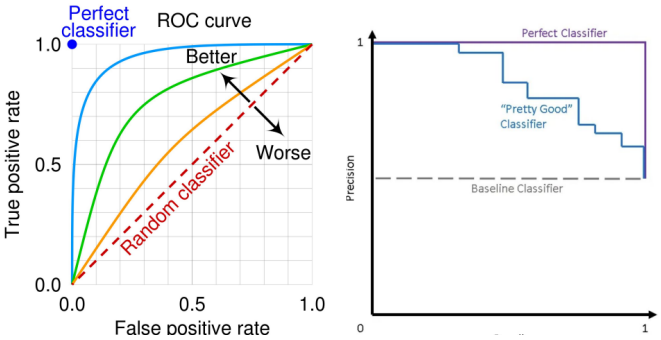
- $k = 1$: kNN = NN
- $k = N$: Majority class

2.4 Performance measures

- Accuracy: #corr / #all = (TP + TN) / (TP + FN + FP + TN)
- Error: #wrong / #all = 1 - Accuracy
- Recall,Sensitivity: TP /(TP + FN), How many relevant samples are correctly detected
- Specificity: TN/(TN + FP)
- Precision: TP/(TP + FP), How many detected samples are relevant
- F1 score: 2 · Precision · Recall / (Precision + Recall)

2.4.1 ROC (Receiver Operating Characteristic) curve

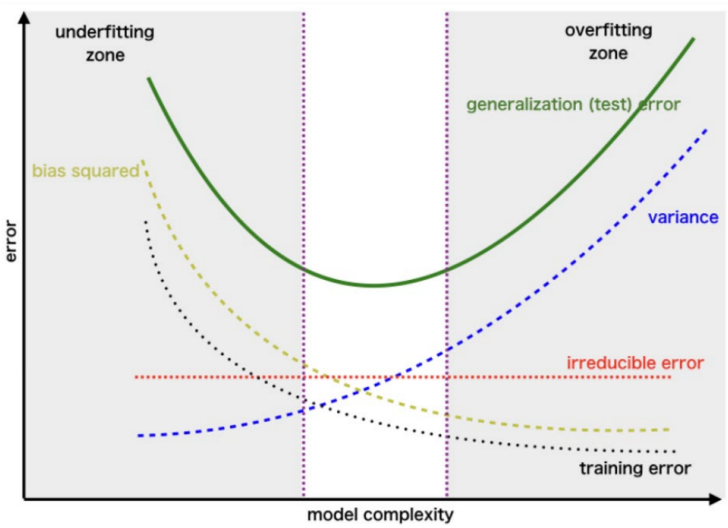
TPR, FPR, to find best threshold.



2.4.2 PRC(Precision-Recall Curve)

Precision, Recall, to find best threshold.

3 Bias-Varinace tradeoff



3.1 No free lunch Theorem

There is no universally best learner (across problems):

3.2 Ockham's Razor

Given 2 models with the same empirical (test) error, the simpler one should be preferred because simplicity is desirable in itself.

3.3 Error sources and bias-variance tradeoff

Different Error sources:

- The model: the best hypothesis is at distance to the true function
- The dataset: different datasets potentially provide different information
- Uncertainty in (X,Y) and its representation:

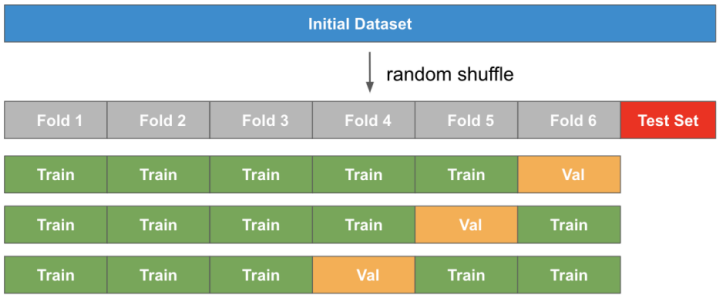
  Partial view of the task: have all relevant features been observed?

  Noisy data

Error Decomposit MSE:

$$E_{MSE} = bias + variance + Irreducible\,error$$

Bias(systematic error): average predictions deviation from the truth
Variance(dependence on specific sample): Sensitivity of prediction to specific training sample
Irreducible error(random nature of process): due to noise
Generally, for more complex/capable model: bias ↓,variance ↑.
Its a trade-off: Only way to redue both is to increase the size of the dataset.



$$\text{Error\_Estimate} = \frac{1}{K}\sum_1^K val\_error_{foldK}$$

3.4 Model selection: Validation score and CV score

k-Fold Corss Validation(CV): We can get a more realistic estimate of the test error using many validation sets (Typically K is 5 to 10)

3.4.1 Error Estimate Summary

In practice:

- Validation score(s) or CV score provide estimates of the test error
- The test error provides an estimate of the true error
- Never use any test data in the model training and model selection process

Which model to choose?

- The one with best validation or CV score
- Use student's t-test to check that an improvement is significant
- Ockham's razor: prefer simpler models in abscence of other evidence

Model selection is an empirical science.

3.5 Loss minimization (gradient descent)

Linear Model: $f(x) = w_0 + w_1 x = \hat{y}$

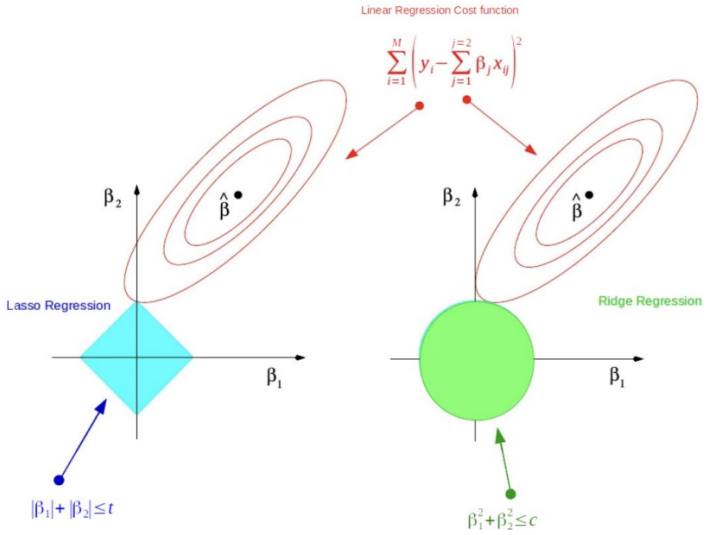
$$RSS = \sum_{i=1}^{N}(y_i - \hat{y_i})^2$$

Use gradient of RSS to find optimal weights for model.

$$\mathbf{w}_{new} = \mathbf{w}_{old} - stepsize \cdot \frac{df(\mathbf{w})}{d\mathbf{w}}$$

The effectiveness of gradient descent depends on the choice of the learning rate (step size):

- Too big, might not reach optimal value
- Too small, it will take a long time to converge



3.6 Regularization

Reduce overfitting of model by penalizing model complexity Tradeoff:

- increase bias
- decrease variance

$$H^* = \arg\min_{H\in\mathcal{H}} \sum_{i=1}^{n} \mathcal{L}_\theta(x_i, y_i) + \lambda R(H)$$

$R(H)$ is the models complexity. λ controlls the model complexity.

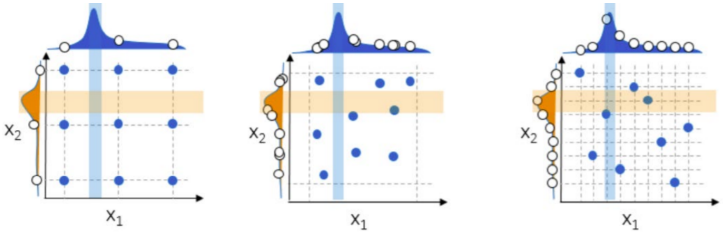
- Lasso:L_1 regularization

$$\arg\min \underbrace{||y - X\beta||_2^2}_{\text{Loss}} + \lambda \underbrace{||\beta||_2^2}_{\text{Penalty}}$$

- Ridge:L_2 regularization

$$\arg\min \underbrace{||y - X\beta||_2^2}_{\text{Loss}} + \lambda \underbrace{||\beta||_1}_{\text{Penalty}}$$

3.7 Hyperparameter tuning



- Grid-search
- Random-search
- {Optimization}

4 Decision Tree, Ensamble Methods & Random Forest

4.1 Decision Tree

- A flow-chart-like tree structure
- Internal node denotes a test on a attribute
- Branch represent an outcome of the test
- Leaf nodes represent class labels or class distribution

Gini Index:

$$I_G = 1 - \sum_{j=1}^{c} p_j^2$$

Entropy:

$$I_H = -\sum_{j=1}^{c} p_j \log(p_j)$$

p_j is the proportion of samples that belongs to calss j. Tree construction:

1. Initialization: whole region R_0(i.e., all given data)
2. Repeat:
    - For each region R_i, for each feature X_j, for each split $R_i = R_{i,l} \cup R_{i,r}$ with respect to feature x_j. Calculate change in impurity score (e.g., gini, entropy, error)
    - Choose best split,i.e., maximum decrease of the impurity score
    - Replace R_i with the two new split regions

Avoiding overfitting:

- Select a proper depth of the tree
- Select a proper minimum number of samples in a leaf to stop further splitting
- Tree pruning - remove split nodes bottom up or top down

All these performed using either a validation set or cross validation!

4.2 Pros/Cons

Pros:

- Easily visualized and interpreted
- No feature normalization or scaling needed
- Works well with mixed feature data types (categorical, continuous)

Cons:

- Easly ovefits
- Not robust, high variance

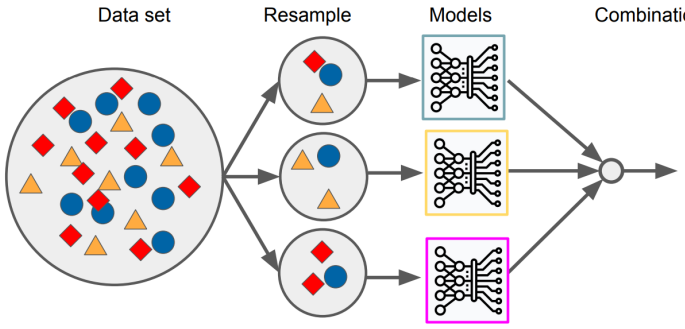
4.3 Ensamble methods

Appproach:

- Suppose you have n classifier
- Each classifier has error rate e
- Assume the classifier are independent
- Take Majority vote

The combined result is wrong if $n/2$ classifiers are wrong. According to central limit theorem variance reduces by factor n.

4.3.1 Bagging



To have independent classifier use many independent training sets S_i to train models.

- Variance reduces linearly (sub-linearly in practice beacuse S_i are correlated)
- Bias unchanged (increases slightly in practice)

4.3.2 Boosting

Weight samples. Samples where the model makes mistakes are weighted higher.
Algorithm:

1. Initialization: Train first model on data
2. Repeat:
    - Compute error of the model on each training sample
    - Give higher importance to samples where the model makes mistakes
    - Train next model using importance weighted training samples

In each iteration, introduce a weak model to compensate the shortcoming of the existing string (= combined) model.

Adaptive Boosting(AdaBoost)

Take a weak learning algorithm and turn it into a strong one by making it focus more on the accurate predictions of difficult cases.

$$F_T(x) = \sum_{t=1}^{T} f_t(x)$$

In each iteration t:

- A new weak classifier $h(x_i)$ with a coefficant α is added to the existing ones such that the error E_t of the ensemble at iteration t is minimized.

$$E_t = \sum_i [F_{t-1}(x_i) + \alpha_t h(x_i)]$$

- The new weak classifier is training using a weighted training set where the weight assigned to each sample is identical to the error of the current ensemble classifier on that sample $E(F_{t-1}(x_i))$.

4.3.3 Comparison

No Ensamble:

- complete training set, train one model

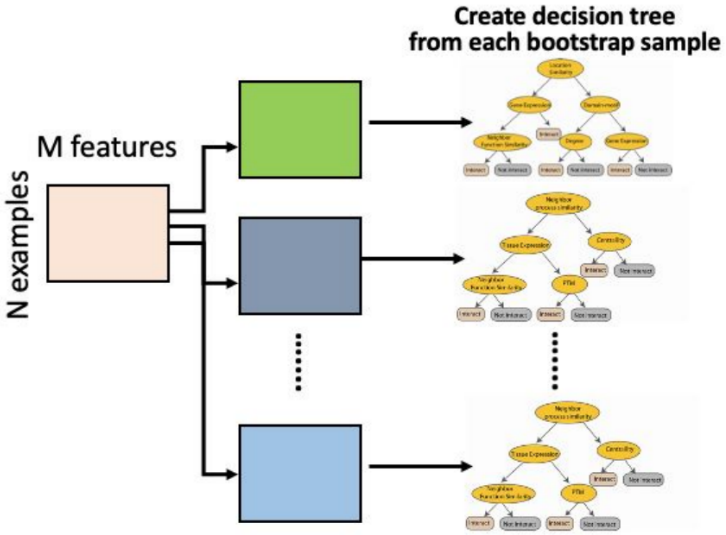
Bagging:

- randomly sample with replacement to obtain different training set
- minimizes variance (usually cannot reduce bias) – fights overfitting
- Computationally efficient (all models can be trained in parallel)

Boosting:

- randomly sample with replacement over weighted data to obtain different trainin sets
- Minimize bias by adding models to the ensemble – fight underfitting
- Address variance by using simple models with low variance

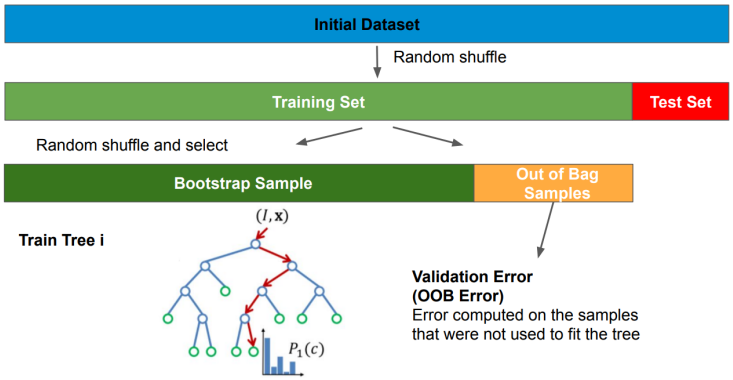
4.3.4 Random forest



Basic idea:

- Grow many trees in bootstrapped samples of training data
- Minimize bias by growing trees sufficiently deep (overfitting)
- Maximize variance reduction by minimizing correlation between trees by means of bootstrapping data for each tree and sampling variable set at each node
- Reduce variance of noisy but unbiased trees by averaging

Out of Bag Errors (OOB Error)



4.3.5 Summary Random Forrest

Pros:

- Simple - no assumption of the underlying distribution
- OOB error for free
- Many variables, even when they are not relevant for the task at hand or noisy
- Robust against outliers
- Multiclass
- Limit overfitting (trees have to be independent!)
- Unbalanced dataset (supsampling)

5 Probability Recap, Loss Functions, Logistic Regression, Neural Networks Intro

5.1 Probability Basics

Random Variable (RV) x denotes a quatity that is uncertain, discret or continuous. $p(x = \mathbb{X})$: the probability of variable x being in state $\mathbb{X}$.
Domain of RV denotes all the values it can take (states it can be in). $\text{dom}(coin) = \{\text{heads,tails}\}$
Joint distribution of two RVs x and y takes a particular combination of values and the joint probability density satisfies:

$$\int\int Pr(x, y) \cdot dxdy = 1$$

Marginal distributions $Pr(x)$ and $Pr(y)$ are obtained by:

$$\int Pr(x, y) \cdot dx = Pr(y)$$

$$\int Pr(x, y) \cdot dy = Pr(x)$$

Conditional probability $Pr(x|y)$ is the probability of a variable x taking a certain value assuming we know the value of y:

$$Pr(x|y) = \frac{Pr(x,y)}{Pr(y)}$$

5.1.1 probability rules

Sum rule:

$$p(X) = \sum_{Y} p(X,Y)$$

Product rule:

$$p(X,Y) = p(Y|X) \cdot p(X) = p(X|Y) \cdot p(Y)$$

Bayes Theorem:

$$p(Y|X) = \frac{p(X|Y) \cdot p(Y)}{p(X)} = \frac{p(X|Y) \cdot p(Y)}{\sum_{y \in Y} p(X,y)} = \frac{p(X|Y) \cdot p(Y)}{\sum_{y} p(X,y) \cdot p(y)}$$

Probability distributions and PDFs

$$\mathbb{E}[x] = \mathbb{E}_{x \sim p}[x] = \int x \cdot p(x)\, dx = \mu$$

$$\mathbb{E}[x^2] = \mathbb{E}_{x \sim p}[x^2] = \int x^2 \cdot p(x)\, dx = \mu^2 + \sigma^2$$

$$\mathbf{VAR}[x] = \mathbb{E}[x^2] - \mathbb{E}[x] = \sigma^2$$

5.2 Designing Loss Functions

Loss/cost function measures how bad the model is - the lower the value of the loss function the better the model maps inputs to output $\hat{\phi} = \arg\min [\mathrm{L}[\phi]]$.
Model training is finding parameter values that minimize the loss.
The negative log likelihood (to be minimized) gives us a loss function.

$$\hat{\phi} = \arg\min_{\phi} \left[-\sum_{i=1}^{I} \log [Pr(y_i|f(x_i,\phi))] \right]$$

5.3 Logistic Regression

Binary classification. Output is the probability that the input belongs to a class.

$$y = \text{logistic}(\mathbf{w}^T\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$$

5.3.1 Loss function (Binary Cross Entropy Loss)

$$J(\theta) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log \hat{y}_i + (1-y_i) \log(1-\hat{y}_i)]$$

Leads to the following update rule:

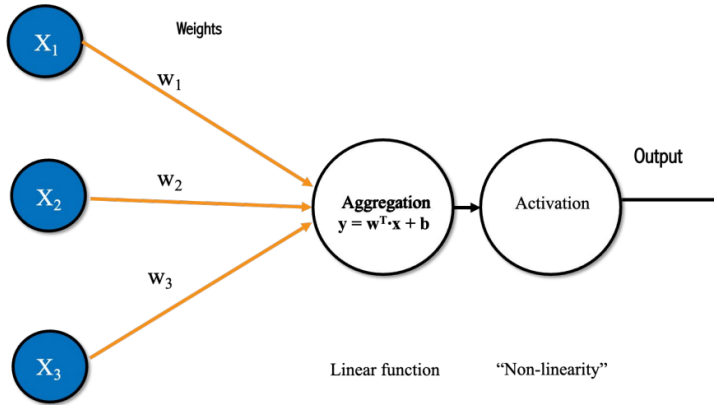
$$\theta_j = \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}$$

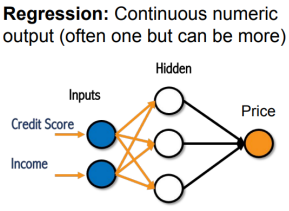
this is simplified to:

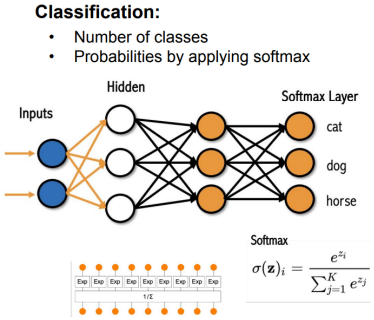
$$\theta_j = \theta_j - \alpha \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i) x_i^{(j)}$$

5.4 Neural Networks Basics

Neural Networks (NN) consist of Neurons. The value of a Neuron is defined by its connections to tha last layer, the weights(w_i) of the connection, the bias(b) and the activation funtion of the Neuron.







5.4.1 Number of parameters

Every connection(weights) + every Neuron(bias) = # of parameters

6 Neural Networks

Neural Networks are functions $y = f(x,\theta)$ with parameters θ that map mulitvariate inputs x to mulitvariate outputs y.
Unversal approximation theorem: A shallow neural network (MLP) using nonlinear activation function can approximate any given continuous function defined ib a compact subset of R^D to arbitrary precision given enough hidden units (finite number).

6.1 Loss function

Regression: Mean squared error loss

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

Classification: Cross entropy loss

$$Loss = -\sum_{i=1}^{\#\text{output classes}} y_i \cdot \log(\hat{y}_i)$$

6.2 Training

Repeat:

- Choose a training sample
- Forward pass: Compute the prediction
- Backward pass: If error > 0, update weights

Adjust weigths by Gradient descent

$$w_i = w_i - \alpha \frac{\partial J(w_i)}{\partial w_i}$$

Training Terms:

- Epoch = a forward pass and backward pass complete for all the training examples
- Batch size = the number of training samples in a Batch
- Iteration = forward and backward pass each using a Batch
- Iterations per Epoch = #training data /size of Batch

Avoide overfitting with:

- Regularization (Lasso, Ridge)
- Dropout, turn off some neurons during training
- Batch normalization
- Early stopping

6.3 Convolutional Neural Networks (CNN)

Idea: nearby pixels in an image are correlated - using shared parameters across whole input.

6.3.1 CNN 1D

1D convolution is a weighted sum of nearby inputs. A convolution operation is determined:

- stride (kernel shift)
- kernel size (typically odd)
- dilation (number of zero weights in kernel)

Convolutional Layer: apply convolution, add bias, apply activation function

7 Feature Engineering

8 Support Vector Machines

9 Gaussian Processes

10 Dimensionality Reduction

11 Cluster Analysis

12 Gaussian Mixture Models and EM

13 Reinforcement Learning

14 Generative AI