HMM (continued)

Sriram Sankararaman

The instructor gratefully acknowledges Fei Sha, Ameet Talwalkar, Kai-Wei Chang, Eric Eaton, Andrew Moore and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Administrivia

- Final exam on Tuesday March 21. Details posted on campuswire.
- Course evaluation open.

Outline

1. Review of last lecture
   - Markov chains
   - Hidden Markov models

Markov Process

Also known as Markov Chain or Markov model

- For a Markov process, the next state depends on the current state :

$$P(x_{t+1}|x_1, \dots, x_t) = P(x_{t+1}|x_t)$$

  Conditioned on the present, the future is independent of the past

  X_{t+1} is independent of $X_i, i < t$ given X_t

- This property implies that

$P(x_3|x_2)$

$$P(x_1, \dots, x_t) = P(x_1)P(x_2|x_1)\dots P(x_t|x_{t-1})$$

Specifying a Markov chain

Initial probability

$$\pi_i = P(X_1 = i)$$

Transition probability

$$q_{ij} = P(X_{t+1} = i | X_t = j)$$

Computing on Markov chains

$x_1, \ldots, x_T$

$P(x_T = i)$

Compute $P(x_T = i)$

Assume uniform probability of starting in each of the states

$$Q = \begin{pmatrix} 0.5 & 0.1 & 0.0 \\ 0.3 & 0.0 & 0.4 \\ 0.2 & 0.9 & 0.6 \end{pmatrix}$$

Q_{11}

Q_{21}

What is the probability of observing a state $X_3 = 3$?

$\Rightarrow P(x_1 = 1, x_2 = 1, x_3 = 3)$

$+$

$\Rightarrow P(x_1 = 1, x_2 = 2, x_3 = 3)$

$+$

$\vdots$

Computing on Markov chains

Compute $P(x_T = i)$

$x_1 \in \{1..k\}$

Assume uniform probability of starting in each of the states

$$\boldsymbol{Q} = \begin{pmatrix} 0.5 & 0.1 & 0.0 \\ 0.3 & 0.0 & 0.4 \\ 0.2 & 0.9 & 0.6 \end{pmatrix}$$

What is the probability of observing a state $X_3 = 3$?

More generally, use the previous computation to sum over all paths of length t that end in the state i

$$P(X_t = i) = \sum_{x_1, x_2, \ldots, x_{t-1}} P(x_1, x_2, \ldots, x_{t-1}, x_t = i)$$

$k \cdot k \ldots k$

K^{t-1}

Computational cost?

$\mathrm{O}(K^{T-1})$: Exponential in T!

Computing on Markov chains

Compute $P(x_T = i)$

Define $p_t(i) = P(X_t = i)$.

$$p_1(i) = \pi_i$$

Assume we already know $p_{t-1}(i)$ for all i.
Use inductive definition to compute $p_t(i)$:

$$\begin{aligned}
p_t(i) &= P(X_t = i) \\
&= \sum_j P(X_{t-1} = j, X_t = i) \\
&= \sum_j P(X_{t-1} = j)P(X_t = i | X_{t-1} = j) \\
&= \sum_j p_{t-1}(j) q_{ij}
\end{aligned}$$

$O(K^{T-1}) \gg O(TK^2)$ $O(K \times K)$

$p_t(i)$ $O(K)$

$t = 1, 2, \ldots, T-1, T$

$t = 1$

$p_1(i) = P(X_1 = i) = \pi_i$

$p_{t-1}(i) = P(X_{t-1} = i)$

$p_t(i) = P(X_t = i)$?

$= \sum_j P(X_t = i, X_{t-1} = j)$

$= \sum_j P(X_{t-1} = j) P(X_t = i | X_{t-1} = j)$

$= \sum_j p_{t-1}(j) q_{ij}$

$O(K)$

$p_{t+1}(i)$

$p_t(j)$

Computing on Markov chains

Compute $P(x_T = i)$

$x_1 \rightarrow x_2 \ldots$

Define $p_t(i) = P(X_t = i)$.

$$p_1(i) = \pi_i$$

$$p_t(i) = \sum_j p_{t-1}(j) q_{ij}$$

Computational cost?

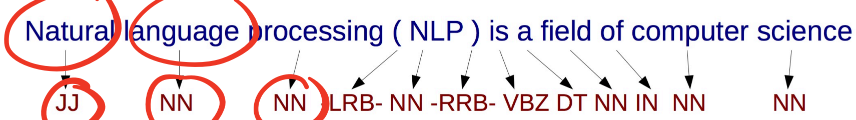
- Computing each $p_t(i)$ for a given t, i takes $O(K)$.
- Computing each $p_t(i)$ for a given t and all i takes $O(K^2)$.
- Computing each $p_t(i)$ for all $t \in \{1, \ldots, T\}$ and all i takes $O(K^2(T-1))$.

Example of Dyanmic Programming

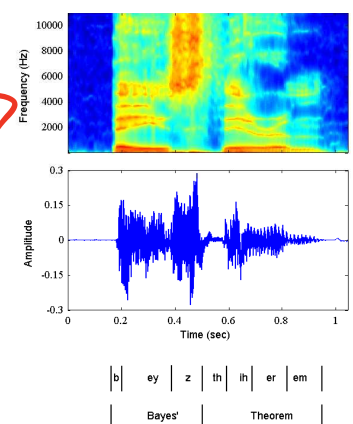
Hidden Markov models

$X_1, \ldots, X_t$

In many applications, we observe only a noisy or indirect measurement of the state X_t



In POS tagging, the state is the part-of-speech which we do not see. Instead, we observe words.



In speech recognition, the state is the word spoken but we only get to see the waveform.

Hidden Markov models

- Previously, in Markov chains, we directly observed the state X_t.
- Now, we observe a new random variable Y_t for each time t that is affected by the state X_t at that time t.
- Can think of Y_t as a noisy version of the true state.

Having observed $(Y_1, \ldots, Y_T)$, we want to ask questions about $X_1, \ldots, X_T$

Hidden Markov models

$P(X_1, .., X_t) = P(X_1)\, P(X_2 | X_1)\, P(X_3 | X_2) \dots P(X_t | X_{t-1})$

- We now define the set of observed states (also called emission symbols) $B = \{1, \dots, L\}$: the set of values that Y_t can take.
- Since X_t is not observed, X_t are called hidden states.

To link up the hidden and the observed states, we have emission probabilities

$$P(Y_t = b | X_t = k) = e_k(b)$$

Constraints: $\sum_b e_k(b) = 1$.

Hidden Markov model

Hidden states: $\{1, \ldots, K\}$
Observed states: $\{1, \ldots, L\}$
Initial probability

$$\pi_i = P(X_1 = i)$$

Transition probability

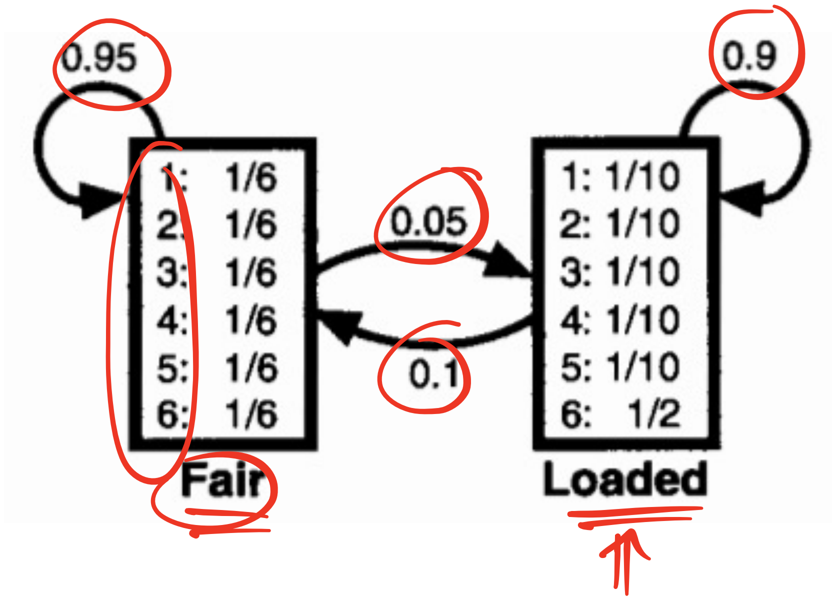
$$q_{ij} = P(X_{t+1} = i | X_t = j)$$

Emission probability

$$e_i(b) = P(Y_t = b | X_t = i)$$

HMM: Example

The occasionally dishonest casino



1

- Hidden State: Is the casino using the fair or unfair die?
- Observed State: Roll of die ($\{1, \ldots, 6\}$)

Durbin et al. Biological Sequence Analysis

Querying an HMM

We observe $(Y_1, \ldots, Y_5) = (2, 6, 6, 1, 3)$. Can we infer for which of the throws the casino used the unfair die ?

$X_1, \ldots, X_5$

HMM

$x = x_1, \ldots, x_T = x_{1:T}$
$y = y_1, \ldots, y_T = y_{1:T}$

The joint probability of the sequence of hidden states and the observed states

$$
\begin{aligned}
P(\boldsymbol{y}, \boldsymbol{x}) &= P(y_{1:T}, x_{1:T}) \\
&= P(y_{1:T} | x_{1:T}) P(x_{1:T}) \\
&= \prod_{t=1}^{T} P(y_t | x_t) \underbrace{P(x_{1:T})}_{\text{Markov chain}} \\
&= \prod_{t=1}^{T} e_{x_t}(y_t) \pi_{x_1} \prod_{t=1}^{T-1} q_{x_{t+1} x_t}
\end{aligned}
$$

Easy to compute but not very useful because we don't know the hidden states.

HMM: The most probable path problem

Given a sequence of observations $(y_1, \ldots, y_T)$, what is the most probable sequence of hidden states $(x_1, \ldots, x_T)$?

$$\arg\max_{x_{1:T}} P(y_{1:T}, x_{1:T})$$

- Often called the decoding problem.
- One way to solve this problem is to search over all possible values of $(x_{1:T})$.
- There are exponentially many of them ($\mathrm{O}(K^T)$)
- Fortunately, it turns out there is an efficient dynamic programming algorithm to solve this problem.

The Viterbi algorithm

MPP

$\arg\max_{x_{1:T}} P(y_{1:T}, x_{1:T})$

$t = 1, 2, .., T$

$v_1(k) \rightarrow v_2(k) \rightarrow \dots v_T(k)$

A recursive algorithm

- Suppose we have computed the the probability $v_t(k)$ for all values of $t \in \{1, \dots, T\}$ and $k \in \{1, \dots, K\}$:
  The probability of the most probable path (MPP) for observations $(y_1, \dots, y_t)$ (so that path has length t) that ends up in state k.

$$v_t(k) = \max_{x_{1:t-1}} P(y_{1:t}, x_{1:t-1}, x_t = k)$$

$t=1 \rightarrow t=2 \rightarrow \dots \; t=T$

Why is this a useful quantity?

- If we look at $v_T(k)$, it tells us the probability of the MPP of length T that ends in state k.
- The answer to the MPP problem then is $\max_k v_T(k)$!

The Viterbi algorithm

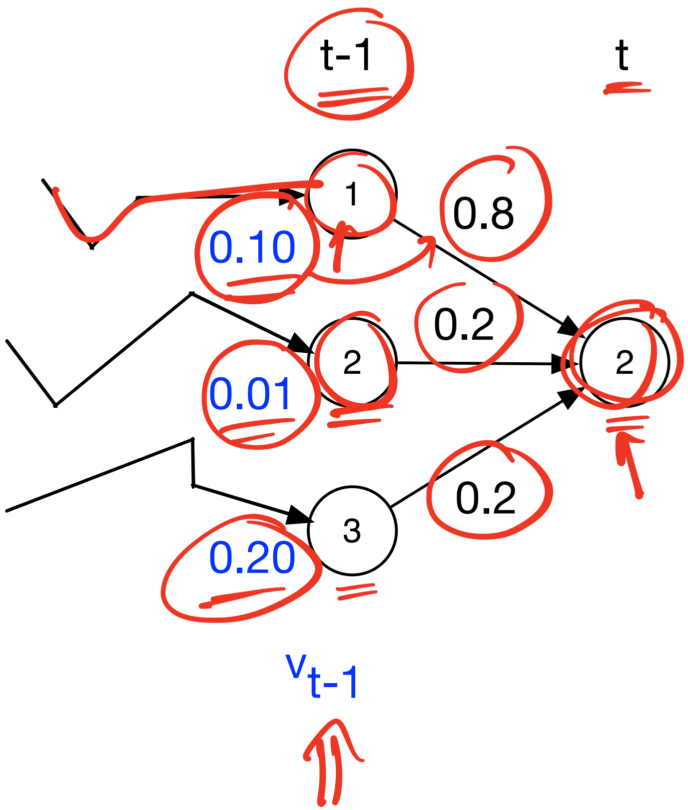
Can we compute $v_t(k)$ efficiently?

Assume we have computed $v_{t-1}(l)$.

We have computed the probability of the MPP of length $t-1$ that ends in state l for all values of l

How do we use this to compute the probability of MPP of length t that ends in state k?

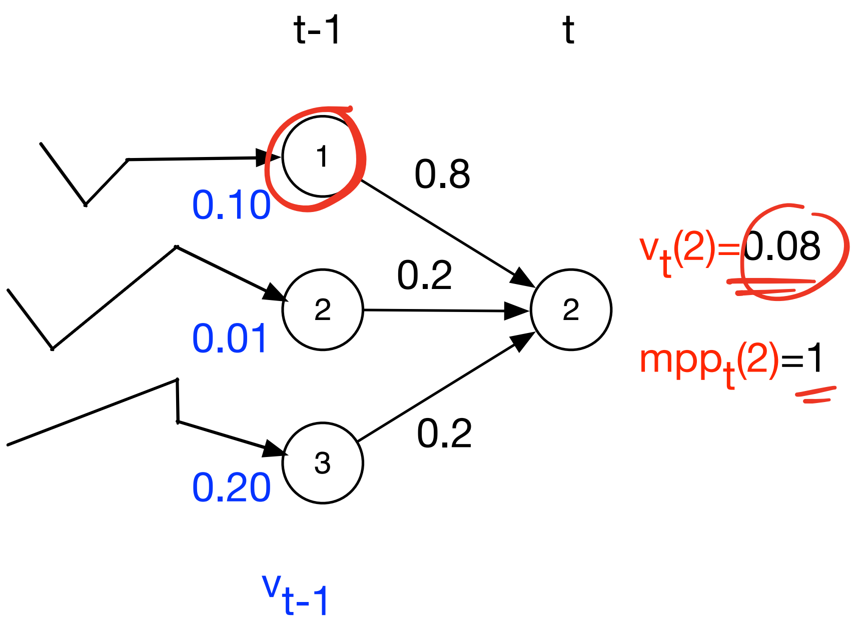
Viterbi example



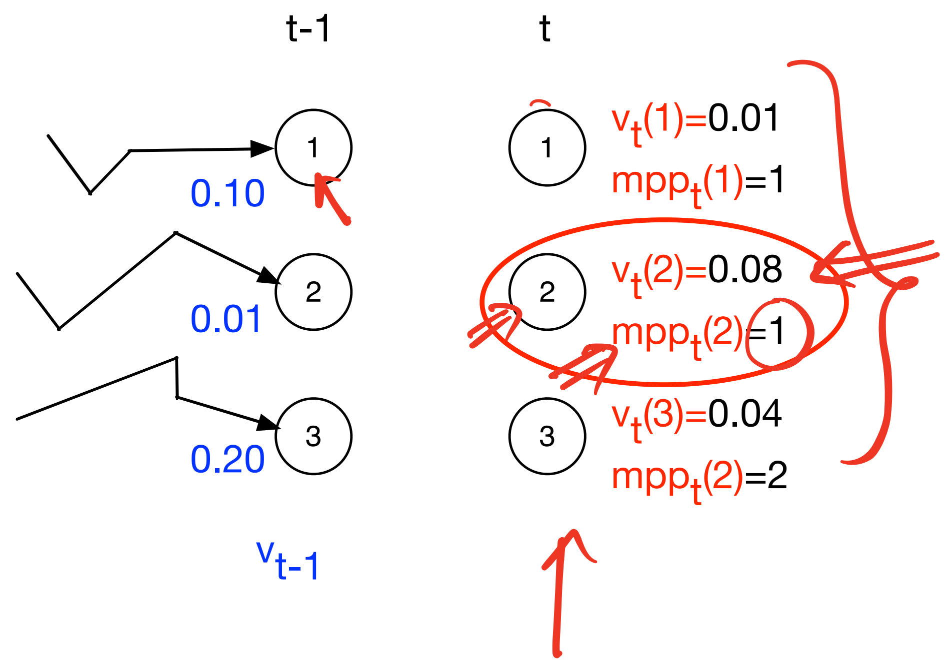
$v_t(2)$=max(0.8*0.10,
~~0.2*0.01,~~
0.20*0.2)

$mpp_t(2)$=argmax(0.8*0.10,
0.2*0.01,
0.20*0.2

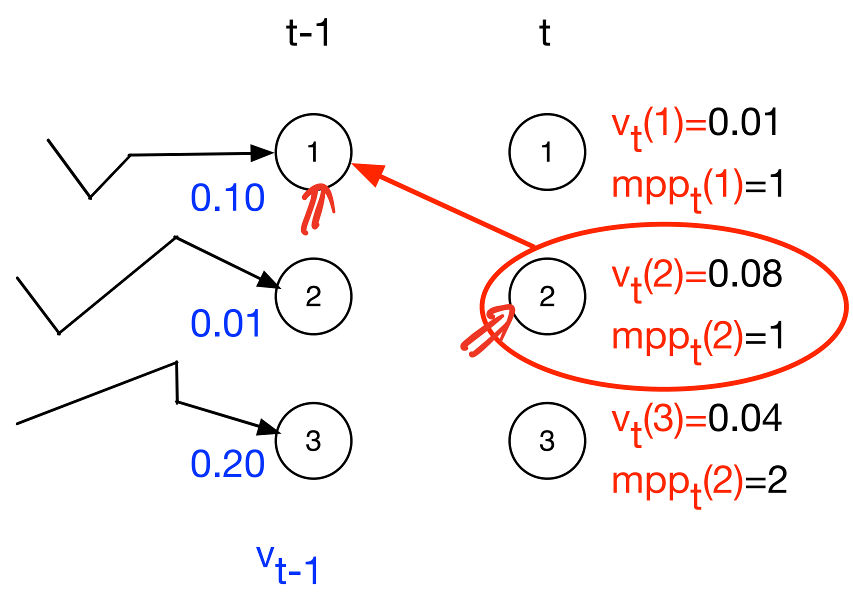
Viterbi example



Viterbi example



Viterbi example



MPP for observations upto time t (backwards):

$(2, 1, mpp_{t-1}(1), mpp_{t-2}(mpp_{t-1}(1)), \ldots)$

In other words, state 2 at time t, state 1 at time $t-1$,...

The Viterbi algorithm

Can we compute $v_t(k)$ efficiently?

$v_t(k) =$ $v_1(k) = P(y_1, x_1 = k)$

$= P(y_1 | x_1 = k) \, P(x_1 = k)$

Begin with $t = 1$

$$
\begin{aligned}
v_1(k) &= P(y_1, x_1 = k) \\
&= P(y_1 | x_1 = k) P(x_1 = k) \\
&= e_k(y_1) \pi_k
\end{aligned}
$$

The Viterbi algorithm

Can we compute $v_t(k)$ efficiently?

Assume we have computed $v_{t-1}(l)$.

We have computed the probability of the MPP of length $t-1$ that ends in state l for all values of l

How do we use this to compute the probability of MPP of length t that ends in state k?

The Viterbi algorithm

Can we compute $v_t(k)$ efficiently?

The most probable path with last two states (l, k) is the most probable path with state l at time $t-1$ followed by a transition from state l to state k and emitting the observation at time t.

What is the probability of this path?

$$v_{t-1}(l)P(X_t = k | X_{t-1} = l)P(y_t | X_t = k)$$
$$= v_{t-1}(l)q_{lk}e_{k}(y_t)$$

So the most probable path that ends in state k at time t is obtained by maximizing over all possible states l in the previous time $t-1$.

$$v_t(k) = \max_l v_{t-1}(l)q_{kl}e_t(y_t)$$

$O(K^2 T)$

Also keep a pointer to the state that lead to the current state

$$mpp_t(k) = l^*$$
$$l^* = \arg\max_l v_{t-1}(l)q_{kl}e_t(y_t)$$

The Viterbi algorithm

Can we compute $v_t(k)$ efficiently?

Continue till we compute $v_T(k), k \in \{1, \dots, K\}$. Let:

$$k^* = \arg\max_k v_T(k)$$

To obtain the MPP, follow the pointers defined by $mpp_t(k)$.

The Viterbi algorithm

Can we compute $v_t(k)$ efficiently?

$P(X_t = k \mid X_{t-1} = l)$

$$v_t(k) = \max_l v_{t-1}(l) q_{kl} e_t(y_t)$$

- The cost of computing this is O(K) for a given k.
- The total cost of computing this is O(K^2) for all k.
- Total cost of computing $v_T(k)$ is O(TK^2).

Other HMM computations

$P(X_2 = k \mid y_1, \ldots, y_5)$ $T = 5$

$y_1, \ldots, y_5$

$\neq P(X_2 = k \mid y_1, y_2)$

Given a sequence of observations $(y_1, \ldots, y_T)$, what is the probability that state at time t is k ?

$$P(X_t = k | y_{1:T})$$

$\neq P(X_t = k \mid y_{1:t})$

Given a sequence of observations $(y_1, \ldots, y_T)$, what is the probability of the observations ?

$$P(y_{1:T})$$

Can also be computed efficiently using dynamic programming.

Learning HMMs

We assume the parameters are known. Can we learn parameters from data?

Parameters of the HMM

$$\boldsymbol{\theta} = (\underline{\underline{\boldsymbol{\pi}}}, \underline{\underline{\boldsymbol{Q}}}, \underline{\underline{\boldsymbol{E}}})$$

Here $\boldsymbol{E}$ is the matrix of emission probabilities. $E_{kb} = e_k(b)$.

Given training data of observed states $(\underline{\underline{y_{1:T}}})$, find parameters $\boldsymbol{\theta}$ that maximizes the log likelihood.

Learning HMMs

We assume the parameters are known. Can we learn parameters from data?

Our model contains observed and unobserved random variables and hence is incomplete.

- Observed: $\mathcal{D} = y_{1:T}$
- Unobserved (hidden): $x_{1:T}$

$$
\begin{aligned}
\widehat{\boldsymbol{\theta}} &= \arg\max_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) \\
&= \arg\max_{\boldsymbol{\theta}} \log P(y_{1:T}|\boldsymbol{\theta}) \\
&= \arg\max_{\boldsymbol{\theta}} \log \sum_{x_{1:T}} P(y_{1:T}, x_{1:T}|\boldsymbol{\theta})
\end{aligned}
$$

The objective function $\ell(\boldsymbol{\theta})$ is called the incomplete log likelihood.

We can optimize this function using the EM algorithm (we won't get into the details in this course).

Summary

HMM

- Allows us to model dependencies.
- Can perform computations efficiently using dynamic programming.
- Can learn the parameters using the EM algorithm.

Summary of the course

Types of learning problems

- Supervised, unsupervised and reinforcement
- Labeled vs unlabeled data
- Labeled: Supervised learning. Type of label: classification (categorical) vs regression (quantitative)
- Unlabeled: Unsupervised learning.

Summary of the course

Supervised learning

- Model/hypotheses
- Loss function
- Regularizer
- Algorithm to solve optimization problem

Summary of the course

Supervised learning

- Key goal is to pick hypothesis h that minimizes risk for some loss function:

$$\mathcal{R}[h(\boldsymbol{x})] = \sum_{\boldsymbol{x},y} \ell(h(\boldsymbol{x}), y) p(\boldsymbol{x}, y)$$

  Difficulty: we don't know the data generating distribution $p(\boldsymbol{x}, y)$.

- Instead pick h that minimizes empirical risk (a.k.a training error)

$$\mathcal{R}^{\text{EMP}}[h(\boldsymbol{x})] = \frac{1}{N} \sum_{n} \ell(h(\boldsymbol{x}_n), y_n)$$

Summary of the course

Supervised learning

- Setup: given a training dataset $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$, learn a function $h(\boldsymbol{x})$ to predict y given $\boldsymbol{x}$.
  - Choose hypothesis space/models.
  - Define a loss function.
  - Define a cost function (typically loss function evaluated over the training data + regularizer).
  - Algorithm to solve optimization problem.

Summary of the course

Supervised learning

- Hypotheses
  - Decision trees, Nearest neighbors
  - Linear models: $h(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x}$
  - Kernels to extend to non-linear functions.
  - Neural Networks: jointly learn ϕ and $\boldsymbol{w}$.
  - Ensembles as a way to combine classifiers.
- Loss functions
  - Squared loss: least squares for regression
  - 0-1 loss for binary classification and surrogates for 0-1 loss.
  - Logistic loss, Exponential loss, Hinge loss

    Logistic Regression Adaboost SVM
- Main principles
  - Many of these learning algorithms can be thought of as solving the problem of finding "good" parameters for some probabilistic model.
  - Principles for finding good parameters: Maximum likelihood, regularize likelihood
  - Generative vs discriminative models.

Summary of the course

Supervised learning

- Optimization
  - Convex vs non-convex optimization problems
  - Methods: gradient descent (batch vs stochastic)
  - Constrained optimization. Lagrange function. Primal vs dual formulations.
- Concepts
  - Training error vs generalization error
  - Overfitting vs underfitting
  - The role of inductive bias
- Practical issues
  - How to tune hyperparameters, how to estimate generalization error
  - Importance of train-validation-test setup and cross-validation

Unsupervised learning

- Finding structure in data.
- Dimensionality reduction, Clustering and mixture models, Modeling dependencies
- Clustering
  - K-means. Requires solving a non-convex problem
  - Can be viewed as a probabilistic model with hidden variable (GMM)
  - EM algorithm: iterative algorithm to estimate MLE
- PCA
  - Linear Dimensionality reduction
  - Finds projections that maximize variance, minimize reconstruction error
  - Obtained by computing the top eigenvectors
- Hidden Markov Models (HMM)
  - Model dependency among observations
  - Use dynamic programming to efficiently query the HMM

- Thank you for your participation!
- All the best for the exam!