DD2421 Machine Learning

K-nearest neighbor

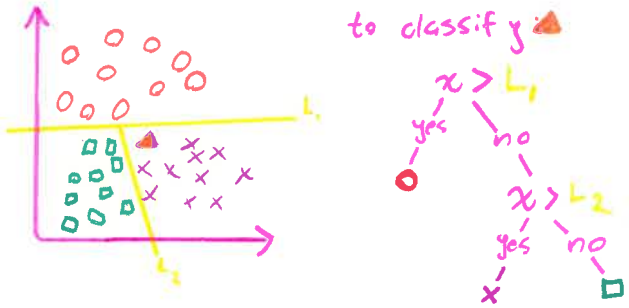
Presented with an unlabeled sample its predicted label will be the same as the majority of the k nearest neighbor.

Δ will be labeled as x

If k increases the decision boundary becomes smoother, it also generalizes better.
Memory & computational requirements scale linearely with training size.
Lower k higher overfitting

Decision Tree

Create logical questions that partitions the search space into classes.



train by finding best questions that maximize information gain, and splitting data sets accordingly.

Information gain is meeasured in entropy.

$$\text{Entropy} = \sum_i -p_i \log_2 p_i \quad \forall p$$

average over binary

Entropy is the number of questions needed to find an answer.
also number of bits of information

Alternative: Gini Impurity

$$\sum p_i(1-p_i) = 1 - \sum p_i^2$$

Expected error rate at node N if label from class distribution at N
Similar to entropy but peaks stronger at equal probabilities.

Idea:
Ask question that maximizes expected reduction of entropy

$$\text{Gain} = \underbrace{E_{\text{entropy}}(S)}_{\text{Before}} - \underbrace{\sum_{S'} \frac{|S'_i|}{|S|} \text{Entropy}(S'_i)}_{\text{After}}$$

high gain = good

weighted subset

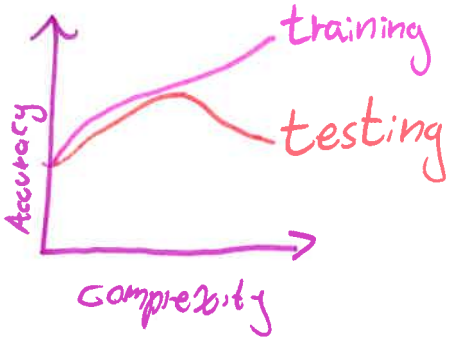
entropy of subset

Choose attribute that says the most about the answer.

Overfitting

The model performs too specialized on training data

↳ bad representation ↳ too complex model ↳ noisy examples

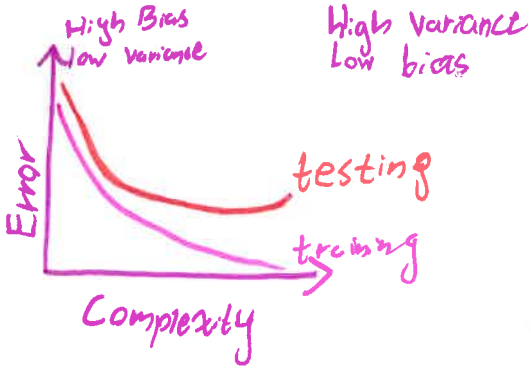


Three datasets, training, validation, testing

K-fold Validation:

split data into k folds

train k times using different validation set



Bias: Difference between average Prediction and correct value

Variance: Variability in model prediction between realizations of model

Regression

fit a function to a dataset

$$D = \{(\bar{x}_1, \bar{y}_1) \dots (\bar{x}_n, \bar{y})\}$$

Linear regression

$$\hat{f}(\bar{x}) \approx \overbrace{\sum w_i x_i}^{f(x)} = W^T \bar{x}$$

Mean squared error

$$\left(\hat{f}(\bar{x}) - f(x)\right)^2$$

$$E(\hat{f}) = \frac{1}{N} \sum \left(\hat{f}(\bar{x}_n) - f(\bar{x}_n)\right)^2$$

$$= \frac{1}{N} \|\bar{X}w - Y\|^2$$

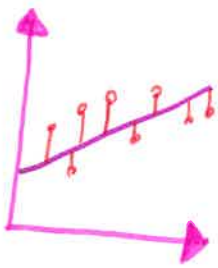
compute w that minimizes E

Residual Sum of Squares

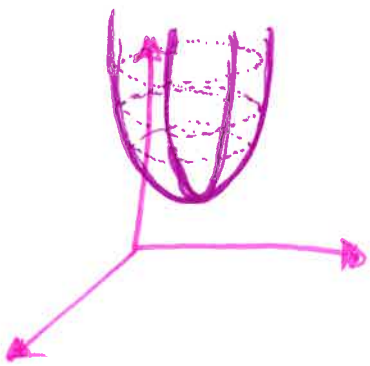
$$E_{in}(w) = \|Xw - Y\|^2$$

$$\frac{\partial}{\partial w} E_{in}(w) = 2X^t(Xw - Y)$$

when gradient is zero the error is minimized



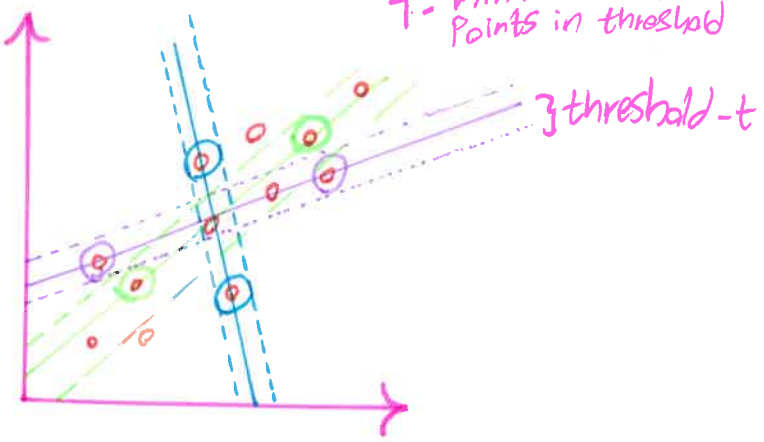
Least squares



Random Sampling Consensus

Ransac, more robust regression

1) sample 2 points
2) Draw line with a threshold
3) repeat from 1 until line is good enough.



2 hyperparameters:
distance threshold - t
subset threshold - T

Least Absolute Shrinkage & Selector Operator

Lasso Regularization aka l_1 norm. will force some some coefficient to Zero
This is added as a penalty to the Loss function

$\lambda \sum |w|$

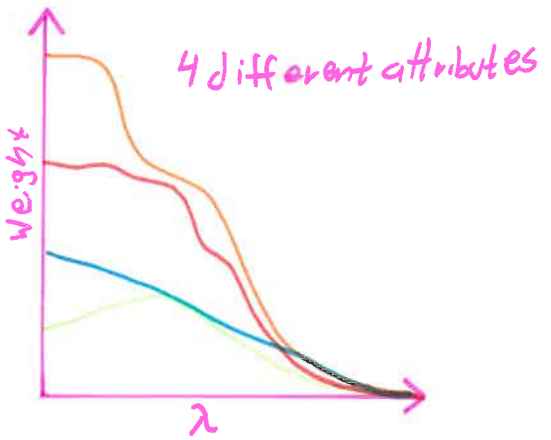
factor ↑ ↑ total weights

λ increases Bias while decreasing Variance

Ridge Regression

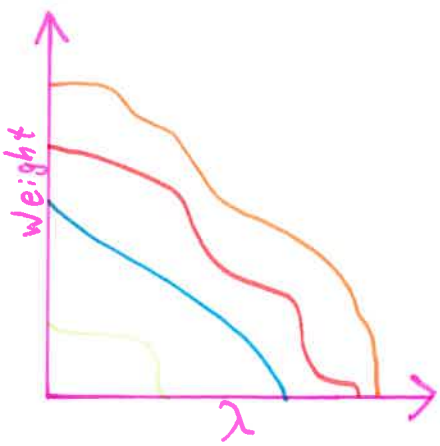
Similar to Lasso but squared

$\lambda \sum w^2$



All weights for the attributes tend to zero as $\lambda \to \infty$ but they are always above zero

Ridge Regression



All weights decrease and eventually reach zero. some do it before others

Lasso

Probabilistic Learning

axiom $P > 0$, $\sum P = 1$

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$ → if A and B are independent $P(A \cap B) = P(A)P(B)$ → $P(A|B) = P(A)$

Bayes Theorem

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

Posterior — Likelyhood — evidence — Prior

Probabilistic Regression

1) find Joint distribution of X and Y
   $P(x, y)$
2) compute posterior of Y
   $P(y|x)$
3) compute Expectation
   $E(y|x)$

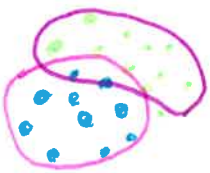
Probabilistic Classification

Same as regression, but use Argmax

Learning As Inferense

Discriminative & Generative models

- Discriminates classes
  - $P(y|X, D)$
  - Draw boundary
- tries to model underlying distribution
  - $P(x|y, D)$
  - Draw distribution



Parametric vs Nonparametric

$P_r(y|x) = P(y|x, \theta)$

- Estimate $\hat{\theta}$ using D
- compute $P(y|x\hat{\theta})$
- learning means finding $\hat{\theta}$

Estimate $P(\theta|D)$

Compute $P(y|x, D)$ from $P(y|x, \theta, D)\,P(\theta|D)$ by marginalizing out θ

Number of parameters grow with data

Fundemental Assumptions

- observations are independent and identically distributed

Thus: $D = \{o_1, o_2 \cdots o_N\}$ $o_i = (x_i, y_i)$

$$P(D) = \prod_i P(o_i)$$

Take log

$$\log(P(D)) = \sum_i \log(P(o_i))$$

Maximum Likelihood Estimate

Find parameters that make D most likely.

$$\Theta_{ML} = \underset{\Theta}{\operatorname{argmax}}\, P(D|\Theta) = \underset{\Theta}{\operatorname{argmax}}\, \log(P(D|\Theta))$$

This can be used to approximate the x or y

$$P(x|y,D) \approx P(x|y,\Theta_{ML}) \qquad P(y|x,D) \approx P(y|x,\Theta_{ML})$$

Naive Bayes Classifier

Assumes features are independent.

$$MAP_{NB} = \underset{y}{\operatorname{Argmax}}\, P(y) \cdot \prod_{D} P(x_d|y)$$

Generative model

$P(X|y)$

Logistic Regression

Treating a binary classification problem as regression problem.

Discriminative

$P(y|x)$

$$Y|X \sim \text{Bernoulli}(\lambda(x))$$

$$P(y|x) = \lambda(x)^y (1-\lambda(x))^{(1-y)}$$

$$\lambda(x) = \text{sigmoid}(w^T x)$$

Maximum A Posteriori

Choose most likely Θ given D

$$\Theta_{MAP} = \underset{\Theta}{\operatorname{argmax}}\, P(\Theta|D)$$

$$= \underset{\Theta}{\operatorname{argmax}}\, \frac{P(\Theta)P(D|\Theta)}{P(D)}$$

$$= \underset{\Theta}{\operatorname{argmax}}\, P(\Theta)P(D|\Theta)$$

$$P(\Theta) \cdot \prod_{N} P(x_n|\Theta)$$

$$\log(P(\Theta)) + \sum_{N} \log(P(x_n|\Theta))$$

works as a regularizer

Bayesian Estimation

1) consider Θ as a random variable
2) Characterize Θ with $P(\Theta|D)$
3) Compute new posterior $P(y|x,D)$
   Marginalize over Θ

$$P(y|x\,D) = \int_{\Theta\in\Theta} P(y|x\Theta)\,P(\Theta|D)\,d\Theta$$

ML	D	Θ_{ML}	$P(y|x,\Theta_{ML})$
MAP	$D, P(\Theta)$	Θ_{MAP}	$P(y|x,\Theta_{MAP})$
BAYES	$D, P(\Theta)$	$P(\Theta|D)$	$P(y|x,D)$

Clustering K-Means

1) Place k cluster centers in D
2) assign each $d \in D$ to closest cluster.
3) Move cluster centre to the mean position of all $d \in$ Cluster
4) repeat 2-3 untill mean difference for points is less than ε

Expectation Maximization

Mix of K-means and GMM.

1) for each sample introduce h_i
2) estimate which gaussian distributed model is most responsible for x_i
   $P(h|x)$ ← called responsibility

Place K random gaussian
for each point do $P(h|x)$
Update gaussians accordingly

Gaussian Mixture Models

distribution is a weighted sum of gaussian distributions.

$$P(x|\Theta) = \sum_{k\in K} \tilde{\pi}_k P(x|\Theta_k)$$

$$\Theta = \{\tilde{\pi}_1, \tilde{\pi}_2, \tilde{\pi}_3 \ldots \tilde{\pi}_K, \Theta_1, \Theta_2 \ldots \Theta_K\}$$

We cannot assume classes are independent

Linear Separation

Use a line to separate data into True and False. Train weights and bias to correctly separate classes. In high-dimensional spaces linear separation becomes easy.

Perceptron Learning

only change weights when input is wrong. will always find solution if it exists.

$$w_i = w_i + \eta(t-o)x_i$$

May give bad generalization since first viable solution is returned.

Risk Minimization

find Plane that has the largest margin to datapoints

$$w^T\vec{x} = 0$$ any point in input

$$t\vec{w}^T\vec{x} \geq M \rightarrow \text{margin}$$

t: +1 for positive target, -1 for negative target

Best Margin is when $\frac{1}{\|w\|}$ is maximized.

$\Rightarrow \|w\|$ is minimized

SUPPort Vector Machine

Project data into higher dimension

Projections use Kernel to pretent there is a projection while saving resources.

Kernel:

$$\phi(\vec{x})^T \cdot \phi(\vec{y}) = K(\vec{x}, \vec{y})$$

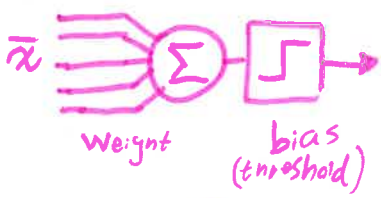
New equation

$$t\vec{w}^T\phi(x) \geq M$$

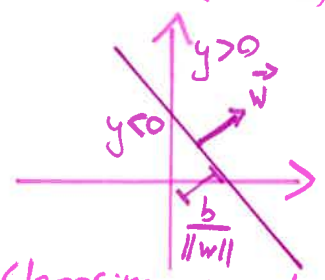
Slacks: introduce $C\sum\xi_i$

$$t\vec{w}^T\phi(x) \geq M + \xi_i$$

Artificial Neuron

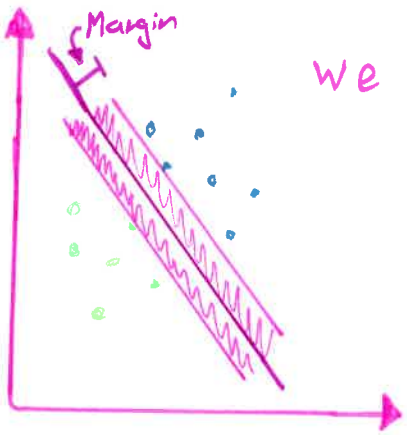


$$y = \text{sgn}\left(\sum x_i w_i - b\right)$$



Choosing w and b is "learning"

Always a straight line. Does not work on non-linearely separable data.



We want to maximize M

Lagrange multiplyer

$$L = \frac{1}{2}w^Tw - \sum_i \alpha_i[t_i w\phi - 1]$$

minimize L w.r.t w

$$\Rightarrow w = \sum_i \alpha_i t_i \phi(x_i)$$

$$\Rightarrow \sum_i \alpha_i - \frac{1}{2}\sum_{ij} \alpha_i \alpha_j t_i t_j \phi(x_i)\phi(x_j)$$

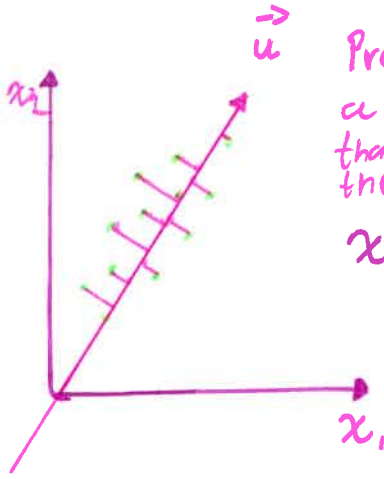
some will tend to zero; $\phi(x_i)\phi(x_j)$: kernel

Principal Component Analysis

Reduce the number of variables by only dealing with components with large variance.

Premice: you are given many factors which describe a dataset but some are correlated meaning they can be combined.

Projecting multiple dimentions onto a single vector $\vec{u}$ will reduce dimensions.



Projecting $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ onto a single vector $\vec{u}$ that will preserve the most information

$$x' = (x^T \cdot \vec{u})\vec{u}$$

$$x' = (\vec{x}^T \cdot \vec{u})\vec{u}$$

information perserverd

$(x^T \cdot u)^2$ ← maximize

$C\vec{u} = \lambda\vec{u}$

λ ← eigenvalue

$\vec{u}$ ← eigenvector

C ← covariance matrix

The eigenvalue determens the weight of the component.

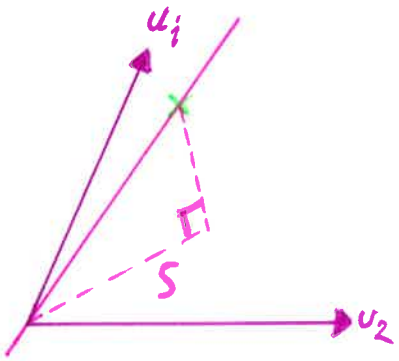
Alternative criterion: Minimum squared distance

$\|x - x'\|^2$

since C is symetric, all $\vec{u}$ will be orthogonal

Subspace Method

Project a class onto a subspace of the dataset. then classify new points based on how far they are projected on the subspace.



$$S = \sum_{i=1}^{p} (x, u_i)^2$$

p ← dimension of subspace

u_i ← basis

large projection ⇒ more similar

Fisher's Criterion

The eigenvector for the greatest eigenvalue of $S_w^{-1} S_B$ gives A that maximizes fishers criterion.

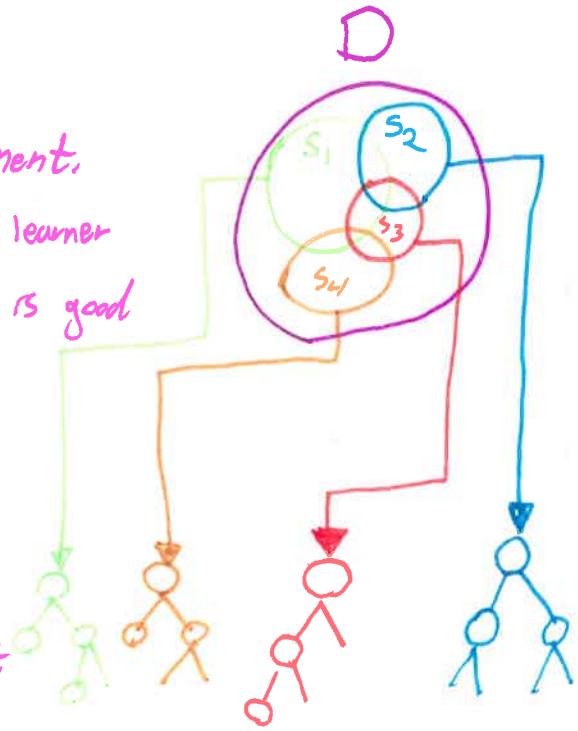
Ensemble Learning

Combine multiple weak learners.
Allows for each classifyer to have higher variance.
Assumes weak learners are independent

Diversity of opinion
Independence
Decentralization
Aggregation.

Bootstrap Aggregating

Bagging
Train on dataset by sampling with replacement.
Variance: High - Overtrain each individual learner
Bias: Low - On average classification is good



Random Forest

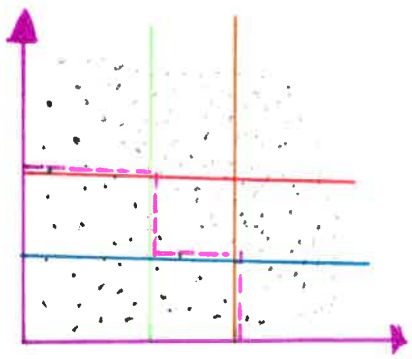
Bagging + feature selection randomization
Instead of selecting best feature to split at each node it is random so that trees are built different from each other.

Boosting

Treat misclassified samples as more important for each additional classifyer
Variance: low - each model is very simple
Bias: High - Each learner is bad but if you combine them they are strong

Voting: $f(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

α_t: Weight of each classifyer.
$h_t(x)$: individual weak learner



Four weak learners combine to one stronger one.

Adaboost:
Lab 3

$\alpha_t = \log_e\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$

$W_i = w_i \cdot \exp(-\alpha_t y_i h_t(x_i))$ ← normalize to 1

Dropout

Randomly kill a node in an ANN and force other nodes to improve