Final Exam Review

CM146 Winter 2018

March 16, 2018

Outline

Computational Learning Theory

- Set up:
  - Instance Space: X, the set of examples
  - Concept Space: C, the set of possible target functions: $f \in \mathcal{C}$ is the hidden target function
  - Hypothesis Space: H, the set of possible hypotheses. This is the set that the learning algorithm explores
  - What we want: A hypothesis $h \in H$ such that $h(x) = f(x)$
- Given a distribution D over examples, the **error of a hypothesis** h with respect to a target concept f is

$$err_D(h) = Pr_{x \sim D}[h(x) \neq h(f)]$$

- For a target concept f, the **empirical error** of a hypothesis h is defined for a training set S as the fraction of examples x in S for which the functions f and h disagree, denoted by $err_S(h)$.
- Overfitting: When the empirical error on the training set $err_S(h)$ is substantially lower than $err_D(h)$.

Probably Approximately Correct (PAC) Learning

$$m > \frac{n}{\epsilon}\left(\log(n) + \log\left(\frac{1}{\delta}\right)\right)$$

- The concept class C is **PAC learnable** by L using H if for $f \in C$, for all distribution D over X, and fixed $\epsilon > 0, \delta < 1$, given m examples sampled i.i.d. according to D, the algorithm L produces, with probability at least $(1 - \delta)$, a hypothesis $h \in H$ that has error at most ϵ, where m is polynomial in $1/\epsilon$, $1/\delta$, n and size H.
- The concept class C is **efficiently learnable** if L can produce the hypothesis in time that is polynomial in $1/\delta$,$1/\delta$ and size(H).

Shattering and VC dimension

- Definition: A set S of examples is **shattered** by a set of functions H if for every partition of the examples in S into positive and negative examples there is a function in H that gives exactly these labels to the examples.
- The **VC dimension** of hypothesis space H over instance space X is the size of the largest finite subset of X that is shattered by H.
- When there are infinite number of hypotheses in H, VC dimension is used to represent the expressiveness of a hypothesis space.

Kernel Perceptron

- Perceptron:

$$\begin{aligned} &\text{if } y_i(\mathbf{w}^T\mathbf{x}_i) \leq 0 \\ &\quad \mathbf{w} \leftarrow \mathbf{w} + y_i\mathbf{x}_i \\ \Rightarrow &\mathbf{w}^* = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i \end{aligned}$$

Kernel Perceptron:

$$\begin{aligned} &\text{if } y_i(\mathbf{w}^T\Phi(\mathbf{x}_i)) \leq 0 \\ &\quad \mathbf{w} \leftarrow \mathbf{w} + y_i\Phi(\mathbf{x}_i) \\ \Rightarrow &\mathbf{w}^* = \sum_{i=1}^{m} \alpha_i y_i \Phi(\mathbf{x}_i) \end{aligned} \tag{1}$$

- In prediction:

$$\text{perceptron prediction } sgn(\mathbf{w}^T\mathbf{x}^{test}) = sgn(\sum_i \alpha_i y_i \mathbf{x}_i^T \mathbf{x}^{test})$$

$$\text{kernel perceptron prediction } sgn(\mathbf{w}^T\Phi(\mathbf{x}^{test})) = sgn(\sum_i \alpha_i y_i \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}^{test})) \tag{2}$$

where α_i is the number of mistakes made on x_i.

Kernel

- Kernel Trick: Save time/space by computing the value of $K(\mathbf{x}, \mathbf{z})$ by performing operations in the original space (without a feature transformation!)

$$K(\mathbf{x}, \mathbf{z}) = \Phi(\mathbf{x})^T \Phi(\mathbf{z})$$

  So prediction with this high dimensional lifting map is

$$sgn(\mathbf{w}^T \Phi(\mathbf{x}^{test})) = sgn(\sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}^{test}))$$

- Positive semi-definite
- Symmetric
- Find the corresponding feature map of a kernel
- Prove the kernel is valid (How to?)

Support Vector Machine

- Intuition: We want $\max_w \gamma$, where margin

$$\gamma = \min_{\mathbf{x}_i, y_i} \frac{y_i(\mathbf{w}^T \mathbf{x}_i + b)}{||\mathbf{w}||}$$

- Hard SVM:

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} \\ \text{s.t.} \quad & \forall i, y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \end{aligned} \tag{3}$$

- Soft SVM:

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \xi_i \\ \text{s.t.} \quad & \forall i, y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned} \tag{4}$$

or equivalently,

$$\min_{\mathbf{w}, b} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)) \tag{5}$$

Support Vector Machine

- Hinge Loss: $L_{hinge}(y, \mathbf{x}, \mathbf{w}) = \max(0, 1 - y\mathbf{w}^T\mathbf{x})$
- The effect of hyperparameter C.
- Solving the SVM optimization problem: Gradient descent (still very slow) $\rightarrow$ Stochastic gradient descent

$$J^t(w) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))$$

$$\nabla J^t = \begin{cases} w & \text{if } \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)) = 0 \\ w - Cy_i\mathbf{x}_i & \text{otherwise} \end{cases} \tag{6}$$

Kernel Support Vector Machine

- Dual (Kernel) SVM

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - \mathbf{e}^T\boldsymbol{\alpha} \\ \text{s.t.} \quad & \forall i, 0 \leq \alpha_i \leq C \\ & \mathbf{y}^T\boldsymbol{\alpha} = 0 \end{aligned} \tag{7}$$

  where $Q_{ij} = y_i y_j \phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$ and $\mathbf{e} = [1, \cdots, 1]^T$

- Dual Representation: In the optimum, the solutions of the primal and dual problem have the following relation

$$w^* = \sum_i \alpha_i^* y_i \Phi(\mathbf{x}_i)$$

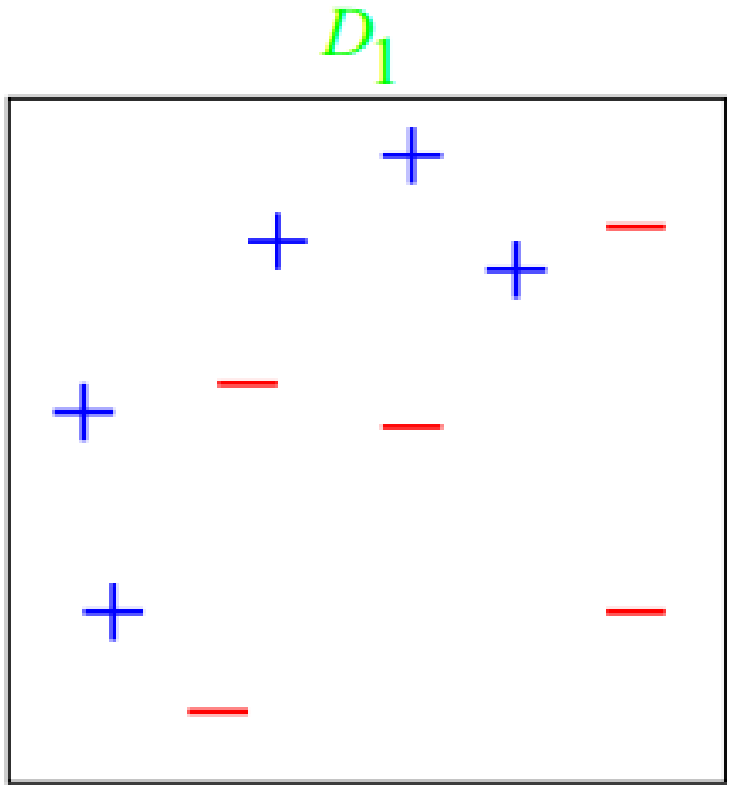
- Decision function:

$$sgn(\sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}^{test}) + b)$$

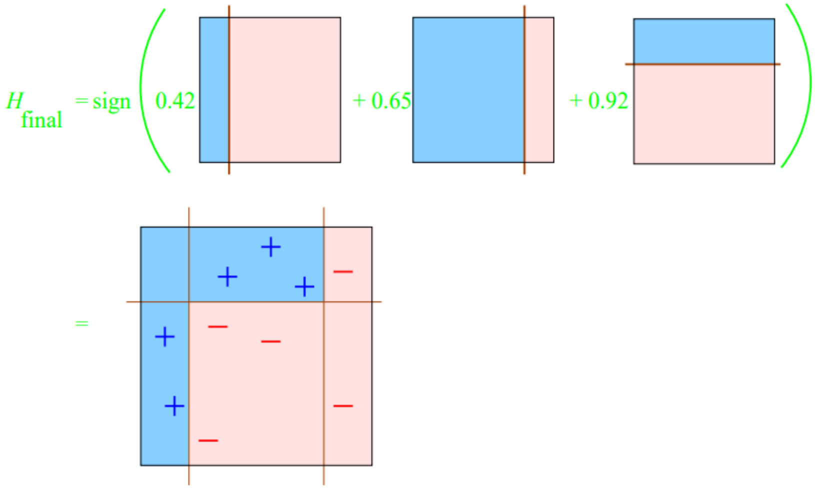
- $\alpha_i = 0 \Rightarrow$the training sample doesnt affect the prediction
  $\alpha_i > 0 \Rightarrow$ support vectors: only SVs determine the weight!

Boosting

- Intuition - finding many rough rules of thumb can be a lot easier than finding a single, highly accurate classifier.



Boosting



Boosting

- Define a strong classifier $H(x) = sgn\left[\sum_{t=1}^{T} \alpha_t h_t(x)\right]$
- $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
- Relation between α and ϵ
- Overtraining vs overfitting
- Adaboost can fail if -
  - The weak classifiers are too complex and overfit.
  - The weak classifiers are too weak, essentially underfitting.

Clustering

Goal

Find an underlying distribution or organization in the data.

Setup: Given a dataset $D = \{x_n\}_{n=1}^{N}$ and k we want to output

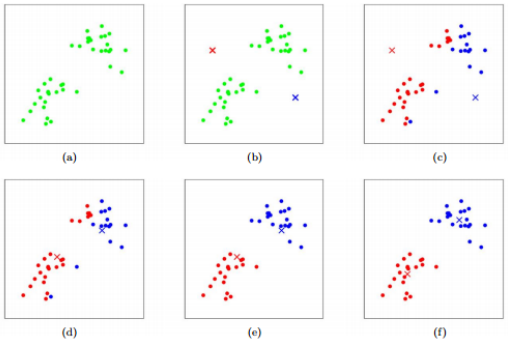
- $\{\mu_k\}_{k=1}^{K}$ prototypes or centroids
- $A(x_n) \in \{1, 2, ..., K\}$: the cluster membership, i.e. cluster assigned to x_n

Distance measures (recap):

L2 Norm: $||x_1 - x_2||_2 = \sqrt{\sum_{i=1}^{n}(x_{1,i} - x_{2,i})^2}$

L1 Norm $||x_1 - x_2||_1 = \sqrt{\sum_{i=1}^{n}|x_{1,i} - x_{2,i}|}$

K-Means



Algorithm

Given a training set $D = \{x_n\}$ group the data in K clusters each represented by centroids.

1. cluster centroids $\mu_1, \mu_2, ..., \mu_k \in \Re^n$
2. for every datapoint commpute the closest centroid $c_j := \arg\min_j ||x_i - \mu_{uj}||_2$
3. update $u_j := \frac{\sum_i^m I\{c_i=j\}x_i}{\sum_i^m \{c_i=j\}}$

Bayesian Learning

Goal

To find the best hypothesis from some space H of hypotheses, using the observed data D.

What does best mean?

- Bayesian learning uses $P(h|D)$, the conditional probability of a hypothesis given the data, to define best.
- Calculate posterior $P(h|D)$ using Bayes theorem.

Maximum a Posteriori hypothesis, h_{MAP} (recap):

- $h_{MAP} = argmax_{h \in H} P(h|D) = argmax_{h \in H} P(D|h)P(h)$.
- If we assume that the prior is uniform, we get Maximum Likelihood hypothesis, $h_{ML} = argmax_{h \in H} P(D|h)$.

Expectation Maximization

Goal

To compute MLE in unsupervised settings.

Setup: We only have observation of $\widetilde{P}(X)$

- In generative model, we have $P(X, Y|\theta)$
- We know $P(X|\theta) = \sum_Y P(X, Y|\theta)$
- Therefore, MLE is $argmax_\theta P(X|\theta) = argmax_\theta \sum_Y P(X, Y|\theta)$

EM algorithm (recap):

- Solves $argmax_\theta \sum_Y P(X, Y|\theta)$ by iteratively updating θ
- In general, known to converge to a local maximum of the maximum likelihood function

EM algorithm

Given

A statistical model which generates a set X of observed data, a set of unobserved latent data or missing values Z, and a vector of unknown parameters θ, along with a likelihood function $L(\theta; X, Z) = P(X, Z|\theta)$.

- First, initialize the parameters θ to some random values.
- Compute the probability of each possible value of Z, given θ.
- Then, use the just-computed values of Z to compute a better estimate for the parameters θ.
- Iterate steps 2 and 3 until convergence.

Naive Bayes

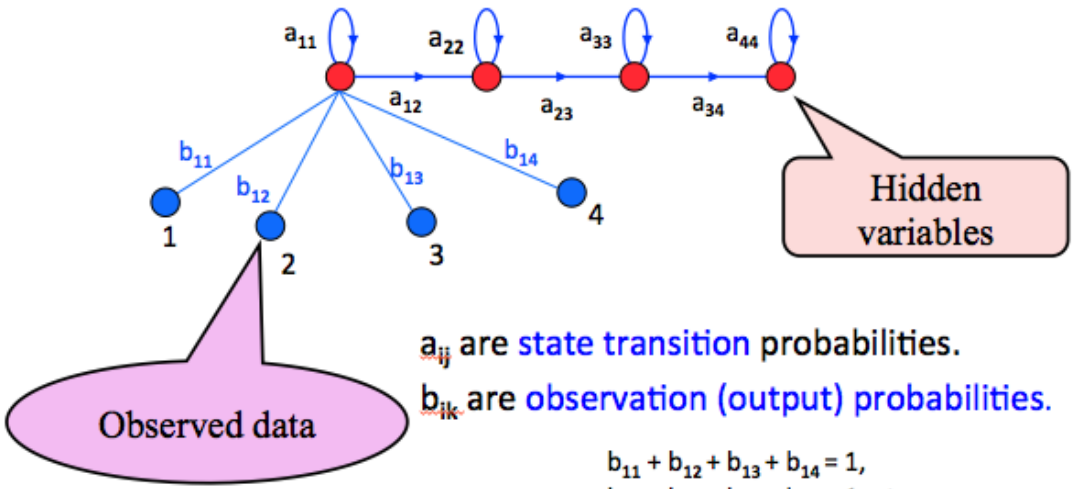
- Bayes classification
  $P(c|x) \propto P(x|c)P(c) = P(x_1, \cdots, x_n|c)P(c) for c = c_1, \cdots, c_L$
  Difficulty: learning the joint probability $P(x_1, \cdots, x_n|c)P(c)$ is infeasible!
- Naive Bayes classification
  Assume all input features are class conditionally independent!
  $P(x_1, x_2, \cdots, x_n|c) = P(x_1|x_2, \cdots, x_n, c)P(x_2, \cdots, x_n|c)$
  $= P(x_1|c)P(x_2, \cdots, x_n|c)$
  $= P(x_1|c)P(x_2|c)(x_n|c)$
- Apply the MAP classification rule

Naive Bayes

- Learning phase
  Given a training set S of F features and L classes,
  For each target value of $c_i (c_i = c_1, \cdots, c_L)$
  $\hat{P}(c_i) = estimateP(c_i) with examples in S$;
  $for every feature value x_j k of each feature x_j$;
  $\hat{P}(x_j = x_j k | c_i) = estimateP(x_j k | c_i) with examples in S$
- Test phase
  given an unknown instance $x' = (a'_1, \cdots, a'_n)$
  Look up tables to assign the label c^* to X' if
  $[\hat{P}(a'_1|c^*) \cdots \hat{P}(a'_n|c^*)]\hat{P}(c^*) > [\hat{P}(a'_1|c_i) \cdots \hat{P}(a'_n|c_i)]\hat{P}(c_i)$
  $c_i \neq c^*, c_i = c_1, \cdots, c_L$

Hidden Markov Models

- Markov Models
  - N distinct states
  - Begins in some initial state(s)
  - At each time step, the system moves from current to next state according to transition probabilities associated with current state
  - This model is a discrete (finite) system
- **Hidden** Markov Models
  - We can not observe state



Hidden Markov Models

- Set of states: $\{s_1, s_2, \cdots, s_N\}$
- Process moves from one state to another generating a sequence of states: $s_{i1}, s_{i2}, , s_{ik},$
- Markov chain property: probability of each subsequent state depends only on what was the previous state:
  $P(s_{ik}|s_{i1}, s_{i2}, , s_{ik-1}) = P(s_{ik}|s_{ik-1})$
- States are not visible, but each state randomly generates one of M observations (or visible states) $\{v_1, v_2, , v_M\}$
- To define hidden Markov model, the following probabilities have to be specified:
  - matrix of transition probabilities $A = (a_{ij})$, $a_ij = P(s_i|s_j)$
  - matrix of observation probabilities $B = (b_i(v_m))$, $b_i(v_m) = P(v_m|s_i)$
  - a vector of initial probabilities $\pi = (\pi_i)$, $\pi_i = P(s_i)$
  - Then, model is represented by $M = (A, B, \pi)$

Thank you!