Lecture 4 MLE & Opt

Logistics:

- Participation 1 due today
- Participation 2 posted later today
- HW1 posted next Tuesday
- Out of town Feb 28 & Mar 3:
  guest lecturers or recorded lectures

Agenda

- MLE for the Categorical Model
- Practical ML procedure
- Regularization
- Risk decomposition: how well do we need to optimize?

MLE for Categorical Model

$$y|x \sim \text{Categorical}\left(p_1(x), \cdots, p_K(x)\right)$$

where $p_i(x) = \dfrac{e^{\theta_i^T x}}{Z_\theta(x)}$ where $Z_\theta(x) = \sum_{j=1}^{K} e^{\theta_j^T x}$

recall $p_\theta(x) = \begin{bmatrix} p_1(x) \\ \vdots \\ p_K(x) \end{bmatrix} = \dfrac{e^{\theta x}}{1^T e^{\theta x}} = \text{softmax}(\theta x)$

where $\theta = \begin{bmatrix} \theta_1^T \\ \vdots \\ \theta_K^T \end{bmatrix}$

Given $\{(y_i, x_i)\}_{i=1}^n$, where $y_i \in \mathbb{R}^k$ one-hot encodes the class of the ith example (e.g. class 1 encodes as $\begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix}$ and class k as $\begin{bmatrix}0\\0\\\vdots\\1\end{bmatrix}$), our goal is to learn the model parameters $\theta \in \mathbb{R}^{k \times d}$ using MLE

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^{n} -\log p_\theta(y_i | x_i)$$

Recall

$$p_\theta(y_i = e_j | x_i) = e_j^T p_\theta(x_i) = y_i^T p_\theta(x_i)$$

$$= y_i^T \text{softmax}(\theta x_i)$$

So

$$\mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n} -\log p_\theta(y_i|x_i)$$

$$= \frac{1}{n}\sum_{i=1}^{n} -\log\left[y_i^T \operatorname{softmax}(\theta x_i)\right]$$

$$= \frac{1}{n}\sum_{i=1}^{n} -y_i^T \log\left[\operatorname{softmax}(\theta x_i)\right]$$

apply log entrywise to its vector argument

$$\overset{?}{=} \frac{1}{n}\sum_{i=1}^{n} \ell\left(\operatorname{softmax}(\theta x_i), y_i\right)$$

what is the correct choice of ℓ to make the NLL have the form of an empirical risk

Guess: ℓ is the cross-entropy loss

What is cross-entropy? Easiest to first define KL-divergence

$$D_{KL}(p \| q) = \sum p_i \ln \frac{p_i}{q_i}$$ if p and q are two probability vectors

$$= \underbrace{\sum p_i \ln p_i}_{\text{Entropy of } p \text{, } H(p)} \underbrace{- \sum p_i \ln q_i}_{\text{cross-entropy of } p \text{ and } q \text{, } H(p,q)}$$

So minimizing $H(p,q)$ w.r.t. q minimizes $D_{KL}(p \| q)$ w.r.t. q

Recall the negative log-likelihood w.r.t. the ith sample

$$-y_i^T \log(\text{softmax}(\theta x_i))$$

$$= \sum_{j=1}^{k} (y_i)_j \ln(p_\theta(x_i)_j)$$

$$= H(y_i, p_\theta(x_i))$$

Note that if the ith example is from class j, so $y_i = e_j$, then

$$-y_i^T \log(\text{softmax}(\theta x_i)) = -e_j^T \ln(p_\theta(x_i))$$

$$= -\ln(p_\theta(x_i)_j)$$

is the log of the probability assigned to the true class

Upshot is that

$$\mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n} H(y_i, \text{softmax}(\theta x_i))$$

$$= \frac{1}{n}\sum_{i=1}^{n} -\ln\left[p_\theta(x_i)_{y_i}\right]$$

So minimizing the NLL learns model parameters $\hat{\theta}$ that maximize the probabilities assigned by the model to the true classes of the observations, and the loss function for this ERM is

$$\ell(u, v) = H(u, v)$$

The optimization problem is

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\ \mathcal{L}(\theta)$$

$$= \underset{\theta}{\operatorname{argmin}}\ \frac{1}{n}\sum_{i=1}^{n} -y_i^T \ln\left(\operatorname{softmax}(\theta x_i)\right)$$

$$= \underset{\theta}{\operatorname{argmin}}\ \frac{1}{n}\sum_{i=1}^{n} -y_i^T \ln\left(\frac{e^{\theta x_i}}{Z_\theta(x_i)}\right)$$

$$= \underset{\theta}{\operatorname{argmin}}\ \frac{1}{n}\sum_{i=1}^{n} -y_i^T\left[\ln\left(e^{\theta x_i}\right) - \left(\ln Z_\theta(x_i)\right)1\right]$$

$$= \underset{\theta}{\operatorname{argmin}}\ \frac{1}{n}\sum_{i=1}^{n}\left[-y_i^T\theta x_i + y_i^T 1 \ln Z_\theta(x_i)\right]$$

$$= \underset{\theta}{\operatorname{argmin}} \; \frac{1}{n} \sum_{i=1}^{n} \left[-y_i^T \theta x_i + \ln Z_\theta(x_i) \right]$$

$$= \underset{\theta}{\operatorname{argmin}} \; \frac{1}{n} \sum_{i=1}^{n} -y_i^T \theta x_i + \frac{1}{n} \sum_{i=1}^{n} \ln Z_\theta(x_i)$$

$$= \underset{\theta}{\operatorname{argmin}} \; \frac{1}{n} \sum_{i=1}^{n} -y_i^T \theta x_i + \frac{1}{n} \sum_{i=1}^{n} \ln \left(\sum_{j=1}^{k} e^{\theta_j^T x_i} \right)$$

This is the optimization problem we solve in practice to learn θ using MLE for multiclass classification.

Practical Considerations

train vs test vs validation splits

use training data to solve our ERM, that is

$$\hat{f} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n_{train}} \sum_{i=1}^{n_{train}} \ell(f(x_i), y_i)$$

$$= R_{n_{train}}(f)$$

use fresh samples to estimate the risk

$$R(\hat{f}) = \mathbb{E}_{\mathcal{D}}\, \ell(\hat{f}(x), y)$$

$$\approx \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} \ell(f(x_i), y_i)$$

We use the test set to estimate the "generalization error"

$$R(\hat{f}) - R_{n_{train}}(\hat{f}) \approx R_{n_{test}}(\hat{f}) - R_{n_{train}}(\hat{f})$$

What if we want to choose between different model classes, e.g. $\mathcal{F}_1$ and $\mathcal{F}_2$?

- use training set to find best (empirically) models $\hat{f}_1 \in \mathcal{F}_1$ and $\hat{f}_2 \in \mathcal{F}_2$
- use validation dataset to select best of $\hat{f}_1$ and $\hat{f}_2$ on a fresh independent set of samples
- estimate the generalization gap of this selected model on a fresh independent test set

General flow for supervised ML

- Fix our problem domain, risk (loss) measure, collect appropriate data $\{(x_i, y_i)\}_{i=1}^{n}$
- Split our data into train/validation/test (60/20/20)
- Determine ahead of time a set models and hyperparameters to try
- Fit each of these using the same training data
- Measure their performance using the validation data and select the best, f_{opt}
- Report an estimate of the gen gap / pop risk of f_{opt} using the test data