Recitation 14

Final Review - Questions

Vishakh

CDS

May 4, 2022

Announcement

- HW 7 is due on Friday night
- Finals next Thursday

Agenda

MLE for Conditional Probability Models

- Observe the data $\mathcal{D} = \{x_{1\ldots n}, y_{1\ldots n}\}$

MLE for Conditional Probability Models

- Observe the data $\mathcal{D} = \{x_{1\ldots n}, y_{1\ldots n}\}$
- Compute likelihood of the data as a function of parameter(s) θ

$$L_{\mathcal{D}}(\theta) = \prod_{i=1}^{n} p(y_i | x_i; \theta)$$

- Find that value of $\theta \in \Theta$ which maximizes the likelihood $\rightarrow$ MLE
    - MLE is the ERM of NLL loss

$$\hat{\theta}_{MLE} = \arg\max_{\theta} \prod_{i=1}^{n} p(y_i | x_i; \theta)$$

MLE for Conditional Probability Models

- Observe the data $\mathcal{D} = \{x_{1\dots n}, y_{1\dots n}\}$
- Compute likelihood of the data as a function of parameter(s) θ

$$L_{\mathcal{D}}(\theta) = \prod_{i=1}^{n} p(y_i|x_i;\theta)$$

- Find that value of $\theta \in \Theta$ which maximizes the likelihood $\rightarrow$ MLE
    - MLE is the ERM of NLL loss

$$\hat{\theta}_{MLE} = \arg\max_{\theta} \prod_{i=1}^{n} p(y_i|x_i;\theta)$$

- And we make predictions on new points x' as:

$$\hat{f}(x') = p(y|x';\hat{\theta}_{MLE})$$

Bayesian Methods

- Prior represents belief about θ before observing data $\mathcal{D}$.
- Posterior represents the **rationally "updated" beliefs** after seeing $\mathcal{D}$.

Bayesian Methods

- Prior represents belief about θ before observing data $\mathcal{D}$.
- Posterior represents the **rationally "updated" beliefs** after seeing $\mathcal{D}$.
- All inferences and action-taking are based on the posterior distribution.

Bayesian Methods

- Prior represents belief about θ before observing data $\mathcal{D}$.
- Posterior represents the **rationally "updated" beliefs** after seeing $\mathcal{D}$.
- All inferences and action-taking are based on the posterior distribution.
- In the Bayesian approach,
  - We choose a **family of distributions**, indexed by Θ, and the **prior distribution** on Θ
  - For decision making, need a **loss function**.
  - Everything after that is **computation**.

Bayesian Methods

1. **Define the model**:
    - Choose a parametric family of densities:

$$\{p(\mathcal{D} \mid \theta) \mid \theta \in \Theta\}.$$

    - Choose a distribution $p(\theta)$ on Θ, called the **prior distribution**.

Bayesian Methods

1. **Define the model**:
   - Choose a parametric family of densities:
   $$\{p(\mathcal{D} \mid \theta) \mid \theta \in \Theta\}.$$
   - Choose a distribution $p(\theta)$ on Θ, called the **prior distribution**.
2. After observing $\mathcal{D}$, compute the **posterior distribution** $p(\theta \mid \mathcal{D})$.
$$\begin{aligned} p(\theta \mid \mathcal{D}) &\propto p(\mathcal{D} \mid \theta)p(\theta) \\ &= \underbrace{L_{\mathcal{D}}(\theta)}_{\text{likelihood}} \underbrace{p(\theta)}_{\text{prior}} \end{aligned}$$
3. Choose **action** based on $p(\theta \mid \mathcal{D})$.

Bayesian Methods

Suppose we have a coin with unknown probability of heads $\theta \in (0, 1)$. We flip the coin n times and get a sequence of coin flips with n_h heads and n_t tails.

Recall the following: A Beta (α,β) distribution, for shape parameters $\alpha, \beta > 0$, is a distribution supported on the interval (0, 1) with PDF given by

$$f(x; \alpha, \beta) \propto x^{\alpha-1}(1-x)^{\beta-1}$$

The mean of a Beta (α,β) is $\frac{\alpha}{\alpha+\beta}$. The mode is $\frac{\alpha-1}{\alpha+\beta-2}$ assuming $\alpha, \beta \geq 1$ and $\alpha + \beta > 2$. If $\alpha = \beta = 1$, then every value in (0, 1) is a mode.

Bayesian Methods - Continued

1. Give an expression for the likelihood function $L_D(\theta)$ for this sequence of flips.
2. Suppose we have a Beta (α,β) prior on θ, for some $\alpha, \beta > 0$. Derive the posterior distribution on θ and, if it is a Beta distribution, give its parameters.
3. If your posterior distribution on θ is Beta(3, 6), what is your MAP estimate of θ?

Bayesian Methods - Solution

1. $$L_D(\theta) = \theta^{n_h}(1-\theta)^{n_t}$$

Bayesian Methods - Solution

1. $$L_D(\theta) = \theta^{n_h}(1-\theta)^{n_t}$$

2. $$\begin{aligned} p(\theta \mid \mathcal{D}) &\propto p(\theta)L(\theta) \\ &\propto \theta^{\alpha-1}(1-\theta)^{\beta-1}\theta^{n_h}(1-\theta)^{n_t} \\ &\propto \theta^{n_h+\alpha-1}(1-\theta)^{n_t+\beta-1} \end{aligned}$$

Bayesian Methods - Solution

1. $$L_D(\theta) = \theta^{n_h}(1-\theta)^{n_t}$$

2. $$\begin{aligned} p(\theta \mid \mathcal{D}) &\propto p(\theta)L(\theta) \\ &\propto \theta^{\alpha-1}(1-\theta)^{\beta-1}\theta^{n_h}(1-\theta)^{n_t} \\ &\propto \theta^{n_h+\alpha-1}(1-\theta)^{n_t+\beta-1} \end{aligned}$$

3. Based on information box above, the mode of the beta distribution is $\frac{\alpha-1}{\alpha+\beta-2}$ for $\alpha, \beta > 1$. So the MAP estimate is $\frac{2}{7}$.

Multi-class classification

- Problem: Multiclass classification $\mathcal{Y} = \{1, \ldots, k\}$
- Solution 1: One-vs-All
  - Train k models: $h_1(x), \ldots, h_k(x) : \mathcal{X} \to \mathbb{R}$.
  - Predict with $\arg\max_{y \in \mathcal{Y}} h_y(x)$.
  - Gave simple example where this fails for linear classifiers
- Solution 2: Multiclass loss
  - Train one model: $h(x, y) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$.
    - $h(x, y)$ gives compatibility score between input x and output y
  - Prediction involves solving $\arg\max_{y \in \mathcal{Y}} h(x, y)$.
    - $$\mathcal{F} = \{x \mapsto \underset{y \in \mathcal{Y}}{\arg\max} h(x, y) \mid h \in \mathcal{H}\}$$
    - Final prediction function is a $f \in \mathcal{F}$

Multi-class Classification

We are given the dataset $D = \{(x1, y1), ..., (x_n, y_n)\}$ where $x_i \in \mathbb{R}^2$ and $y_i \in \{1, 2, 3\}$.
Using a one-vs-all methodology, we have fit the score functions $f_i(x) = w_i^T x$ for $i = 1, 2, 3$, where $w_1 = (5, -3)^T$, $w_2 = (-0.2, 0.6)^T$, $w_3 = (-0.6, -0.2)^T$.

Multi-class Classification

We are given the dataset $D = \{(x1, y1), ..., (x_n, y_n)\}$ where $x_i \in \mathbb{R}^2$ and $y_i \in \{1, 2, 3\}$.
Using a one-vs-all methodology, we have fit the score functions $f_i(x) = w_i^T x$ for $i = 1, 2, 3$, where $w_1 = (5, -3)^T$, $w_2 = (-0.2, 0.6)^T$, $w_3 = (-0.6, -0.2)^T$.
To fit each w_i, we used a standard linear SVM with regularization parameter $c = 100$. Suppose we have the following multiclass training data: $\{((-2, -3), 3), ((2, -1), 1), ((1, 2), 2)\}$.
What dataset was given to the SVM to find w_3?

Multi-class Classification

We are given the dataset $D = \{(x1, y1), ..., (x_n, y_n)\}$ where $x_i \in \mathbb{R}^2$ and $y_i \in \{1, 2, 3\}$.
Using a one-vs-all methodology, we have fit the score functions $f_i(x) = w_i^T x$ for $i = 1, 2, 3$, where $w_1 = (5, -3)^T$, $w_2 = (-0.2, 0.6)^T$, $w_3 = (-0.6, -0.2)^T$.
To fit each w_i, we used a standard linear SVM with regularization parameter $c = 100$. Suppose we have the following multiclass training data: $\{((-2, -3), 3), ((2, -1), 1), ((1, 2), 2)\}$.
What dataset was given to the SVM to find w_3?

$\{((-2, -3), 1), ((2, -1), -1), ((1, 2), -1)\}$

Multi-class Classification

We are given the dataset $D = \{(x1, y1), ..., (x_n, y_n)\}$ where $x_i \in \mathbb{R}^2$ and $y_i \in \{1, 2, 3\}$.
Using a one-vs-all methodology, we have fit the score functions $f_i(x) = {w_i}^T x$ for $i = 1, 2, 3$, where $w_1 = (5, -3)^T$, $w_2 = (-0.2, 0.6)^T$, $w_3 = (-0.6, -0.2)^T$.

Multi-class Classification

We are given the dataset $D = \{(x1, y1), ..., (x_n, y_n)\}$ where $x_i \in \mathbb{R}^2$ and $y_i \in \{1, 2, 3\}$.
Using a one-vs-all methodology, we have fit the score functions $f_i(x) = {w_i}^T x$ for $i = 1, 2, 3$, where $w_1 = (5, -3)^T$, $w_2 = (-0.2, 0.6)^T$, $w_3 = (-0.6, -0.2)^T$.
What are the predicted labels for $x_1' = (1, 1)$ and $x_2' = (-2, 0)$

Multi-class Classification

We are given the dataset $D = \{(x1, y1), ..., (x_n, y_n)\}$ where $x_i \in \mathbb{R}^2$ and $y_i \in \{1, 2, 3\}$.
Using a one-vs-all methodology, we have fit the score functions $f_i(x) = {w_i}^T x$ for $i = 1, 2, 3$, where $w_1 = (5, -3)^T$, $w_2 = (-0.2, 0.6)^T$, $w_3 = (-0.6, -0.2)^T$.
What are the predicted labels for $x_1' = (1, 1)$ and $x_2' = (-2, 0)$

$x_1' = (1, 1)$
$f_1(x_1') = 5 \times 1 - 3 \times 1 = 2$
$f_2(x_1') = 0.4$
$f_3(x_1') = -0.8$
$y_1' = \arg\max_{y \in \mathcal{Y}} f_y(x_1') = 1$

Similarly, $y_2' = 3$

Decision Tree

- Decision Trees Setup

  Goal Find a tree that minimize the task loss.

Decision Tree

- Decision Trees Setup

Goal	Find a tree that minimize the task loss.
Problem	Finding the optimal binary tree is computationally intractable.

Decision Tree

- Decision Trees Setup

Goal	Find a tree that minimize the task loss.
Problem	Finding the optimal binary tree is computationally intractable.
Solution	*Greedy* algorithm.

  - Find the best split (according to Gini/Entropy) for a non-terminal node (initially the root)
  - Add two children nodes
  - Repeat until a stopping criterion is reached (eg. max depth)

Decision Tree

- Decision Trees Setup

Goal	Find a tree that minimize the task loss.
Problem	Finding the optimal binary tree is computationally intractable.
Solution	*Greedy* algorithm.

  - Find the best split (according to Gini/Entropy) for a non-terminal node (initially the root)
  - Add two children nodes
  - Repeat until a stopping criterion is reached (eg. max depth)

Decision Tree

- Decision Trees Setup

Goal	Find a tree that minimize the task loss.
Problem	Finding the optimal binary tree is computationally intractable.
Solution	*Greedy* algorithm.

  - Find the best split (according to Gini/Entropy) for a non-terminal node (initially the root)
  - Add two children nodes
  - Repeat until a stopping criterion is reached (eg. max depth)
- Properties of Decision Trees
  - Non-linear classifier that recursively partitions the input space
  - Non-parametric: make no assumption of the data distribution

Ensemble methods

- Combine outputs from multiple models to make better predictions

Ensemble methods

- Combine outputs from multiple models to make better predictions
- Parallel ensemble: models are built independently, eg. bagging
    - Reduce variance of a low bias, high variance estimator by ensembling many estimators trained in parallel.

Ensemble methods

- Combine outputs from multiple models to make better predictions
- Parallel ensemble: models are built independently, eg. bagging
  - Reduce variance of a low bias, high variance estimator by ensembling many estimators trained in parallel.
- Sequential ensemble: models are built sequentially, eg. boosting
  - Reduce the error rate of a high bias estimator by ensembling many estimators trained in sequential.
  - Try to add new learners that do well where previous learners lack

Random Forest

Use bagged decision trees, but modify the tree-growing procedure to reduce the dependence between trees.

- Build a collection of trees independently (in parallel).

Random Forest

Use bagged decision trees, but modify the tree-growing procedure to reduce the dependence between trees.

- Build a collection of trees independently (in parallel).
- When constructing each tree node, restrict choice of splitting variable to a randomly chosen subset of features of size m.
    - Avoid dominance by strong features.

Adaboost Algorithm

- Training set $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$.
- Start with equal weight on all training points $w_1 = \cdots = w_n = 1$.
- Repeat for $m = 1, \ldots, M$:
  - Base learner fits weighted training data and returns $G_m(x)$
  - Increase weight on the points $G_m(x)$ misclassifies
- Final prediction $G(x) = \operatorname{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$. (recall $G_m(x) \in \{-1, 1\}$)

Forward stagewise additive modeling

- **FSAM**: a method used in boosting, greedily fit one function at a time without adjusting previous functions.
- **Learning with FSAM**: Optimizing one basis function each step and add it to the target function.

Gradient Boosting

GBM in computing basis function: for each step

- compute the unconstrained gradient considering all training samples, i.e.

$$g = \nabla_{\boldsymbol{f}} J(\mathrm{f}) = \left(\partial_{\mathrm{f}_1} \ell\left(y_1, \mathrm{f}_1\right), \ldots, \partial_{\mathrm{f}_n} \ell\left(y_n, \mathrm{f}_n\right)\right)$$

- then, compute the basis function parameter within hypothesis space that has smallest Euclidean distance to the gradient, i.e.

$$h = \underset{h \in \mathcal{H}}{\arg\min} \sum_{i=1}^{n} \left(-g_i - h\left(x_i\right)\right)^2$$

- The step size can be predefined or learnt using line search. Finally, we have $f_m \leftarrow f_{m-1} + v_m h_m$

Boostrap

1. What is the probability of not picking one datapoint while creating a bootstrap sample?
2. Suppose the dataset is fairly large. In an expected sense, what fraction of our bootstrap sample will be unique?

Bootstrap

1. $\left(1 - \frac{1}{n}\right)^n$
2. As $n \to \infty$, $\left(1 - \frac{1}{n}\right)^n \to \frac{1}{e}$. So $1 - \frac{1}{e}$ unique samples.

Random Forest and Boosting

Indicate whether each of the statements (about random forests and gradient boosting) is true or false.

- True or False: If your gradient boosting model is overfitting, taking additional steps is likely to help

Random Forest and Boosting

Indicate whether each of the statements (about random forests and gradient boosting) is true or false.

- True or False: If your gradient boosting model is overfitting, taking additional steps is likely to help

  False

Random Forest and Boosting

Indicate whether each of the statements (about random forests and gradient boosting) is true or false.

- True or False: If your gradient boosting model is overfitting, taking additional steps is likely to help

  False

- True or False: In gradient boosting, if you reduce your step size, you should expect to need fewer rounds of boosting (i.e. fewer steps) to achieve the same training set loss.

Random Forest and Boosting

Indicate whether each of the statements (about random forests and gradient boosting) is true or false.

- True or False: If your gradient boosting model is overfitting, taking additional steps is likely to help

  False

- True or False: In gradient boosting, if you reduce your step size, you should expect to need fewer rounds of boosting (i.e. fewer steps) to achieve the same training set loss.

  False

Random Forest and Boosting

Indicate whether each of the statements (about random forests and gradient boosting) is true or false.

- True or False: If your gradient boosting model is overfitting, taking additional steps is likely to help

  False

- True or False: In gradient boosting, if you reduce your step size, you should expect to need fewer rounds of boosting (i.e. fewer steps) to achieve the same training set loss.

  False

- True or False: Fitting a random forest model is extremely easy to parallelize.

Random Forest and Boosting

Indicate whether each of the statements (about random forests and gradient boosting) is true or false.

- True or False: If your gradient boosting model is overfitting, taking additional steps is likely to help

  False

- True or False: In gradient boosting, if you reduce your step size, you should expect to need fewer rounds of boosting (i.e. fewer steps) to achieve the same training set loss.

  False

- True or False: Fitting a random forest model is extremely easy to parallelize.

  True

Random Forest and Boosting

Indicate whether each of the statements (about random forests and gradient boosting) is true or false.

- True or False: Fitting a gradient boosting model is extremely easy to parallelize, for any base regression algorithm.

Random Forest and Boosting

Indicate whether each of the statements (about random forests and gradient boosting) is true or false.

- True or False: Fitting a gradient boosting model is extremely easy to parallelize, for any base regression algorithm.

  False

Random Forest and Boosting

Indicate whether each of the statements (about random forests and gradient boosting) is true or false.

- True or False: Fitting a gradient boosting model is extremely easy to parallelize, for any base regression algorithm.

  False

- True or False: Suppose we apply gradient boosting with absolute loss to a regression problem. If we use linear ridge regression as our base regression algorithm, the final prediction function from gradient boosting always will be an affine function of the input.

Random Forest and Boosting

Indicate whether each of the statements (about random forests and gradient boosting) is true or false.

- True or False: Fitting a gradient boosting model is extremely easy to parallelize, for any base regression algorithm.

  False

- True or False: Suppose we apply gradient boosting with absolute loss to a regression problem. If we use linear ridge regression as our base regression algorithm, the final prediction function from gradient boosting always will be an affine function of the input.

  True

Neural Networks

- **Intuition**: Learning intermediate features via the models.

Neural Networks

- **Intuition**: Learning intermediate features via the models.
- **Optimization**: backpropagation, based on chain rule.

Neural Networks

- **Intuition**: Learning intermediate features via the models.
- **Optimization**: backpropagation, based on chain rule.
- Computing partial derivative of affine transformations etc.
- Activation Functions - sigmoid, ReLU (subgradient), tanh, softmax

Neural Networks

- **True or False**: Consider a hypothesis space $\mathcal{H}$ of prediction functions $f : \mathbb{R}^d \to \mathbb{R}$ given by a multilayer perceptron (MLP) with 3 hidden layers, each consisting of m nodes, for which the activation function is $\sigma(x) = cx$, for some fixed $c \in \mathbb{R}$. Then this hypothesis space is strictly larger than the set of all affine functions mapping $\mathbb{R}^d$ to $\mathbb{R}$.

Neural Networks

- **True or False**: Consider a hypothesis space $\mathcal{H}$ of prediction functions $f : \mathbb{R}^d \to \mathbb{R}$ given by a multilayer perceptron (MLP) with 3 hidden layers, each consisting of m nodes, for which the activation function is $\sigma(x) = cx$, for some fixed $c \in \mathbb{R}$. Then this hypothesis space is strictly larger than the set of all affine functions mapping $\mathbb{R}^d$ to $\mathbb{R}$.

  False, activations should be non-linear for this

Neural Networks

- **True or False**: Let $g : [0,1]^d \to \mathbb{R}$ be any continuous function on the compact set $[0,1]^d$. Then for any $\epsilon > 0$, there exists $m \in \{1, 2, 3, \ldots\}$, $a = (a_1, \ldots, a_m) \in \mathbb{R}^m$,$b = (b_1, \ldots, b_m) \in \mathbb{R}^m$, and $W = \begin{pmatrix} \text{-} & w_1^T & - \\ \vdots & \vdots & \vdots \\ - & w_m^T & - \end{pmatrix} \in \mathbb{R}^{m \times d}$ for which the function $f : [0,1]^d \to \mathbb{R}$ given by

$$f(x) = \sum_{i=1}^{m} a_i \max(0, w_i^T x + b_i)$$

satisfies $|f(x) - g(x)| < \epsilon$ for all $x \in [0,1]^d$.

Neural Networks

- **True or False**: Let $g : [0,1]^d \to \mathbb{R}$ be any continuous function on the compact set $[0,1]^d$. Then for any $\epsilon > 0$, there exists $m \in \{1,2,3,\ldots\}$, $a = (a_1,\ldots,a_m) \in \mathbb{R}^m$,$b = (b_1,\ldots,b_m) \in \mathbb{R}^m$, and $W = \begin{pmatrix} \text{-} & w_1^T & - \\ \vdots & \vdots & \vdots \\ - & w_m^T & - \end{pmatrix} \in \mathbb{R}^{m \times d}$ for which the function $f : [0,1]^d \to \mathbb{R}$ given by

$$f(x) = \sum_{i=1}^{m} a_i \max(0, w_i^T x + b_i)$$

satisfies $|f(x) - g(x)| < \epsilon$ for all $x \in [0,1]^d$.

True, refer Universal Approximation Theorem

K-Means and GMM

- K-means - Initialize cluster centers, compute hard assignments, update centers $\rightarrow$ Iterate to covergence
- GMMs - Similar, but you obtain probabilities of each point belonging to each cluster instead
- **Differences K-Means v.s. GMM**:
  - Hard v.s. soft clustering (utilizes the density in Gaussian).
- **Optimization in GMM**: Expectation Maximization

EM

- **Optimization in GMM**: Expectation Maximization
- **Idea from Latent Variable Model**:
    - We want to compute $p(x)$ parameterized by θ
    - $\mathcal{L}(q, \theta) = -\mathrm{KL}(q(z) \| p(z \mid x; \theta)) + \log p(x; \theta) \leq \log p(x; \theta)$
    - Maximize the ELBO ($\mathcal{L}(q, \theta)$) instead of $p(x; \theta)$

EM

- **Optimization in GMM**: Expectation Maximization
- **Idea from Latent Variable Model**:
  - We want to compute $p(x)$ parameterized by θ
  - $\mathcal{L}(q, \theta) = -\mathrm{KL}(q(z) \| p(z \mid x; \theta)) + \log p(x; \theta) \leq \log p(x; \theta)$
  - Maximize the ELBO $(\mathcal{L}(q, \theta))$ instead of $p(x; \theta)$
- **Expectation Maximization**:
  - E-step: we update $q(z)$ (GMM: the γ, the weights associated with each point given the cluster centroids)
  - M-step: we update parameters $p(x|z)$ of, i.e. θ. (GMM: μ, Σ updating the centroids)

Mixture Models

Suppose we have a latent variable $z \in \{1, 2, 3\}$ and an observed variable $x \in (0, \infty)$ generated as follows:

$$z \sim \text{Categorical}(\pi_1, \pi_2, \pi_3)$$

$$x \mid z \sim \text{Gamma}(2, \beta_z),$$

where $(\beta_1, \beta_2, \beta_3) \in (0, \infty)^3$, and Gamma$(2, \beta)$ is supported on $(0, \infty)$ and has density $p(x) = \beta^2 x e^{-\beta x}$. Suppose we know that $\beta_1 = 1, \beta_2 = 2, \beta_3 = 4$. Give an explicit expression for $p(z = 1 | x = 1)$ in terms of the unknown parameters π_1, π_2, π_3.

Mixture Model

$$p(z=1|x=1) \propto p(x=1|z=1)p(z=1) = \pi_1 e^{-1}$$
$$p(z=2|x=1) \propto p(x=1|z=2)p(z=2) = \pi_2 4e^{-2}$$
$$p(z=3|x=1) \propto p(x=1|x=3)p(z=3) = \pi_3 16e^{-4}$$

$$p(z=1|x=1) = \frac{\pi_1 e^{-1}}{\pi_1 e^{-1} + \pi_2 4e^{-2} + \pi_3 16e^{-4}}$$