Course Contents

- 11 lectures
- 1-3 labs (mandatory)
- Programming challenge
- Written Exam

1. Nearest Neighbour Classifier (Memory-based)
2. Decision Trees (Logical inference)
3. Challenges in Machine Learning
4. Regression
5. Probabilistic Methods
6. Learning as Inference
7. Learning with Latent Variables
8. Support Vector Machines
9. Artificial Neural Networks
10. Ensemble Methods
11. Dimensionality Reduction
12. Mini lectures (beyond the scope of DD2421), exam Q&A

1. Decision Trees
2. Support Vector Machines
3. Bayes Classifier & Boosting

- labs are carried out by students and examined by TAs
- use Canvas to book time slots before the deadlines

Examination:

- It is your task to convince the examiner that you have done the assignment and understood the results.
- Strongly encouraged to work+report by pairs of 2 students (not 3!).
- 10 minutes, **be there on time**.
- No programming code to be shown
- Bring your ID (tell the TA if you are yet to be registered)

Classification

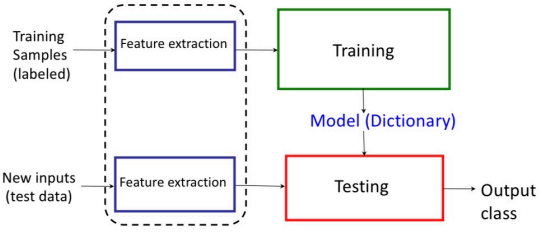
We would like to enable a computer to learn from data to answer a question "what is it?" You are given sample data (to find pattern)

The framework of classification

1) **Training Phase**: To give the concept of classes to a machine using labeled data

2.) **Testing Phase**: To determine the class of new unseen (unlabeled) data

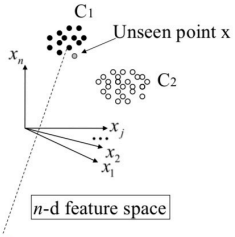
Schematic of classification



Nearest Neighbour Methods

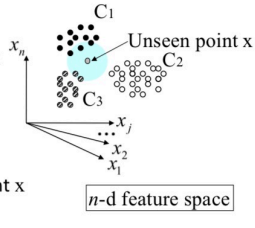
Nearest Neighbour methods

- Binary classification
  - N_1 samples of class C_1
  - N_2 samples of class C_2
  - Unseen data x
  - → Compute distances to all the $N_1 + N_2$ samples



- Find the nearest neighbour
  → classify x to the same class

- k-nearest neighbour rule
  - Compute the distances to all the samples from new data x
  - Pick k neighbours that are nearest to x
  - → Majority vote to classify point x (Nearest Neighbour is 1-NN)



Pros and cons of k-NN

- k-NN / 1-NN comparison summary
  - the boundary becomes smoother as k increases
  - lower computational cost for lower k
  - k-NN better generalizes given many samples
- Pros:
  - simple; only with a single parameter k
  - applicable to multi-class problems
  - good performance, effective in low dimension data
- Cons:
  - costly to compute distances to search for the nearest
  - memory requirement: must store all the training set

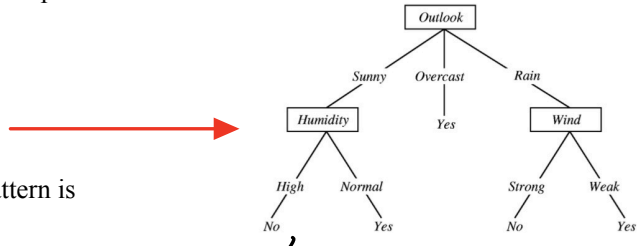
Lecture Contents

Decision trees
- The representation
- Training

Right now →

Basic idea: Test attributes (features) sequentially = Ask questions about the target/ status sequentially

Example: Building a concept of whether someone would like to play tennis



What does this tree encode?

Each leaf node bears a category label, and the test pattern is assigned the category of the leaf node reached

How do we grow the tree?

(Sunny ∧ Normal Humidity) ∨ (Cloudy) ∨ (Rainy ∧ Weak Wind)

1. Choose the best question, according to the information gain, and split the input data into subsets
2. Terminate: Call branches with a unique class labels leaves (no need for further questions)
3. Grow: Recursively extend other branches with subset bearing mixtures of labels

Unpredictability
- Entropy
- Information gain
- Gini imparity

Right Now →

How to measure information gain?

- The Shannon information of an outcome is $\log_2 \frac{1}{p_i}$ → (p_i: probability for event i)
- The entropy - Measure of uncertainity (unpredictability)

$$\text{Entropy} = \sum_i -p_i \log_2 p_i$$

is a sensible measure of expected information content

Examples of Entropy

Example: rolling a die
$p_1 = \frac{1}{6}; \quad p_2 = \frac{1}{6}; \ldots \quad p_6 = \frac{1}{6}$

$$\text{Entropy} = \sum_i -p_i \log_2 p_i =$$
$$= 6 \times (-\frac{1}{6} \log_2 \frac{1}{6}) =$$
$$= -\log_2 \frac{1}{6} = \log_2 6 \approx 2.58$$

The result of a die-roll has 2.58 bit of information

Example: rolling a fake die
$p_1 = 0.1; \ldots \quad p_5 = 0.1; \quad p_6 = 0.5$

$$\text{Entropy} = \sum_i -p_i \log_2 p_i =$$
$$= -5 \cdot 0.1 \log_2 0.1 - 0.5 \log_2 0.5 =$$
$$\approx 2.16$$

A real die is more unpredictable (2.58 bit) than a fake (2.16 bit)

Example: tossing a coin
$p_{\text{head}} = 0.5; \quad p_{\text{tail}} = 0.5$

$$\text{Entropy} = \sum_i -p_i \log_2 p_i =$$
$$= -0.5 \underbrace{\log_2 0.5}_{-1} - 0.5 \underbrace{\log_2 0.5}_{-1} =$$
$$= 1$$

The result of a coin-toss has 1 bit of information

Unpredictability of a dataset (think of a subset at a node)

- 100 examples, 42 positive

$$-\frac{58}{100} \log_2 \frac{58}{100} - \frac{42}{100} \log_2 \frac{42}{100} = 0.981$$

- 100 examples, 3 positive

$$-\frac{97}{100} \log_2 \frac{97}{100} - \frac{3}{100} \log_2 \frac{3}{100} = 0.194$$

Back to Decision trees

Smart Idea: Ask about the attribute which maximize the expected reduction of the entropy,

Information gain:

↳ Ask about attribute A for a data set S that has the Entropy $Ent(S)$, and get subsets S_v according to the value of A

$$Gain = \underbrace{Ent(S)}_{\text{Before}} - \underbrace{\sum_{v \in Values(A)}}_{\text{weighted sum}} \frac{|S_v|}{|S|} Ent(S_v)$$ After

What is the entropy of this binary dataset (attributes=$\{A, B, C, D\}$, $n = 25$)?

$$Ent = -\tfrac{12}{25}\log_2\tfrac{12}{25} - \tfrac{13}{25}\log_2\tfrac{13}{25} \approx \mathbf{0.9988}$$

$A = \bullet$: $\frac{3}{6}$ positive $\rightarrow 1.0$
$A = \circ$: $\frac{9}{19}$ positive $\rightarrow 0.9980$
Expected: $\frac{6}{25} \cdot 1.0 + \frac{19}{25} \cdot 0.9980 \approx \mathbf{0.9985}$

$B = \bullet$: $\frac{9}{11}$ positive $\rightarrow 0.684$
$B = \circ$: $\frac{3}{14}$ positive $\rightarrow 0.750$
Expected: **0.721**

$C = \bullet$: $\frac{6}{12}$ positive $\rightarrow 1.0$
$C = \circ$: $\frac{6}{13}$ positive $\rightarrow 0.9957$
Expected: **0.9977**

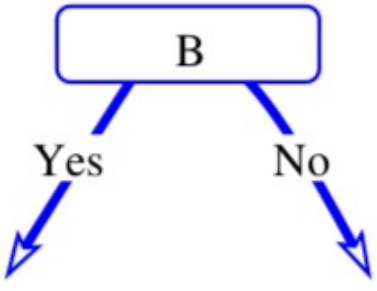
$D = \bullet$: $\frac{3}{5}$ positive $\rightarrow 0.9710$
$D = \circ$: $\frac{9}{20}$ positive $\rightarrow 0.9928$
Expected: **0.9884**

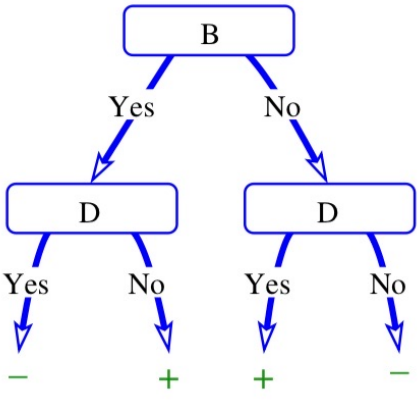
A	B	C	D	
○	●	●	○	+
●	●	○	○	+
○	○	○	○	
○	○	●	●	+
○	●	○	○	+
●	○	●	○	
○	●	●	○	+
○	○	○	○	
●	○	○	○	
○	●	●	○	+
○	○	○	●	+
○	○	●	○	
●	●	●	○	+
○	●	○	●	
○	○	○	○	
○	○	●	○	
○	●	●	●	
○	●	○	○	+
○	○	●	○	
●	○	○	○	
○	●	○	○	+
○	○	●	●	+
●	●	○	○	+
○	○	○	○	
○	○	●	○	

$$Gain(A) = 0.9988 - 0.9985 = \mathbf{0.0003}$$
$$Gain(B) = 0.9988 - 0.7210 = \mathbf{0.2778}$$
$$Gain(C) = 0.9988 - 0.9977 = \mathbf{0.0011}$$
$$Gain(D) = 0.9988 - 0.9884 = \mathbf{0.0104}$$

Attribute B gives most information gain

Then D afterwards





Greedy Approach: Choose the attribute which tells us most about the answer

In sum, we need to find good questions to ask. More than one attribute could be involved in one question.

Gini Impurity

↳ Another definition of predictability (impurity)

$$\sum_i P_i(1-P_i) = 1 - \sum_i P_i^2$$

(P_i: Probability for event i)

↳ The expected error rate at a node N, if the category label is randomly selected from the class distribution present at N

↳ Similar to the entropy but more strongly peaked at equal probabilities

Overfitting

Overfitting: When the learned models are overly specialised for the training samples

↳ Good results on training data but generalizes poorly

When does this occur:

- Non representative sample
- Noisy examples
- Too complex models

What can be done about it?

Choose a simpler model and accept some errors for the training examples

Separate the available data into two sets of examples

- Training set: To form the learned model
- Validation set: To evaluate the accuracy of this model

The motivations:

- The training may be misled by random errors but the validation set is unlikely to exhibit the same random fluctuations
- The validation set to provide a safety check against overfitting the spurious characteristics of the training set

Validation Set needs to be large enough to provide statistically meaningful instances

Reduced Error Prunning

Split the data into training and validation set

Do until further pruning is harmful:

- Evaluate impact on validation set of pruning each possible node
- Greedily remove the one that most improves validation set accuracy

Produces smallest version of most accurate subtree

Possible ways of improving/extentending the decision trees:

- Avoid overfitting
  - Stop growing when the data split not statistically significant
  - Grow full tree, then post-prune

- A collection of trees
  - Bootstrap aggregating (bagging)
  - Decision Forests

Challenges in Machine Learning

Basic idea

↳ How should we select/determine the right model f from data?

Given training data: $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$

of inputs $x_i \in \mathbb{R}^d$ and their labels y_i

Compute the misclassification rate on D } note: $\text{Ind}(x) = 1$ if $x = \text{TRUE}$ otherwise $\text{Ind}(x) = 0$

$$err(f, D) = \frac{1}{n} \sum_{i=1}^{n} \text{Ind}(f(x_i) \neq y_i)$$

Overfitting: Models are overly specialized for the training samples

↳ Good results on training data, but generalizes poorly due to

↳ non-representative sample

↳ Noisy examples

↳ If we are in a data-rich situation:

→ Partition the data into three sets: Training set, Validation set and Test set for assessment of the generalization error of the final chosen model.

Curse of Dimensionality

Image: inputs represented by 30 features but some of them are less relevant to target function. Will you use all of them?

- Easy problems in low dimensions are harder in high-dimensions
- In High dimensions everything is far from everything else
  ↳ Issues in Nearest Neighbours
- Any method that attempts to produce locally varying functions in small isotropic neighborhood will run into problems in high dimensions

Intuitions in low dimension do not apply in high dimensions. Real world → 30 but we deal with data for instance in 1000-d

- Uniform distribution on hypercube
- Volume of hypersphere

Techniques for dimensionality reduction / feature selection exists

The Bias-Variance Trade-off

Let us imagine we could repeat the modelling for many times - each time by gathering new set of training samples, D

The resulting models will have a range of prediction due to randomness in the underlying data set

- **Bias:** The difference between the average prediction of our model and the correct value
- **Variance:** The variability of a model prediction for a given point between realizations of the model

Let us consider

$f(x)$: true function

$\hat{f}_D(x)$: Prediction function (= model) estimated with D

And a conceptual tool:

$E_D[\hat{f}_D(x)]$: average of models due to different sample sets

The mean square error (MSE) for estimating $f(x)$

$$E_D[(\hat{f}_D(x) - f(x))^2] = E_D[(\hat{f}_D(x) - E[\hat{f}_D(x)])^2] + (E[\hat{f}_D(x)] - f(x))^2$$

Variance + (Bias)^2

Characterization of a classifier: Bias

Bias of a classifier is the discrepancy between its average estimated and true function

$$E[\hat{f}_D(x)] - f(x)$$

Regression

Function approximation

- How do we fit this dataset D?

  $D = \{(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)\}$

  → of n pairs of inputs x_i and targets $y_i \in R$ D can be measurements in an experiment

- Task of Regression:

  → to predict target associated to any arbitrary new input

Linear Regression

- Linear regression tries to estimate the function $f(x)$ (x is d-dimensional) and predict output by

  $\hat{f}(x) = \sum_{i=0}^{d} w_i x_i = w^T x$

- How to measure the error for N samples:

  Mean Squares is used to approximate $\hat{f}(x)$ to $f(x)$

- Mean square error

  $E_{in}(\hat{f}) = \frac{1}{N} \sum_{n=1}^{N} (\hat{f}(x_n) - y_n)^2$

- Minimizing in-Sample MSE

  $E_{in}(w) = \frac{1}{n} \sum_{n=1}^{n} (w^T x_n - y_n)^2 = \frac{1}{N} \| Xw - Y \|^2$

  Where: $X = \begin{bmatrix} x_i^T \\ \vdots \\ x_n^T \end{bmatrix}$, $Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$

Residual Sum of Squares

- The sum of squared errors is a convex function of w

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

- The gradient with respect to the weights is:

  $\frac{d}{dw} E_{in}(w) = 2X^T(Xw - Y)$

- The weight vector that sets the gradient to zero minimises the errors

  $X^T X w = X^T Y$

  → $w = (X^T X)^{-1} X^T Y$

RANSAC: Random Sampling Consensus

Repeat M times

- Sample two points to estimate the line
- Calculate the number of inliers or posterior likelihood for relation
- Choose relation to maximize the number of inliers
  (Last two in Least Median of Square (LMeds))
- Calculate error of all data
- Choose relation to minimize median of error

RANSAC: RANdom SAmpling Consensus

Objective

Robust fit of model to data set S which contains outliers

Algorithm

(i) Randomly select a (minimum number of) sample of s data points from S and instantiate the model from this subset.

(ii) Determine the set of data points S_i which are within a distance threshold t of the model. The set S_i is the consensus set of samples and defines the inliers of S.

(iii) If the subset of S_i is greater than some threshold T, re-estimate the model using all the points in S_i and terminate

(iv) If the size of S_i is less than T, select a new subset and repeat the above.

(v) After N trials the largest consensus set S_i is selected, and the model is re-estimated using all the points in the subset S_i

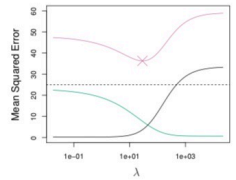
(in Hartley and Zisserman, adapted from Fischler '81)

Ridge Recession

- Similar to least squares but minimises different quantity:
  $RSS + \lambda \sum_{i=1}^{d} \omega_i^2$
  The second term is called Shrinkage penalty)
- Shrinkage penalty: Small when ω_i are close to zero
- The parameter λ: Control the relative impact of the two terms, the selection is critical!
  - gradient decent
  - Coordinate decent

Ridge Regression Bias/Variance

- Green: Variance
- Black: Bias
- Purple: MSE

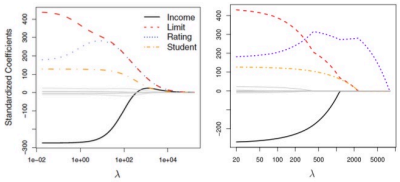


Increased λ decreases variance while increasing bias

The lasso (Least absolut shrinkage and selection operator)

- Similar to ridge regression but with slightly different term:
  $RSS + \lambda \sum_{i=1}^{d} |\omega_i|$
- The lasso could be proven mathematically that some coefficients end up being set to exactly zero
  - Variable selection
  - Yield sparse model

Comparison of estimated coefficients



Ridge regression The Lasso

Another formulation

- For every value of λ there is some s such that the the equations will give the same coefficients estimate

- Ridge regression: Minimizing $RSS + \lambda \sum_{i=1}^{d} \omega_i^2$

  Minimizing RSS, subto $\sum_{i=1}^{d} \omega_i^2 \leq s$

- Lasso: Minimizing $RSS + \lambda \sum_{i=1}^{d} |\omega_i|$

  RSS, subto $\sum_{i=1}^{d} |\omega_i| \leq s$

K-NN Regression (non-Parametric)

- Similar to the K-NN classifier
- To regress Y for a given value of X, consider K closest points to x in training data and take the average of the reponses $\Leftrightarrow f(x) = \frac{1}{K} \sum_{x \in N_i} y_i$
- Larger values of K provide a smoother and less variable fit (lower variance)

Parametric or non-Parametric ← Important

- if the parametric form is close to the true form of f, the parametric approach will outperform the non-parametric
- Parametric methods tend to outperform non-parametric when there is a small number of observations per predictor (high dimension)
- Interpretability stand point: Linear regression prefferred to KNN if the test MSEs are similar or slightly lower.

Summary

- KNN regression
- Ridge regression → Gradient descent, Coordinate descent
- Lasso
- RANSAC
- MSE, RSS

Another formulations

For every value of λ there is some s such that the equations will give the same coefficient estimates:

- Ridge regression: Minimizing $RSS + \lambda \sum_{i=1}^{d} w_i^2$

  Minimizing $RSS, sub.to \sum_{i=1}^{d} w_i^2 \le s$

- Lasso: Minimizing $RSS + \lambda \sum_{i=1}^{d} |w_i|$

  $RSS, sub.to \sum_{i=1}^{d} |w_i| \le s$

Probabilistic Reasoning

Probability theory in ML

↳ Incorporate probabilistic thinking at all levels
- We start with incomplete knowledge
- We then reduce uncertainitity by incorporating observations
- We then propagate belief and update our estimation

Advantages of probability based Methods

↳ **interpretability:** More transparent and Mathematical rigous than other ML-models

↳ **Transparency:** Assumptions can be made more explicit than in other methods

↳ **Efficiency:** Works with Poor data

↳ **Flexibility:** Easy to merge different parts of a complex System and update Current knowledge

↳ **Encompassing:** Aspects of learning and interferences can be cast under the same theory

Disadvantages of probabilistic Methods

Complicated: Often hard to derive closed solutions. Need to resort to computation and heuristic approximations

Scalability: Not computational scalable to large datasets but many argue that the need for large data sets is a disadvantages

Notes from slides

Probability Theory in ML

- We start with incomplete knowledge
- We then reduce uncertainty by incorporating observation
- We then propagate belief and update our estimate

Probabilistic Machine Learning provides something like a unified theory for ML

Statistic Crash Course

- **Random Variables:**
  - Sample space: set of outcomes are not numbers.
  - We need to "convert" outcomes to number
  - Let's map the set of outcome to a set of number!

- **Probability distribution:**
  - The probability distribution function PDF of a r.v maps its range to positive numbers
    $Pr(x): X \to \mathbb{R} > 0$
  - $Pr(x)$ describes how probability density is distributed over the range of X, e.g the probability of $0 \leq X \leq 1$ is given by:
    $P[0 \leq x \leq 1] = \int_{x=0}^{1} Pr(x)\,dx$
  - X is distributed $Pr(x)$ written more compactly
    $X \sim Pr(x)$

- **Independence**
  - Indepence if the joint distribution can be factorised $P(A \cap B) = P(A)\,P(B)$
    $\Rightarrow P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A)\,P(B)}{P(B)} = P(A)$
    B happening says nothing about A
  - Bayes Theorem ← Memorise its derivation
    $P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$

Common Distribution

- Bernoulli: Domain: Binary variables ($x \in \{0,1\}$)
  Parameter: $\lambda = P[x=1]$, $\lambda \in [0,1]$
  $P[x=0] = 1-\lambda$ and
  $Pr(x) = \lambda^x (1-\lambda)^{1-x} = \begin{cases} \lambda, & \text{if } x=1 \\ 1-\lambda, & \text{if } x=0 \end{cases}$
- Categorical: Domain: discrete variables ($x \in \{x_1, \dots x_k\}$)
  Parameter: $\lambda = [\lambda_1, \dots \lambda_k]$ $P[x_k=1] = \lambda_k$
  with $\lambda_k \in [0,1]$ and $\sum_{k=1}^{K} \lambda_k = 1$
- Gaussian distribution: Univariate normal distribution
  Domain: Real number ($x \in \mathbb{R}$)
  $Pr(x; \mu, \sigma^2) = N(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$

Central Limit Theorem: The distribution of a large number of independent, identically distributed variables will tend to normal distribution, regardless of the underlying distribution.

- Gaussian Distribution: D Distribution
  - Known as multivariate normal distribution $X = \begin{bmatrix} x_1 \\ x_2 \\ x_D \end{bmatrix}$ $\mu = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_D \end{bmatrix}$ $\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1D} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma \\ \vdots & & \ddots & \\ \sigma_{D1} & \cdots & & \sigma_D^2 \end{bmatrix}$
  - Domain: Real numbers $(x \in \mathbb{R}^D)$

$$Pr(x;\mu,\Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{\frac{1}{2}}} \cdot E\left[-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right]$$

Eigen Value decomposition of Σ matrix

$$\Sigma = Q \Lambda Q^T = [u|v]\, \text{diag}[\lambda_1, \lambda_2]\, [u|v]^T$$

Expectations

- $E[x] = \mu_x = \int x\, Pr(x)\, dx$ $\Rightarrow$ The center of gravity of a distribution
- $E[x] = \mu_x = \int x\, Pr(x)\, dx$ Sampled Expected Value = Mean $\bar{\mu_x} = \frac{1}{N}\sum_i^N x_i$

- Variance

$Var[x] = \sigma_x^2 = E\left[(x - E[x])^2\right]$ $\bar{\sigma_x^2} = \frac{1}{N-1}\sum_i^N (x_i - \bar{\mu_x})^2$

The spread of a distribution

- Covariance
- $\sigma_{x,y} = E\left[(x - E[x])(y - E[y])\right]$ $\bar{\sigma}_{xy} = \frac{1}{N-1}\sum_i^n (x_i - \bar{\mu_x})(y_i - \bar{\mu_y})$
- $\Sigma_x = E\left[((x - E[x])(x - E[x]))^T\right]$

→ Unbiased Sample Co-variance

Probabilistic Machine Learning

General ML Outline

Data: $\{(x,y)_1, (x,y)_2 \ldots (x,y)_n\}$

→ Where x are features (independent variables) and y is the answer (dependent variable)

- if Y is discrete, finite: Classification
- if Y is continuous: Regression

Learning: we want to learn how variables are related

- we estimate $Pr(x,y)$ from observations $\{(x,y)_i\}$

Inference: we want to predict an answer given an observation

- we estimate $Pr(Y|X=x)$ from observation $\{(x,y)_i\}$

Bayes' Rule

$$Pr(y \mid \mathbf{X} = \mathbf{x}) = \frac{Pr(\mathbf{x} \mid Y = y) Pr(Y = y)}{Pr(\mathbf{X} = \mathbf{x})}$$

- $Pr(\mathbf{x} \mid Y = y)$ ← Likelihood represents the probability density of observing data $\mathbf{x}$ given the *hypothesis* $Y = y$.
- $Pr(Y = y)$ ← Prior represents the knowledge about Y before any observation.
- $Pr(y \mid \mathbf{X} = \mathbf{x})$ ← Posterior represents the probability density of hypothesis y given observation $\mathbf{X} = \mathbf{x}$.
- $Pr(\mathbf{X} = \mathbf{x})$ ← Evidence describes how well the model fits the evidence.

$$Pr(\mathbf{X} = \mathbf{x}) = \begin{cases} \sum_y Pr(\mathbf{x} \mid Y = y) Pr(Y = y) & \text{classification} \\ \int_y Pr(\mathbf{x} \mid Y = y) Pr(Y = y) & \text{regression} \end{cases}$$

Probabilistic Regression

- Regression via Conditional probability
- Compute posterior of Y: $Pr(Y|X=x) = Pr(x,y)/Pr(X=x)$
- Compute Conditional expectation: $E[Y|X=x]$

Explicit Regression Model:

- Define a deterministic model $Y = f(x) + \varepsilon$
- Define probability distribution of the error = $\varepsilon = Y - f(x)$
- Explicit parameters in $f(x)$

Skipping

Most Probable hypothesis

- Maximum A posterior (MAP)

  choose hypothesis from Y with the highest probability given observed data x:

  $\rightarrow y_{map}(x) = \underset{y \in Y}{\operatorname{argmax}} \; Pr(Y=y|X=x)$

  $\underset{y \in Y}{\operatorname{argmax}} \; \frac{Pr(X=x|Y=y)\,Pr(Y=y)}{Pr(X=x)}$

  $\underset{y \in Y}{\operatorname{argmax}} \; Pr(X=x|Y=y)\,Pr(Y=y)$

- Maximum Likelyhood (ML):

  If we do not know prior distribution, then choose hypothesis with highest likelyhood of generating the observed data:

  $\rightarrow y_{MLE} = \underset{y \in Y}{\operatorname{argmax}} \; Pr(X=x|Y=y)$

Before I took a COVID test, the doctor said 99% of the people in the area have COVID, and 90% of those with COVID are testing positive. A few days later the doctor called and said my test was positive, and that the probability I have COVID given this positive test is p% — I can't remember because I was in shock. Find the minimum value of p such that I can compute the probability I got a positive test but don't have COVID, and then compute the maximum probability I don't have COVID given my positive test. *Show your work.*

Lecture 6- Learning as Inference

Probabilistic Classification and Regression

$$Pr(Y \mid X=x) = \frac{Pr(x \mid Y=y)\, Pr(Y=y)}{Pr(X=x)}$$

Likelyhood, Prior, Evidence

- Classification: Y is discrete, finite
- Regression: Y is continous

How do we obtain these distributions

↳ Given:
  ↳ the training data $D = \{(x,y), (x,y), \ldots (x,y)_N\}$
  ↳ New observation x

Estimate the posterior probability of y $Pr(y \mid x, D)$

Discriminative vs Generative Model

Discriminative model:
↳ Models $Pr(y, x, D)$ directly } Calculates Prior directly
↳ Logistic regression

Generative Modelling
↳ Models $Pr(x \mid y, D)$
↳ Naives Bayes

Parametric vs Non-Parametric Inference

$Pr(y \mid x) = Pr(y \mid x, \theta)$

The distribution is characterized by parameters θ

Parametric Inference
↳ Estimate θ using D
↳ Compute $Pr(y \mid x, \hat{\theta})$ to make inference

Learning corresponds to estimating θ

Non-Parametric Inference
↳ Estimate $Pr(\theta, D)$
↳ Compute $Pr(y \mid x, D)$ from $Pr(y \mid x, \theta, D)$ $Pr(\theta \mid D)$ by marginalizing out θ

grows with Data

Fundamental Assumption iid

↳ observations are independent and identically distributed

$D = \{o_1, o_2 \ldots o_n\}$ $o_i = (x,y)_i$

↳ The likelihood of the whole data set can be factorised

$$Pr(D) = Pr(o_1, o_2, o_n) = \prod_{i=1}^{N} Pr(o_i)$$

↳ Taking the log creates the log-likelyhood

$$\text{Log } Pr(D) = \sum_{i=0}^{N} \text{Log } Pr(o_i)$$

Maximum Likelyhood Estimate (ML)

$Pr(x|y) \equiv Pr(x|y,\theta)$ or $Pr(y|x) \equiv Pr(y|x,\theta)$

Find the parameter values that makes the data most likely

- ML optimality is defined as maximizing the likelyhood of D:

$$\theta_{ML} = \arg\max_{\theta} P(D,\theta) = \arg\max_{\theta} \log P(D|\theta)$$

D = Distribution θ = Maximize this

- We can then approximate distributions given the data:

$Pr(x|y,D) \simeq Pr(x|y,\theta_{ML})$ or $Pr(y|x,D) \simeq Pr(y|x,\theta_{ML})$

Lecture 7- Priori and latent Variable

Check your understanding!

Goal of lecture: Answer both

Consider a dataset $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$ of N independent and identically distributed observations where each $\mathbf{x}_n$ is a p-dimensional real vector. Assume the random variable Y_n is distributed Laplacian with a mean $\boldsymbol{\beta}^T \mathbf{x}_n$ and known scale parameter $b > 0$. In other words, $Y_n | \mathbf{x}_n, \boldsymbol{\beta} \sim \mathcal{L}(\boldsymbol{\beta}^T \mathbf{x}_n, b)$. Define the a priori distribution of the parameters $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_p)$ *multivariate* Gaussian with parameters mean $\mathbf{0}$ and variance $\sigma^2 \mathbf{I}$.

1. Derive the maximum likelihood (ML) estimate of $\boldsymbol{\beta}$.
2. Derive the maximum a posteriori (MAP) estimate of $\boldsymbol{\beta}$.

Infomation given: $Pr(Y_n | X_n, B) = \frac{1}{2B} \exp\left[-\frac{|Y_n - X^T B|}{b}\right]$

$Pr(B) = N(0, \sigma^2 I)$

Maximum likelyhood (ML) estimate:

- $Pr(Y_n | X_n, B) = \frac{1}{2B} \exp\left[-\frac{|Y_n - X^T B|}{b}\right]$

  $P(B) = N(0, \sigma^2 I)$

  log likelyhood

- $\log Pr(D|B) = \sum_{n=1}^{n} \log Pr(Y_n | X_n, B)$

Maximum likelyhood estimate is then

$$B_{ML} = \arg\max_B \log Pr(D|B) = \arg\max_B \sum_{n=1}^{N} \log Pr(Y_n | X_n, B)$$

$$= \arg\max_B \sum_{n=1}^{N} \log\left(\frac{1}{2B} \exp\left[-\frac{|Y_n - X_n^T B|}{b}\right]\right)$$

$$= \arg\max_B \sum_{n=1}^{N} -\log(2b) - \frac{1}{b} |Y_n - X_n^T B|$$

$$= \arg\min_B N \log(2b) + \frac{1}{b} \sum_{n=1}^{N} |Y_n - X_n^T B|$$

$$= \arg\min_B \|Y - X^T B\|_1 \leftarrow \text{L1 Sparsing the Solution}$$

where $Y = (Y_1, Y_2, Y_n)$ and $X = [X_1, X_2 \ldots X_n]$

Maximum a Posterior estimation

- ML estimates θ to maximize probability of D
- MAP estimation chooses the most likely θ given D

$$\theta = \arg\max_\theta Pr(\theta|D)$$

$$= \arg\max_\theta \frac{Pr(\theta) Pr(D|\theta)}{P(D)}$$

$$= \arg\max_\theta Pr(\theta) Pr(D|\theta)$$

$$= \arg\max_\theta \left[Pr(\theta) \prod_{n=1}^{N} Pr(X_n|\theta)\right] =$$

Works like a regularisor

$$\arg\max_\theta \left[\log Pr(\theta) + \sum_{n=1}^{N} \log Pr(X_n|\theta)\right]$$

→ Ridge Regression → Fit the data and keep the weights small

1) arg max
2) Bayes theorem without nominator
3) Add log
4) $\sum_{n=1}^{N}$ and start logging
5) Arg min

Impact of different Priors

$$w^*(P) = \arg\min_w \left[\sum_{n=1}^{N} (y_n - w^T x_n)^2 + \lambda \sum_{d=1}^{D} |w_d|^p \right]$$

Fit data — Regularise

- $\lambda = 0$ or $w_d \sim$ uniform → Fit the data
- $\lambda > 0$ → Fit the data and make w short (Ridge Recession) ← assuming normal
- $\lambda > 0$ → Fit the data and make w sparse (Lasso) ← assuming laplace

Check your understanding!

Consider a dataset $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$ of N independent and identically distributed observations where each $\mathbf{x}_n$ is a p-dimensional real vector. Assume the random variable Y_n is distributed Laplacian with a mean $\boldsymbol{\beta}^T \mathbf{x}_n$ and known scale parameter $b > 0$. In other words, $Y_n | \mathbf{x}_n, \boldsymbol{\beta} \sim \mathcal{L}(\boldsymbol{\beta}^T \mathbf{x}_n, b)$. Define the a priori distribution of the parameters $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_p)$ *multivariate* Gaussian with parameters mean $\mathbf{0}$ and variance $\sigma^2 \mathbf{I}$.

1. Derive the maximum likelihood (ML) estimate of $\boldsymbol{\beta}$.
2. Derive the maximum a posteriori (MAP) estimate of $\boldsymbol{\beta}$.

Derive the maximum a posteriori (MAP) estimate of $\boldsymbol{\beta}$.

- $Pr(y_n | \mathbf{x}_n, \boldsymbol{\beta}) = \frac{1}{2b} \exp\left[-\frac{|y_n - \mathbf{x}_n^T \boldsymbol{\beta}|}{b} \right]$
- $P(\boldsymbol{\beta}) = \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$

The MAP estimate of $\boldsymbol{\beta}$ is defined

$$\begin{aligned} \boldsymbol{\beta}_{\text{MAP}} &= \arg\max_{\boldsymbol{\beta}} Pr(\boldsymbol{\beta}|\mathcal{D}) = \arg\max_{\boldsymbol{\beta}} Pr(\mathcal{D}|\boldsymbol{\beta}) Pr(\boldsymbol{\beta}) \\ &= \arg\max_{\boldsymbol{\beta}} \log Pr(\boldsymbol{\beta}) + \sum_{n=1}^{N} \log Pr(y_n | \mathbf{x}_n, \boldsymbol{\beta}) \\ &= \arg\min_{\boldsymbol{\beta}} \frac{1}{2\sigma^2} \|\boldsymbol{\beta}\|_2^2 + \frac{1}{b} \|\mathbf{y} - \mathbf{X}^T \boldsymbol{\beta}\|_1 \end{aligned}$$

ML:	$\mathcal{D}$	$\rightarrow$	θ_{ML}	$\rightarrow$ $Pr(y\|\mathbf{x}, \theta_{\text{ML}})$
MAP:	$\mathcal{D}, Pr(\theta)$	$\rightarrow$	θ_{MAP}	$\rightarrow$ $Pr(y\|\mathbf{x}, \theta_{\text{MAP}})$
Bayes:	$\mathcal{D}, Pr(\theta)$	$\rightarrow$	$Pr(\theta\|\mathcal{D})$	$\rightarrow$ $Pr(y\|\mathbf{x}, \mathcal{D})$

1. consider θ as a random variable (same as MAP)
2. characterize θ with the posterior distribution $Pr(\theta|\mathcal{D})$ given the data
3. compute new predicting posterior $Pr(y|\mathbf{x}, \mathcal{D})$ marginalizing over θ (predictive posterior)

$$Pr(y|\mathbf{x}, \mathcal{D}) = \int_{\theta \in \Theta} Pr(y|\mathbf{x}, \theta) Pr(\theta|\mathcal{D}) d\theta$$

Occam Razor

Choose the simplest explanation for the data

- Number of model Parameters
- number of data Points
- Model fit to the data

Unsupervised Learning

Clustering vs Classification

- We don't know their classes

• k-means Algorithms

- Describes each class with a centroid
- Favors spherical classes
- Euclidean distance is isotropic

If no class independence → Expectation Maximation

$$Pr(x|\theta) = \sum_{k=1}^{k} \pi_k \, Pr(X|\theta_k)$$

with $\theta = \{\pi_1, \ldots, \pi_k \theta_1 \ldots \theta_k\}$

EM for two Gaussians

For each sample x_i introduce a *hidden variable* h_i

$$h_i = \begin{cases} 1 & \text{if sample } x_i \text{ was drawn from } \mathcal{N}(x|\mu_1, \sigma_1^2) \\ 2 & \text{if sample } x_i \text{ was drawn from } \mathcal{N}(x|\mu_2, \sigma_2^2) \end{cases}$$

and come up with initial values

$$\Theta^{(0)} = (\pi_1^{(0)}, \mu_1^{(0)}, \sigma_1^{(0)}, \mu_2^{(0)}, \sigma_2^{(0)})$$

for each of the parameters.

EM is an *iterative algorithm* which updates $\Theta^{(t)}$ using the following two steps...

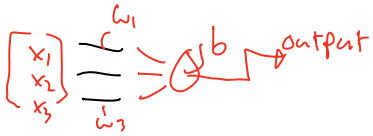
Lecture 7- SVMs

Concept learning: Supervised learning of Boolean valued functions. Learn from positive and negative examples to classify yes/no corrections

Examples of concepts: Is there are a dog present, mammal, vehicle and so on. **Input is an array of attribute of values**. You could use abstract concepts such as "is there a crime committed here" and so on.

Artificial Neural Neuron

Gets a input in vector format. Weights in vector format. B is the threshold for it to output either yes or no. And there's a output.



$$y = \text{Sign}\left(\sum_i x_i w_i - b\right)$$

Training ANNs

- Find the best weights and biases. Done by Delta Rule and Perceptron learning

Perceptron Learning (Binary output):

1) Perceptron Learning

- Incremental Learning
- Weights only change when the output is wrong
- Update rule
- Always converge if the problem is solvable

} normal of the wrong

Delta rule (LMS, continuous output)

2) Delta Rule

- Incremental Learning
- Weights always change
- Converges only in the mean
- Will find an optimal solution even if the problem cannot be fully solved.

Structural Risk Minimization

Margins: A safe distance d from the margin to the datapoints. Wide margins restrict the possible hyperolanes to choose from. Less risk to choose a bad hyperplane. Goal is to find the largest margin from all data points. This sill most likely generalise the best.

↳ Separating Hyperplane

$\vec{w}^T \vec{x} = 0$

↳ Hyperplane with a margin

$\vec{w}^T \vec{x} \geq 1 \quad t = 1$ ↳ Positive target

$\vec{w}^T \vec{x} \leq -1 \quad t = -1$ ↳ Negative target

↳ Combined

$t \vec{w}^T \vec{x} \geq 1$

How wide is the Margin

1) Select 2 point $\vec{p}, \vec{q}$ on the two margins:
$\vec{w}^T \vec{p} = 1 \quad \vec{w} \vec{q} = -1$

2) Distance between $\vec{p}$ and $\vec{a}$ along $\vec{w}$

$$2d = \frac{\vec{w}^T}{\|w\|} (\vec{p} - \vec{q})$$

Support Vectors

- Moving a low dimensional data to a high-Dimensional makes stuff linearly seperatable.
  - ↳ Two problems
    - Many free parameters → Bad generalization
    - Extensive Computations.

Ensemble Learning

- What make a crowd wise?
  - Diversity of opinion
  - Independence
  - Aggregation
  - Decentralization

We will exploit Wisdom of crowd ideas for specific task by combining classifies and aim to combine independence and diverse classifiers.

Bagging

→ Use Bootstrap replicates of training set by sampling with replacement. On each replicate, learn one model

High variance: Classifiers produce differing decision boundaries which are highly dependent on the training data

Low bias: Classifies produce decision boundaries which are on average are good approximations to the true decision boundary.

Ensemble prediction using diverse high-variance, low bias classifiers reduce the total variance of the ensemble classifier.

Ensemble Method / Random Forests

- Randomness?
  - Sampling training data (same as bagging)
  - Feature selection at each node
- Trees are less correlated in example higher variance between weak learners
  - A classifier suited to multi-class problem

Ensemble Method / Boosting

↳ You apply weights to certain data points. Thereafter apply learner to weighted samples. Increase weights of misclassified examples

Ensemble Method / Dropout

↳ Similar to decision trees cutting out leaves/nodes. Here you cut out parts of your neural network.

Ensemble / Summary

- Combine many (high bias) weak classifier / regressors into a strong classifier; boosting
  - ↳ if weak classifiers are chosen and combined using knowledge of how well they are others performed on the task on training data
  - ↳ the selection and combination encourages the weak classifiers to be complementary and de-correlated