CSC311 Study Guide

Lec 6: SVMs and Ensembles

Hinge Loss

$$\mathcal{L}_H(z,t) = max\{0, 1 - zt\}$$

Intuition: when $y < 1$, penalize more. Otherwise, loss $= 0$.

SVMs

- Idea: Attempt to find a hyperplane with maximum distance that separates two classes.
- Given labeled training data, output an classifier that is the optimal class separator that can categorize new examples.
- Uses **Hinge loss**

Ensembles

Bagging

- Train classifiers on randomly independently selected subsamples of the training data.
- Reduce variance

Boosting

- Reduces bias by generating an ensemble of weak classifiers
- Each classifer is trained to reduce error from previous example
- increase variance

Weighted Training Set

Some training sets are given more weights than others, e.g., used to emphasize training on a type of mistake, usually with $w^{(n)} > 0, \sum_{n=1}^{N} w^{(n)} = 1$.

$$\frac{1}{N}\sum_{n=1}^{N} \mathbb{I}[h(x^{(n)}) \neq t^{(n)}] \text{ vs. } \frac{1}{N}\sum_{n=1}^{N} \mathbb{I}[w^{(n)} h(x^{(n)}) \neq t^{(n)}]$$

Weak Learner/Classifier

An efficient learning algorithm that outputs predictions using decision stump that are slightly better than random.

Adaptive Boosting (AdaBoost)

```
1. Initialize weights to be 1/n
2. Using weighted data, fit weak classifiers and select the one with min error
3. Compute weighted error
4. Update data weights: weak classifiers with lower weighted error get more weight
in the final classifier
```

Lec 7: Probabilistic Models

$p(t \mid \mathbf{x})$ - estimate parameters of decision boundary separator directly from labeled examples

$p(\mathbf{x} \mid t)$ - model the distribution of inputs and get what each class look like

Naive Bayes

Assumes x_i and x_j are independent given the class c

$$p(c, x_1, \ldots, x_D) = p(c)p(x_1 \mid c) \ldots p(x_D \mid c)$$

1. Train: estimate parameters using maximum likelihood
2. Test: apply Bayes' Rule

Issues

May overfit if data is too little

Gaussian Discriminative Analysis

A generative model that makes strong modeling assumption that class-conditional data is multivariate Gaussian

Lec 8: Principal Component Analysis

Want to map the data to a lower dimensional space to

- Save computation / memory
- Reduce overfitting, achieve better generalization

- visualize better

Goal: find a K-dimensional subspace $\mathcal{S} \subset \mathbb{R}^D$ such that $\mathbf{x}^{(n)} - \hat{\mu}$ is well represented by its projection onto a K-dimensional $\mathcal{S}$.

- Need to project data onto a subspace that:
  - Minimize reconstruction error
  - Maximize the variance of reconstructions
- Maximizing the variance is equivalent to minimizing the reconstruction error

Principal Components

The optimal PCA subspace is spanned by the top K eigenvectors of $\hat{\Sigma}$. These eigenvectors are called principal components, just choose the first K of any orthonormal eigenbasis for $\hat{\Sigma}$.

Autoencoders

Encoder: input $\Rightarrow$ some linear operations $\Rightarrow$ low-dimension representation

Decoder: low-dimension representation $\Rightarrow$ some linear operations $\Rightarrow$ reconstructed input

The best possible K-dimensional subspace that minimizes reconstruction error is the PCA subspace. So the linear autoencoder are just the principal components.

Lec 9: K-Means and EM Algorithm

Clustering: grouping data points into clusters with no observed labels

K-Means

Find cluster centers $\{\mathbf{m}_k\}_{k=1}^K$ and assignments $\{\mathbf{r}^{(n)}\}_{n=1}^N$ that minimize the sum of squared distances of data points $\{\mathbf{x}^{(n)}\}$ to their assigned cluster centers.

We can:

- Fix the assignments $\{\mathbf{r}^{(n)}\}$ and find optimal centers $\{\mathbf{m}_k\}$
- Fix the assignments $\{\mathbf{m}_k\}$ and find optimal centers $\{\mathbf{r}^{(n)}\}$

Algorithm

```
1. Intialize cluster centers randomly
2. Iteratively alternates
    1) Assignment: assign each data point to the closest cluster
    2) Refitting: move each cluster center to the mean of the data assigned to it
```

Soft K-Means

Each data point is given soft degree of assignments to each cluster mean, based on responsibilities

Gaussian Mixture Model

π_k captures the relative proportion of each cluster in the dataset.

Prior: without observing the image content, what is the probability that image i is from cluster k?

$$p(z_i = k) = \pi_k$$

Likelihood: given observation of $\mathbf{x}_i$ is from cluster k, what is the likelihood of seeing $\mathbf{x}_i$?

$$p(x_i \mid z_i = k, \mu_k, \Sigma_k) = \mathcal{N}(x_i \mid \mu_k, \Sigma_k)$$

Expectation-Maximization algorithm

1. E-step: given our current model how much do we think a cluster is responsible for generating a datapoint
   - Compute $\mathbb{E}\Big[\mathbb{I}[z^{(n)} = k \mid \mathbf{x}^{(n)}; \pi_k, \mu_k]\Big]$
2. M-step: update π_k, μ_k of each Gaussian to maximize the probability that it would generate the data it is currently responsible for

Lec 10: Matrix Factorizations & Recommender Systems

PCA with K principal components finds the optimal rank-K approximation of $\mathbf{X} \in \mathbb{R}^{N \times D}$ using two smaller matrices $\mathbf{U} \in D \times K$ and $\mathbf{Z} \in N \times K$.

$$\min \|\mathbf{X}^T - \mathbf{U}\mathbf{Z}^T\|_F^2$$

Matrix Completion

$\mathbf{X}$ is partially observed, wants to fill in missing values.

Alternating Least Squares

$$\min_{\mathbf{U},\mathbf{Z}} \frac{1}{2} \sum_{(n,m)\in O} (R_{nm} - \mathbf{u}_n^T \mathbf{z}_m)^2$$

fix $\mathbf{Z}$ and optimize $\mathbf{U}$, followed by fix $\mathbf{U}$ and optimize $\mathbf{Z}$ until convergence.

Gradient descent

Minimize $f(\mathbf{U}, \mathbf{Z})$ treating both $\mathbf{U}, \mathbf{Z}$ as variables.

$$\begin{bmatrix} \mathbf{U} \\ \mathbf{Z} \end{bmatrix} \leftarrow \begin{bmatrix} \mathbf{U} \\ \mathbf{Z} \end{bmatrix} - \alpha \nabla f(\mathbf{U}, \mathbf{Z})$$

This is expensive.

Stochastic gradient descent

Randomly select n, m in $\mathbf{R}$ to update $\mathbf{u}_n$ and $\mathbf{z}_m$ attempting to minimize $\frac{1}{2} \sum_{(n,m)\in O} (R_{nm} - \mathbf{u}_n^T \mathbf{z}_m)^2$

$$\begin{bmatrix} \mathbf{u}_n \\ \mathbf{z}_m \end{bmatrix} \leftarrow \begin{bmatrix} \mathbf{u}_n \\ \mathbf{z}_m \end{bmatrix} - \alpha \begin{bmatrix} (R_{nm} - \mathbf{u}_n^T \mathbf{z}_m)\mathbf{z}_m \\ (R_{nm} - \mathbf{u}_n^T \mathbf{z}_m)\mathbf{u}_n \end{bmatrix}$$

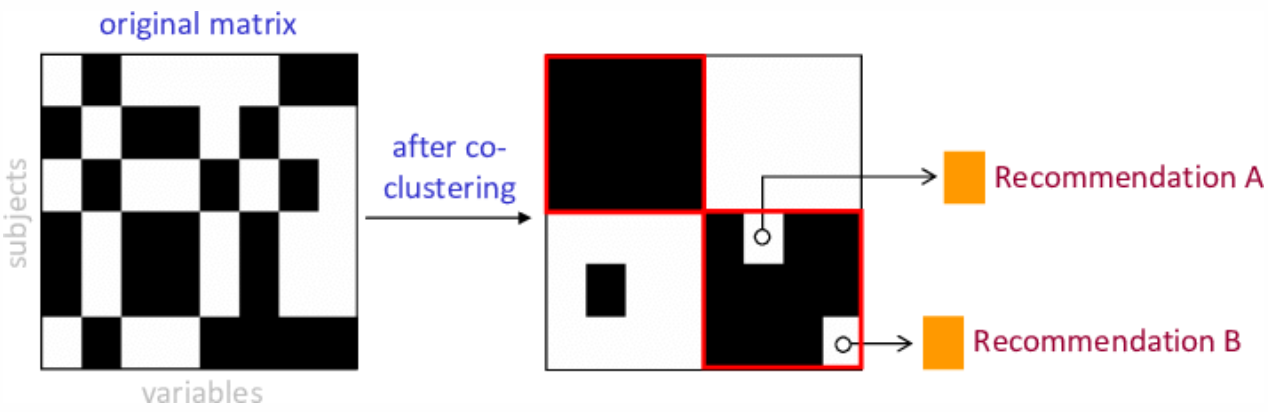
Other models

K-Means

View K-Means as a mmatrix factorization.

1. Stack assignments $\mathbf{r}_i$ into $\mathbf{R}$, and stack cluster centers $\mathbf{m}_k$ into $\mathbf{M}$.
2. Reconstruction of data given by $\mathbf{RM}$

Co-clustering



Sparse Coding

$$\min_{\mathbf{s}} \|\mathbf{x} - \mathbf{A}\mathbf{s}\|^2 + \beta\|\mathbf{s}\|_1$$

Learn a dictionary, uses β to trade off reconstruction error vs. sparsity.

Lec 11: Reinforcement Learning

How should the agent choose its actions so that its long-term rewards are maximized?

Policy

$\pi \leftarrow$ the action selection mechanism that maps from states to actions

- deterministic policy: $A_t = \pi(S_t)$
- stochastic policy: $A_t \sim \pi(\cdot \mid S_t)$

Value function

$V^\pi \leftarrow$ ***state-value*** function, the expected discounted reward if the agent starts from state s and follows policy π

$$V^\pi = \mathbb{E}_\pi\Big[\sum_{t\geq 0}\gamma^t R_t \mid S_0 = s\Big]$$

$Q^\pi \leftarrow$ ***action-value*** function, the expected discounted reward if the agent starts from state s, takes action a, and then follows policy π

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}_\pi\Big[\sum_{t\geq 0}\gamma^t R_t \mid S_0 = s, A_0 = a\Big] \\ &= r(s,a) + \gamma\int_{\mathcal{S}} Q^\pi(s', \pi(s'))\mathcal{P}(s' \mid s,a)\, ds' \end{aligned}$$

Optimal Value

Intuitively, want to solve

$$Q^\pi(s,a) = r(s,a) + \gamma\sum_{s'\in\mathcal{S}}\mathcal{P}(s' \mid s,a)\max_{a'\in\mathcal{A}} Q^\pi(s',a') \quad \forall (s,a)\in\mathcal{S}\times\mathcal{A}$$

Value Iteration

$$Q_{k+1}(s,a) \leftarrow r(s,a) + \gamma\sum_{s'\in\mathcal{S}}\mathcal{P}(s' \mid s,a)\max_{a'\in\mathcal{A}} Q_k(s',a')$$

and obtain the optimal value function when converged.

Challenges:

1) challenging with large state space

2) do not always know $\mathcal{P}$ and $\mathcal{R}$

Batch RL and Approximate Dynamic Programming

Now consider the case with batch data such that S_i, A_i, R_i all follow some distribution.

Then we can define a random variable $t_i = R_i + \gamma \max_{a' \in \mathbb{A}} Q(S_i', a')$

and $\mathbb{E}[t_i \mid S_i, A_i] = (T^*Q)(S_i, A_i)$ so t_i is just the noisy version of $(T^*Q)(S_i, A_i)$. Estimating Q_{k+1} becomes a regression problem.

To calculate Q_{k+1} we minimize the squared error of Q against the regression $R_i + \gamma \max_{a' \in \mathbb{A}} Q_k(S_i', a)$

Remarks: a sequence of regression problems with target changing at each iteration, so the error may accumulate and cause divergence.

Online RL

The agent continually interacts with the environment and update itself with new knowledge of the world and its policy.

Q-Learning

$\rightarrow$ *explore* with probability ϵ : try other actions

$\rightarrow$ *exploit* with probability $1 - \epsilon$: choose the best action from its current, incomplete knowledge of the world

Softmax Action Selection

$$\pi_\beta(S; Q) = A \sim \frac{\exp\{\beta Q(S, a)\}}{\sum_{a \in \mathcal{A}} \exp\{\beta Q(S, a)\}}$$