Learning types

Supervised Learning: Using annotated(labelled) data. Input Matrix $X = M(Nr\ of\ training\ samples) \times N(Nr\ of\ features)$. Output Vector $y = M$. Goal is to find function $f(X_{m,:}) \rightarrow y^{(m)}$

Unsupervised Learning: No labelled data. Goal is to identify and explain patterns and apply the model to new data => Fuzzy problem state

Clustering: Grouping objects in clusters with similar other objects

Dimensionality Reduction: Transform data without losing information

Reinforcement Learning: Agent observes environment, performs action, gets reward/punishment, adjust policy(strategy) by itself

Linear Regression

Hypothesis: $h_\theta = \hat{y}^{(m)} = \theta_0 + \theta_1 x^{(m)}$ **Residual**: $\epsilon^m = y^{(m)} - \hat{y}^{(m)}$

Loss: $LRSS = \sum_{m=1}^{M}\left(y^{(m)} - \hat{y}^{(m)}\right)^2 \approx \sum_{m=1}^{M}(y^{(m)} - (\theta_0 + \theta_1 x^{(m)}))^2$

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

Rooted Mean Squared Dev: $\sqrt{MSE}$ **MAE**: $\frac{1}{M}\sum_{m=1}^{M}|y^{(m)} - \hat{y}^{(m)}|$

Coefficient of Determination: Explains the fraction of variance explained by the model. $R^2 = 1 - \frac{SS_{res}}{SS_{tot}}; SS_{res} = \sum(y_i - \hat{y}_i)^2; SS_{tot} = \sum\left(y_i - \mu_y\right)^2$

Assumptions: Linearity (I/O have linear rel.), Independence (residuals/samples are independent), Normality (larger deviations from mean are less likely), Homoscedasticity (Error dist. is the same for all inputs,~~fan~~)

sample means

$$\mu_y = \frac{1}{M}\sum_{m=1}^{M} y^{(m)}$$

$$\mu_x = \frac{1}{M}\sum_{m=1}^{M} x^{(m)}$$

Closed Form Equations

$$\theta_0 = \mu_y - \theta_1 \mu_x$$

$$\theta_1 = \frac{\sum_{m=1}^{M}(y^{(m)} - \mu_y)(x^{(m)} - \mu_x)}{\sum_{m=1}^{M}(x^{(m)} - \mu_x)^2} = \frac{\tilde{s}_{xy}}{\tilde{s}_x^2}$$

Multi: $\hat{y}^{(m)} = h_\theta(x^{(m)}) = \theta_0 x_0^{(m)} + \theta_1 x_1^{(m)} + \theta_2 x_2^{(m)} + \ldots + \theta_N x_N^{(m)} = \theta^T X_{m,:}$

Dimensions: $X: M \times (N+1), \theta: (N+1) \times 1, y: M \times 1$

Polynomial Regression

Hypo: $h_\theta(z) = \theta^T z = \sum_{j=0}^{n} \theta_j z_j$ **Pred**: $h_\theta(z^{(m)})$

Regularization: $J(\theta) = \frac{1}{2M}\left[costFn + \lambda\sum_{j=1}^{n}\theta_j^2\right]$ Ensure θ is not too big

Hyperparameter: λ, can be chosen, **tuning**(search) to find "best" one

Logistic Regression

Hypo: $\hat{y} = h_\theta(x) = g(\theta_0 + \theta_1 x) = 1/(1 + e^{-(\theta_0 + \theta_1 x)})$

Log-Loss: $(h_\theta(x), y) = -y\log\left(h_\theta(x)\right) - (1-y)\log\left(1 - h_\theta(x)\right)$

Logistic regression (two classes)

$$P(y=1|x) = \sigma(z) = \frac{1}{1+e^{-z}}$$

where $z = w^{\mathrm{T}}x$

$$P(y=1|x) = \frac{1}{1+e^{-(w^T x)}}$$

$$P(y=0|x) = 1 - P(y=1|x) = \frac{e^{-(w^T x)}}{1+e^{-(w^T x)}}$$

Definitions

Features: Individual measurable properties: Independent Variables, Predictors, Attributes or Covariates

Classification: Predict a discrete category or class label, fixed nr of output

Regression: Predict a continuous numerical value or quantity, find trend

Instance based: Pred based on data

Normal Equation: For Linear Regress

$$\begin{pmatrix}\varepsilon^{(1)}\\ \varepsilon^{(2)}\\ \vdots\\ \varepsilon^{(M)}\end{pmatrix} = \begin{pmatrix}y^{(1)}\\ y^{(2)}\\ \vdots\\ y^{(M)}\end{pmatrix} - \begin{pmatrix}1 & x^{(1)}\\ 1 & x^{(2)}\\ 1 & \vdots\\ 1 & x^{(M)}\end{pmatrix}\begin{pmatrix}\theta_0\\ \theta_1\end{pmatrix}$$

^ $\epsilon = y - X\theta \Rightarrow \theta = (X^T X)^{-1} X^T y$

Residual Plot: Scatter of pred vs res

Standard res: res/st.dev(res)

$$\bar{e} = \frac{1}{n}\sum_{i=1}^{n} e_i \quad s_e = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(e_i - \bar{e})^2}$$

Model Complexity: -Bias (Too many assumptions=>underfitting), Variance +(too much attention=>overfitting)

Performance Metrics

Accuracy: corr/all **Precision:** TP/(TP+FP)

Recall/TPR/Sens: TP/(TP+FN)

Confusion Matrix: TP&TN / PN&PP

F1:2 $\frac{prec*recall}{prec+recal} = \frac{2TP}{2TP+FP+FN}$ →1 good

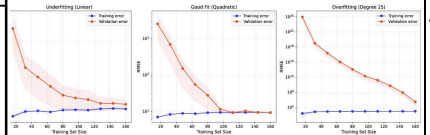
Multi: Calc for every class and average is macro. Micro prec: $\frac{\sum_{k=1}^{K} TP_k}{\sum_{k=1}^{K}(TP_k + FP_k)}$

K-Fold: Shuffle. Divide into k-folds. Hold out 1 k for test, train with rest. Repeat with all k. take average of all k errors

LOOCV: ^but k=M, good for small data

Stratified: Each fold needs same distro

Learning Curves: error=train, validation



Pipeline: Task_Def>Data_Coll+Label> Model Sel<>Eval> Deploy&Monitor

Data

Types: Categorical=> Nominal (Cannot be ordered, gender) & Ordinal (discrete and ordered), Numerical => Continues & Discrete (can be counted infinite)

Structured: Has data model / schema

Semi-Structured: contain tags/markers

Metadata: Data about data, supportive

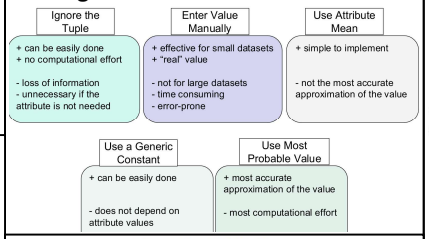
Cleaning: Deduplicate, Outlier detection

Cosine Similarity: How numerically close are two vectors, only looks at angle not length=>proportional similarity

$$cos\theta = \frac{A \cdot B}{|A||B|} = \frac{\sum_i^n A_i B_i}{\sqrt{\sum_i^n A_i^2}\sqrt{\sum_i^n B_i^2}}$$

Levensthein distance: min nr of edits to transform two words to same

Missing Values:



Probabilities

Sample Space: set of all possible outcomes of an experiment. Any subset of the sample space is called an **event** => outcome, or a set of outcomes.

Probability: 0-1 of how likely

Random Variable (RV): num. quantity from the outcome of random process

Axioms: 1. non-negativity 2. Certainty (sum of all=1) 3.Additivity (for mutually exclusive, disjoint events)

Joint: A and B / intersection = *

Conditional: that B occurs given that A occurs P(B|A) = P(A∩B) / P(A)

Law of total probability: Calc chance of each unique scenario, add them up weighted by how likely they are

Gradient Descent

Repeat: $\theta_j = \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta)$. For 0 no x

Batch: $\frac{1}{M}\sum_{m=1}^{M}\left(h_\theta(x^{(m)}) - y^{(m)}\right)x^{(m)}$

SGD: Shuffle data, for i=1-M, for j=1-N

$$\theta_j = \theta_j - \alpha\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

Learning Rate: Needs to be in the middle, dynamic. **Decay**: $\alpha_t = \frac{1}{1+decay*t}\alpha_0$

Mini batch: SGD but batch $1 < b < M$

Neural Networks ($z_j = w_j^T x + w_{0j}$)

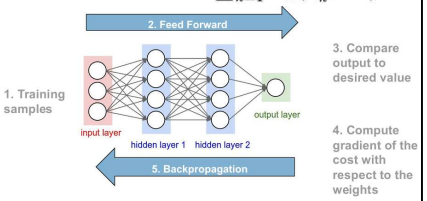
Softmax Reg: $P(y=k|x) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$

$$L_{CE}(\hat{y}^{(m)}, y^{(m)}) = -\sum_{k=1}^{K} y_k^{(m)} \log\hat{y}_k^{(m)}$$

$$= -\log\hat{y}_c^{(m)} \quad (c \text{ is the correct class})$$

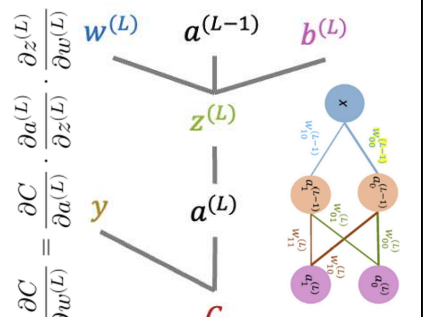
$$= -\log p(y_c^{(m)} = 1|x^{(m)})$$

$$= -\log\frac{\exp(w_c^T x^{(m)})}{\sum_{k=1}^{K}\exp(w_k^T x^{(m)})}$$



Dropout: drop multiple random nodes during each training round

Backpropagation: Los: out*(1-out)



$$\frac{\partial C}{\partial w_{00}^{(L-1)}} = \frac{\partial C}{\partial a_0^{(L)}}\frac{\partial a_0^{(L)}}{\partial z_0^{(L)}}\frac{\partial z_0^{(L)}}{\partial a_0^{(L-1)}}\frac{\partial a_0^{(L-1)}}{\partial z_0^{(L-1)}}\frac{\partial z_0^{(L-1)}}{\partial w_{00}^{(L-1)}} +$$

$$\frac{\partial C}{\partial a_1^{(L)}}\frac{\partial a_1^{(L)}}{\partial z_1^{(L)}}\frac{\partial z_1^{(L)}}{\partial a_0^{(L-1)}}\frac{\partial a_0^{(L-1)}}{\partial z_0^{(L-1)}}\frac{\partial z_0^{(L-1)}}{\partial w_{00}^{(L-1)}}$$

Support Vector Machines

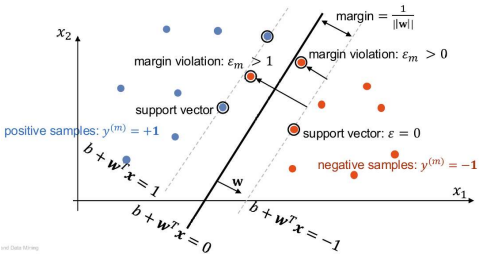
Dimensions: Features **Class Labels**: −1 and +1
Separate with N-1 dimensional hyperplane $b + w_1x_1 + \cdots + w_Nx_N = 0$, take sign for label
Margin: distance of the hyperplane to the closest training samples (gutters)
Support Vectors: Datapoints closest to HP
Distance Point to HP: $|w|| = \sqrt{w_1^2 + \ldots + w_n^2}$

$$r^{(m)} = y^{(m)}\left(\frac{1}{||w||}(w^T x^{(m)} + b)\right)$$

Minimise $\frac{1}{2}||\boldsymbol{w}||^2$ subject to $y^{(m)}(b + \boldsymbol{w}^T\boldsymbol{x}^{(m)}) \geq 1$ for $m = 1, \ldots, M$

Normalization: $x' = \frac{x-\bar{x}}{s_x} s_x = \sqrt{\frac{\sum_{i=1}^{N}(x_i-\bar{x})^2}{N(-1)}}$

Soft margin: Max $+C\sum_{m=1}^{M}\epsilon_m$; $to\ 1 - \epsilon_m$
C is hyperparam >=0, controls how many points violate plane(-C=more)=>bias-variance?



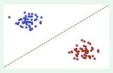
Kernel-Trick:

- **Linear** kernel: $K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \sum_{n=1}^{N} x_n^{(i)} x_n^{(j)}$
  - this yields the standard support vector classifier
- **Polynomial**: $K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \left(c + \mathbf{x}^{(i)T}\mathbf{x}^{(j)}\right)^d = \left(c + \sum_{n=1}^{N} x_n^{(i)} x_n^{(j)}\right)^d$
  - Parameters: c and degree d
  - with $d = 1$ and $c = 0 \rightarrow$ linear kernel
- **Radial basis (RBF)**:

$$K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp\left(-\gamma||\mathbf{x}^{(j)} - \mathbf{x}^{(i)}|\right|^2) = \exp\left(-\gamma \sum_{n=1}^{N}\left(x_n^{(j)} - x_n^{(i)}\right)^2\right)$$

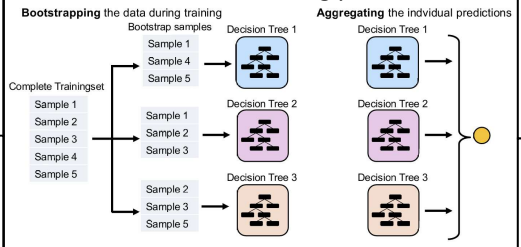
  - Parameter γ

Models Comparison

Type model	+++	---
Generative 	Can generate new data samples	More computationally expensive
	Better understanding of the data	Assumptions of features independence
	Robust with limited data	
Discriminative 	More accurate for classification	Cannot generate data samples
	Weak assumptions for features independence	Less interpretable
	Better for high-dimensional data	Features-dependent performance

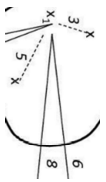
Decision Trees

Definitions: Leaf(end), samples(training instances),value(samples in each class), impure(mix of classes)
Gini: $G(Q_i) = 1 - \sum_{k=1}^{K} p_{i,k}^2$; $p_{ik} = \frac{1}{M_i}\sum I(y = k)$
How often is random(according to node distro) element incorrectly labelled, max = 0.5
Weighted Gini: Weight of childs = goingTo/all. Then multiply weight*Gini for each child and add
Information Gain: Gini(parent)-Weighted Gini^
Entropy: $H(Q_i) = -\sum_{k=1}^{K} p_{ik}\log_2 p_{ik}$; $max = 1$
Algorithm: Separate to maximize information Gain, all possible splits, do again for every split
Regression Trees: Predict the average numerical value over samples that land in the node, training with CART but for training use MSE
Regularization: Pre-Pruning(Min-Sample Pruning: Only split nodes with at least k samples, Max-Depth Pruning: Restrict the maximum depths of the tree), Post-Pruning (Use validation data to decide for each leaf whether it is "reasonable")
Random Forests: Train different diverse! trees by bagging and average decision (Ensemble). For classification hard(majority) or soft(class with the highest cumulated probability) voting
Bagging: with replacement(pasting without) **Out-of-bag error**: Samples can be left out, rate these elements that were after wrongly classified



$$a(x_1) = \frac{3+5}{2} = 4 \qquad b(x_1) = \min\left(\frac{6+8}{2}, \frac{10+12}{2}\right) = 7$$

$$Sil(x_1) = \frac{7-4}{7} = \frac{3}{7} = 0.42$$



Bayes' Theorem

Bayes Theorem: $P(A|B) = \frac{P(B|A)\times P(A)}{P(B)}$ Likelihood × Prior / Evidence

disease test that is 99% accurate, 1% of population has disease then. P(Disease)=0.01 ; P(Pos|Disease)=0.99 so

$$P(\text{Disease}|\text{Positive}) = \frac{P(\text{Positive}|\text{Disease})\cdot P(\text{Disease})}{P(\text{Positive})}$$

P(Positive)=P(Pos|Disease)·P(Disease) +P(Pos|No Disease)·P(No Disease)

Generative Models

Learn the joint probably for each class P(X,Y) Of observing class y for features X. Use Bayers to calc P(Y|X). **Discriminative** models learn the conditional probability distribution $(Y|X)$ directly, generative know how data is placed throughout the space. **Inference**: Gen(Highest prob), Disc(decision boundary) **Naïve Bayes**: Assumption all features are conditionally independent => Bernoulli, Multinomial, Gaussian. Works even if not true sometimes

Probability Distribution

Normal	$f(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$	μ (mean), σ (standard deviation) $\mu \in \mathbb{R}$, $\sigma > 0$
Bernoulli	$P(X = x;p) = p^x(1-p)^{1-x}$ for $x \in \{0,1\}$	p (probability of success) $0 \leq p \leq 1$
Binomial	$P(X = k;n,p) = \binom{n}{k}p^k(1-p)^{n-k}$ for $k = 0,1,\ldots,n$	n (number of trials), p (probability of success per trial) $n \in \mathbb{N}$, $0 \leq p \leq 1$
Beta	$f(x;\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}$ for $0 \leq x \leq 1$	α,β (shape parameters) $\alpha > 0$, $\beta > 0$
Uniform	$f(x;a,b) = \frac{1}{b-a}$ for $a \leq x \leq b$	a,b (minimum and maximum values) $a < b$

Maximum Likelihood Estimation (MLE): Find params to maximize likelihood function. **negative log-likelihood** easier to calc, instead of sum – [log(p)+…] => it becomes a minimization problem

Maximum A-Posteriori Estimation: $\theta_{MAP} = argmax_{\ \theta}\frac{P(X|\theta)*P(\theta)}{P(X)} = P(\theta|X)$ where $\boldsymbol{P(X|\theta)}$ is the likelihood, $\boldsymbol{P(X)}$ is the evidence, $\boldsymbol{P(\theta)}$ is the prior probability

DBSCANoise Algo

Input: Dataset, MinPts, ε **1**. Select an unprocessed point P(mark after) **2.** If P is not a Core Point? Classify as noise and go back to 1 **3**. Start new cluster and add P + all neighbours within ε of it. **4.** For each neighbour that is a core point, recursively add all its neighbours to the cluster, repeat until no more can be added, then go back to 1

Clustering (Unsupervised)

Goals: Understanding(natural), Identification, reduction(find repr.) and outlier detection **Soft**(Probability for cluster) or **Hard**(unique cluster)
Similarity: Jaccard Distance (Intersect/Union), Euclidean **||2** $(d(p,q)^2 = \sum(q_i - p_i)^2)$, Manhattan (straight line, $d_1(p,q) = ||p-q||_1 = \sum|p_i - q_i|$)

DBSCAN

Definitions: minPts (min nr of points (a threshold) clustered together for a region to be considered dense), ε (distance, (x,y) < ε), Core point (has at least minPts points within distance ε from it), Border point(has at least one core point within distance ε), Noise point(is neither) **Properties**: Runnintime of O(n^2) or even O(nlogn) for indexed data. No need to specify k. arbitrarily shape,noise

K-Means

1. Init (Random, Forgy(random from training), Partition(Randomly assign a cluster to each point)k centroids 2. Assign each datapoint to closest k $f^{(t)}(x_i) = argmin\, j\left\|x_i - c_j^{(t)}\right\|^2$ 3. Recalculate centroids as the mean of all points assigned to that cluster=>2 4. Convergence with time complexity O(iter*k*dataPoints*dim)
WCSS: $J(C,f) = \sum_{i=1}^{n}\left\|x_i - c_{f(x_i)}\right\|^2$
Clusters: Init has big impact. **Elbow** Run with multiple k values and graph, select k where elbow point is. **Silhouette** Cohesion a is average distance between points in same cluster, Separation b is average distance between point and all points in nearest cluster. $s = \frac{b-a}{\max(a,b)}$. High+