DD2434 Machine Learning, Advanced Course

Lecture 10: Sampled and Ensemble Models

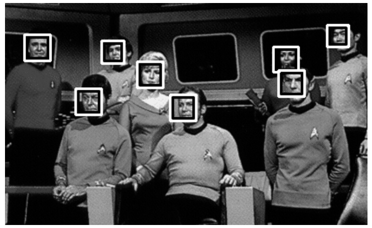
Hedvig Kjellström
hedvig@kth.se
https://www.kth.se/social/course/DD2434/

Complex functions

Global analytical model or collection of local models

Example: subspace of faces in the entire image state-space

- High-dim
- Non-linear
- Singularities



(Wang, CVIU 2007)

Representation Learning = model subspace as efficiently as possible

Ensemble and Sampled Learning = do not try to model globally at all!

Today

Put the 3 methods in a probabilistic context

Boosting (Murphy 16.1-16.4)
- AdaBoost classification
- Relations to Random Forests, Neural Networks

Sampling (Murphy 23.1-23.4, 24.1, 24.3.7)
- Monte Carlo / CDF sampling
- Importance sampling
- MCMC sampling

k Nearest Neighbor (Murphy 1.4.2, Everson and Fieldsend 1)
- Probabilistic classification framework

Particle filtering (Murphy 23.5)

Boosting

Adaptive Basis Function Model

Kernel based methods (Lecture 7):

$$f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}), \quad \phi(\mathbf{x}) = [\kappa(\mathbf{x}, \mu_1), ..., \kappa(\mathbf{x}, \mu_N)]$$

Feature based methods (Adaptive Basis Function Models):

$$f(\mathbf{x}) = w_0 + \sum_{m=1}^{M} w_m \phi_m(\mathbf{x})$$

Special cases:
Random Forests (Bagging)
Boosting
Feedforward Neural Networks

where $\phi_m(\mathbf{x})$ are learned from data

Adaptive Basis Function Model

Goal: Solve the optimization problem

$$\min_f \sum_{i=1}^{N} L(y_i, f(\mathbf{x}_i))$$

where $L(y, y')$ is some loss function and $f(\mathbf{x})$ is an ABM

HARD!

Boosting

A **greedy** approach:
Define a **weak learner**, e.g., a linear classifier or regressor
For each round m
- Train the weak learner on the dataset $\mathcal{D}$, call the trained learner ϕ_m
- Give the data points that fit with ϕ_m low weight, the data points in conflict with ϕ_m high weight

The final learner is a weighted sum of all weak learners ϕ_m

Convergence guaranteed – with enough iterations, the error will be 0

Boosting

Boosting approach – use greedy approach, solve for $f(\mathbf{x})$ term by term:

Define $\phi_m(\mathbf{x}) \equiv \phi(\mathbf{x}, \nu_m)$

$f_0(\mathbf{x})$ is some "good enough" baseline function

Iterate:

$$(\beta_m, \nu_m) = \arg\min_{\beta,\nu} \sum_{i=1}^{N} L(y_i, f_{m-1}(\mathbf{x}_i) + \beta\phi(\mathbf{x}_i, \nu))$$

$$f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \beta_m \phi(\mathbf{x}, \nu_m)$$

AdaBoost

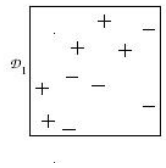
Popular algorithm for binary classification ($y \in \{-1, +1\}$) with exponential loss ($L(y, y') = \exp(-yy')$)

Left for report, Task 3.3:

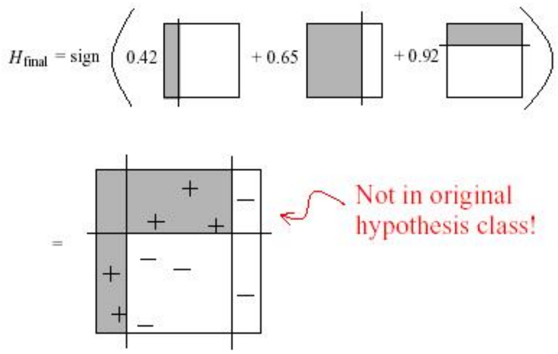
Explain the derivation of Algorithm 16.2 (AdaBoost.M1) from the general boosting algorithm, given the particular labels and loss function.

Implement Algorithm 16.2

AdaBoost Example



AdaBoost Example



AdaBoost

Discuss with your neighbor (5 min):

What happens if the weak learners ϕ_m give **close to** random results?

What happens if the weak learners ϕ_m give **exactly** random results?

Sampling

The Monte Carlo Principle

Start off with **discrete** state space z

Imagine that we can **sample** $z^{(l)}$ from the pdf $p(z)$ but that we do not know its functional form

Might want to estimate for example:

$$E[z] = \sum z\, p(z)$$

$p(z)$ can be approximated by a histogram over $z^{(l)}$:

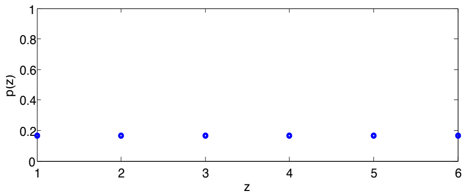
$$\hat{q}(z) = \frac{1}{L} \sum_{l=1}^{L} \delta_{z^{(l)}=z}$$

Example: Dice Roll

The probability of outcomes of dice rolls: $p(z) = \frac{1}{6}$

Exact solution:

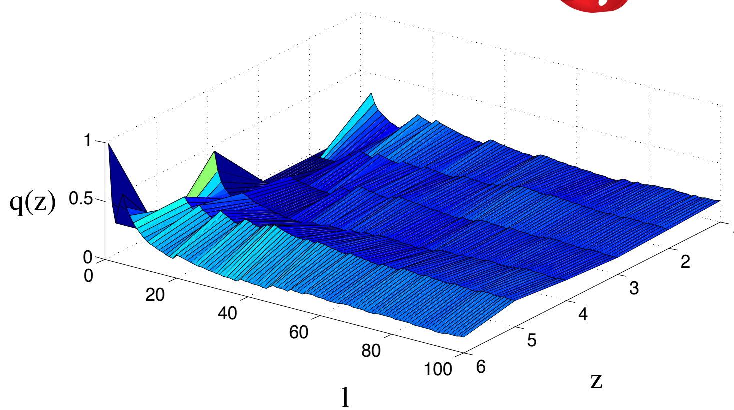
What would happen if the dice was bad?



Monte Carlo approximation:
Roll a dice a number of times, might get

$$z^{(1)} = 6 \quad z^{(2)} = 4 \quad z^{(3)} = 1 \quad z^{(4)} = 6 \quad z^{(5)} = 6$$

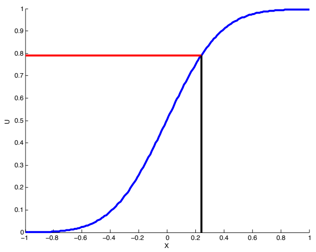
Example: Dice Roll



Monte Carlo Sampling – Inverse Probability Transform

Cumulative distribution function F of distribution f (that we want to sample from)

A uniformly distributed random variable $U \sim U(0,1)$ will render $F^{-1}(U) \sim F$



$f(z)$ does not have to be an analytic function, can also be a histogram like $\hat{q}(z)$!

Importance Sampling

We very often (in particle filters for example) want to approximate integrals of the form

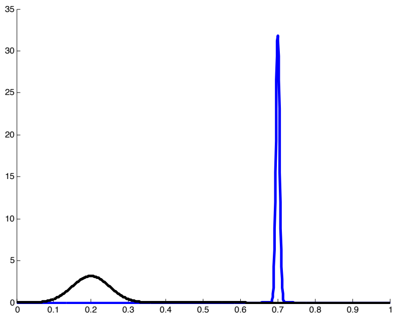
$$E[f] = \int f(x)p(x)dx$$

Monte Carlo sampling approach is to draw samples x^s from $p(x)$ and approximating the integral with a sum

$$E[f] = \int f(x)p(x)dx = \frac{1}{S}\sum_{s=1}^{S} f(x^s)$$

Importance Sampling

Discuss with your neighbor (5 min):
But what if $p(x)$ and $f(x)$ look like this, what happens with the estimation?



Importance Sampling

In these cases, a good idea is to introduce **proposal** $q(x)$ to sample from:

$$E[f] = \int f(x)\frac{p(x)}{q(x)}q(x)dx \approx \frac{1}{S}\sum_{s=1}^{S} w_s f(x^s)$$

where $w_s \equiv \dfrac{p(x^s)}{q(x^s)}$

Reasons:

- $q(x)$ is smoother / less spiky than $p(x)$
- $q(x)$ is of a nicer analytical form than $p(x)$

In general, good to keep $q(x) \propto p(x)$ approximately

Markov Chain Monte Carlo

Standard MC and Importance sampling do not work well in high dimensions

High dimensional space but actual model has lower (VC) dimension => exploit correlation!

Instead of drawing independent samples x^s draw chains of correlated samples – perform random walk in the data where the number of visits to x is proportional to target density $p(x)$

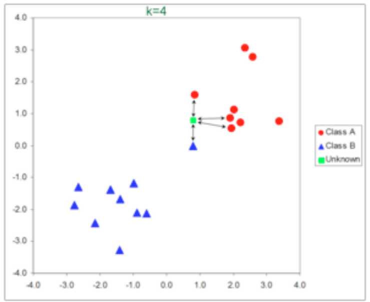
MCMC algorithms:
Gibbs Sampling (special case of)
Metropolis Hastings
Reversible Jump MCMC

k Nearest Neighbor

*k*NN – a Non-Parametric Method

Well known, not repeated here

In Task 3.1 you will implement a binary *k*NN classifier



PNN – a Probabilistic Interpretation of *k*NN (Everson and Fieldsend Section 1)

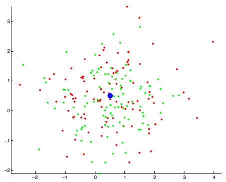
Need to give an ad hoc k value here as well

Probabilistic formulation of *k*NN: $p(y|\mathbf{x}, \mathcal{D})$

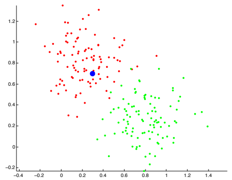
Problem in standard *k*NN: *k* unknown, depends on how correlated data points are "how smooth distribution"

Discuss with your neighbor (5 min):

What is the ideal *k*?



What is the ideal *k*?



PNN – a Probabilistic Interpretation of *k*NN (Everson and Fieldsend Section 1)

Learn *k* from data: Introduce another unknown correlation parameter β, let $\theta = \{k, \beta\}$, integrate out:

$$p(y|\mathbf{x}, \mathcal{D}) = \int p(y|\mathbf{x}, \theta, \mathcal{D}) p(\theta|\mathcal{D}) d\theta$$

where

$$p(y|\mathbf{x}, \theta, \mathcal{D}) = \frac{\exp[\beta \sum_{\mathbf{x}_j \sim \mathbf{x}_i}^{k} u(d(\mathbf{x}_i, \mathbf{x}_j)) \delta_{y_i y_j}]}{\sum_{q=1}^{Q} \exp[\beta \sum_{\mathbf{x}_j \sim \mathbf{x}_i}^{k} u(d(\mathbf{x}_i, \mathbf{x}_j)) \delta_{q y_j}]}$$

Discuss with your neighbor (1 min):

How do you (avoid to) solve this integral?

PNN – a Probabilistic Interpretation of *k*NN (Everson and Fieldsend Section 1)

Reversible Jump MCMC is used to draw samples $\theta^{(t)}$ that approximate $p(\theta|\mathcal{D})$:

$$p(y|\mathbf{x}, \mathcal{D}) \approx \frac{1}{T} \sum_{t=1}^{T} p(y|\mathbf{x}, \theta^{(t)}, \mathcal{D})$$

from the likelihood of data given parameters

$$p(\mathcal{D}|\theta) = \prod_{i=1}^{N} \frac{\exp[\beta \sum_{\mathbf{x}_j \sim \mathbf{x}_i}^{k} u(d(\mathbf{x}_i, \mathbf{x}_j)) \delta_{y_i y_j}]}{\sum_{q=1}^{Q} \exp[\beta \sum_{\mathbf{x}_j \sim \mathbf{x}_i}^{k} u(d(\mathbf{x}_i, \mathbf{x}_j)) \delta_{q y_j}]}$$

Particle Filtering

Particle Filtering

Task: Estimate density $p(\mathbf{z}_t|\mathbf{y}_{1:t})$ over state $\mathbf{z}_t$ given a sequence of observations $\mathbf{y}_{1:t}$

Sequential formulation (simplest case)

$$p(\mathbf{z}_t|\mathbf{y}_{1:t}) \propto p(\mathbf{y}_t|\mathbf{z}_t) \int p(\mathbf{z}_t|\mathbf{z}_{t-1}) p(\mathbf{z}_{t-1}|\mathbf{y}_{1:t-1}) d\mathbf{z}_{t-1}$$

Posterior at *t*

Likelihood at *t*

Motion model at *t*

Posterior at *t*-1

Discuss with your neighbor (1 min):

How do you (avoid to) solve this integral?

Particle Filtering

Correct! Represent the posterior distribution at time t with samples z_t^s

Basic sequential estimation algorithm:

Given set of **particles** $\{z_{t-1}^s\}_{s=1}^S$

Propagate all particles through motion model $p(z_t | z_{t-1})$ to get propagated particles $\{\tilde{z}_t^s\}_{s=1}^S$ which represent the prior at t

Evaluate all particles with likelihood to get weighted particle set $\{w_t^s \tilde{z}_t^s\}_{s=1}^S$ which represent the posterior at t

Degeneracy problem – most particle weights will go to 0!

Particle filtering

Resampling step – central invention of particle filtering

Basic sequential estimation algorithm:

Given set of **particles** $\{z_{t-1}^s\}_{s=1}^S$

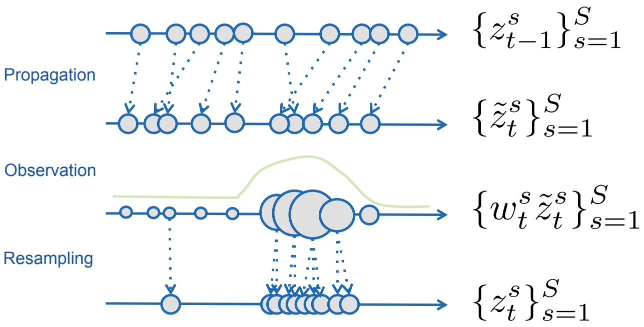
Propagate all particles through motion model $p(z_t | z_{t-1})$ to get propagated particles $\{\tilde{z}_t^s\}_{s=1}^S$ which represent the prior at t

Evaluate all particles with likelihood to get weighted particle set $\{w_t^s \tilde{z}_t^s\}_{s=1}^S$ which represent the posterior at t

Monte Carlo resampling of weighted particle set $\{w_t^s \tilde{z}_t^s\}_{s=1}^S$ to get unweighted posterior set $\{z_t^s\}_{s=1}^S$

Add a bit of noise

Particle Filtering



Particle Filtering

Task 3.5-3.6 of Assignment 3: Study the effect of different motion models

What is next?

Assignment 3 – start with reading the recommended literature (slide 3) to this lecture!

Project – papers will be assigned to groups tonight!

Mon 15 Dec 10:15-12:00 Q31
Exercise 5: Lecture 10 **but in practice, topics of interest to you – post on the webpage what you would like to work on**
Hedvig Kjellström

Tue 16 Dec 08:15-10:00 Q31
Lecture 11: Topic Models
Hedvig Kjellström
Readings: Murphy Chapter 2.3.2, 2.5.4, 10.4.1, 27.1-27.3
If necessary, repeat Murphy Chapter 10!

AdaBoost, tips for Task 3.3

"Linear classifier" is the wrong name for the weak learners. Let us call them linear functions $\phi_m(\mathbf{x}) \equiv \phi(\mathbf{x}, \nu_m)$.

The linear functions ϕ_m are just lines on the surface x=(X,Y), X in [-1,1] Y in [-1,1]. They give functions y=$\phi_m(\mathbf{x})$, $y \in \{-1, +1\}$. On one side of the line, y=+1 and on the other, y=-1. This is the classifier, no svm:s etc are needed. Do not do anything else than what is in Algorithm 16.2!

Using spherical coordinates (which is nice), the parameters for ϕ_m are $\nu_m = (r, \alpha), r \in [-\sqrt{2}, \sqrt{2}], \alpha \in [0, \pi[$.

They are the ones that you should optimize over, when you fit ϕ_m to the weighted data points.

AdaBoost, tips for Task 3.3

Here is a visualization of $\nu_m = (r, \alpha), r \in [-\sqrt{2}, \sqrt{2}], \alpha \in [0, \pi[$

The line itself is perpendicular to the blue vector of length r going out to the line from the origin.

The blue vector has angle α from the positive X axis as in the figure.