Assorted Topics (1)

CS772A: Probabilistic Machine Learning

Piyush Rai

Plan today

- Calibration
- Frequentist approach for estimating uncertainty
- Some classical probabilistic models for sequential data
  - HMM and State-Space Models (SSM)

Calibration

Calibration

- Assume a classifier that outputs probabilities $f(x_n) = [a_{n1}, a_{n2}, \ldots, a_{nC}]$ such that

Predicted label

$$\hat{y}_n = \text{argmax}_{c=\{1,2,\ldots,C\}}\, a_{nc}$$

Probability of the predicted label (confidence of f for this prediction)

$$\hat{a}_n = \max_{c=\{1,2,\ldots,C\}} a_{nc}$$

- Notion of calibration: Predictions should not neither be over-confident, nor under-confident
- Desirable: Predictions with confidence $\mu \in (0,1)$ are correct $(100 \times \mu)\%$ of the time
- Assume $\mathcal{B}_b$ as set of samples for which $\hat{a}_n$ falls in bin $I_b = (\frac{b-1}{B}, \frac{b}{B}]$

Average accuracy of bin b

$$\text{acc}(B_b) = \frac{1}{|B_b|}\sum_{n \in B_b} \mathbb{I}(\hat{y}_n = y_n)$$

Average confidence of bin b

$$\text{conf}(B_b) = \frac{1}{|B_b|}\sum_{n \in B_b} \hat{a}_n$$

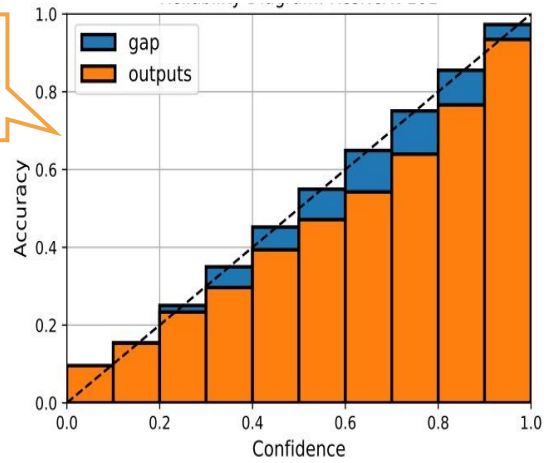
- We want bins' average accuracies to match bins' average confidence

Reliability Diagrams and A Calibration Metric

- Reliability diagrams are plots of accuracy vs confidence

Using $B = 10$ equal-width bins

It's just one simple way; other ways also possible to construct the bins



- Several metrics exist to measure how well-calibrated the model's predictions are
- Expected Calibration Error (ECE) is one such popular metric

Should be small for a well-calibrated model

$$\mathrm{ECE}(f) = \sum_{b=1}^{B} \frac{|\mathcal{B}_b|}{B} \left|\mathrm{acc}(\mathcal{B}_b) - \mathrm{conf}(\mathcal{B}_b)\right|$$

ECE is the average "gap" area in the reliability diagram

Calibration Methods (contd)

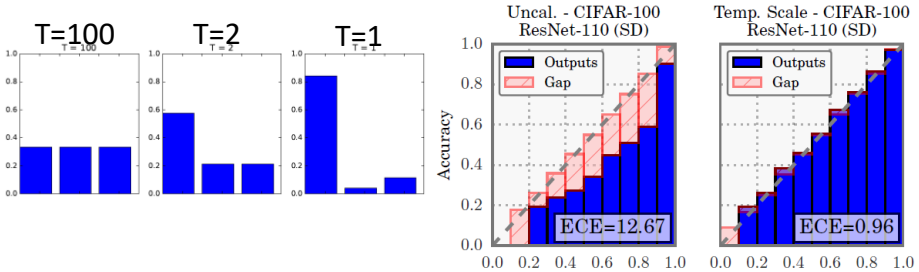
Parameters of the trained model are kept frozen in this process

The scaling parameters (w or T) are learned by minimizing the loss on some validation set.

- Method 1: Calibrate an already trained model in a post-hoc manner, e.g.,
  - Requires learning to scale the logits produced by the model, e.g.,

$$\text{softmax}(z_1, z_2, \ldots, z_C) \Rightarrow \text{softmax}\,(w_1 z_1 + b_1, w_2 z_2 + b_2, \ldots, w_C z_C + b_c)$$

$$\text{softmax}(z_1, z_2, \ldots, z_C) \Rightarrow \text{softmax}\left(\frac{z_1}{T}, \frac{z_2}{T}, \ldots, \frac{z_C}{T}\right)$$



- Method 2: Change the training procedure, e.g.,
  - Add a regularizer which avoids overconfident predictions

Maximize the likelihood

Maximize the entropy of the predictive distribution to reduce overconfidence

$$\mathcal{L} = \sum_{i=1}^{N} \log p(y_i|x_i, w) + \mathbb{H}[\log p(y_i|x_i, w)]$$

  - Trained with smoothed labels instead of one-hot labels

$$[0, 0, 1, 0] \Rightarrow [0.05, 0.05, 0.85, 0.05]$$

Frequentist Statistics
(vs Bayesian Statistics)

Frequentist Statistics

- The Bayesian approach treats parameters/model unknowns as random variables
- In the Bayesian approach, the posterior over these r.v.'s help capture the uncertainty
- The Frequentist approach is a different way to capture uncertainty
  - Don't treat parameters as r.v. but as fixed unknowns
  - Treat parameters as a function of the dataset, e.g., $\hat{\theta}(\mathcal{D}) = \pi(\mathcal{D})$

This can be some point estimate, e.g., MLE, MAP, method of moments, etc.

  - Variations in param estimates over different datasets represents their uncertainty

A random dataset drawn from the true data distribution

True unknown value of the parameter

$$\tilde{\mathcal{D}}^{(s)} = \{\boldsymbol{x}_n \sim p(\boldsymbol{x}_n|\boldsymbol{\theta}^*) : n = 1 : N\} \qquad (s = 1, 2, \ldots, S)$$

The estimated distribution of the parameters given any randomly drawn dataset from the true data distribution

$$p(\pi(\tilde{\mathcal{D}}) = \boldsymbol{\theta}|\tilde{\mathcal{D}} \sim \boldsymbol{\theta}^*) \approx \frac{1}{S}\sum_{s=1}^{S} \delta(\boldsymbol{\theta} = \pi(\tilde{\mathcal{D}}^{(s)}))$$

Param estimate using the s-th sampled dataset

As $S \to \infty$, this is known as the "sampling distribution" of the estimator

Note that sampling distribution is different from a posterior distribution we infer in Bayesian learning (there, we condition on a fixed training set)

But if the estimator is MLE and Bayesian method's prior is uniform, then both distributions are very similar (sampling distribution is often called "poor man's posterior"

Approximating the sampling distribution

- Since the true $\boldsymbol{\theta}^*$ is not known, we can't compute the sampling distribution exactly

$$\tilde{\mathcal{D}}^{(s)} = \{\boldsymbol{x}_n \sim p(\boldsymbol{x}_n|\boldsymbol{\theta}^*) : n = 1 : N\} \qquad (s = 1, 2, \ldots, S)$$

$$p(\pi(\tilde{\mathcal{D}}) = \boldsymbol{\theta}|\tilde{\mathcal{D}} \sim \boldsymbol{\theta}^*) \approx \frac{1}{S}\sum_{s=1}^{S} \delta(\boldsymbol{\theta} = \pi(\widetilde{\mathcal{D}^{(s)}}))$$

- Bootstrap is a popular method to approximate the sampling distribution
- Two types of bootstrap methods: parametric and nonparametric bootstrap

Parametric Bootstrap

- Get a point est. of $\boldsymbol{\theta}$ using training data

$$\hat{\boldsymbol{\theta}} = \pi(\mathcal{D})$$

- Generate multiple datasets using $\hat{\boldsymbol{\theta}}$ as

$$\tilde{\mathcal{D}}^{(s)} = \{\boldsymbol{x}_n \sim p(\boldsymbol{x}_n|\hat{\boldsymbol{\theta}}) : n = 1 : N\} \quad (s = 1, 2, \ldots, S)$$

- Now compute the approximation as

$$p(\pi(\tilde{\mathcal{D}}) = \boldsymbol{\theta}|\tilde{\mathcal{D}} \sim \boldsymbol{\theta}^*) \approx \frac{1}{S}\sum_{s=1}^{S} \delta(\boldsymbol{\theta} = \pi(\widetilde{\mathcal{D}^{(s)}}))$$

Nonparametric Bootstrap

- Use sampling with replacement on original training set to generate S datasets with N datapoints in each

Each dataset will contain roughly 63% unique datapoints from original training set

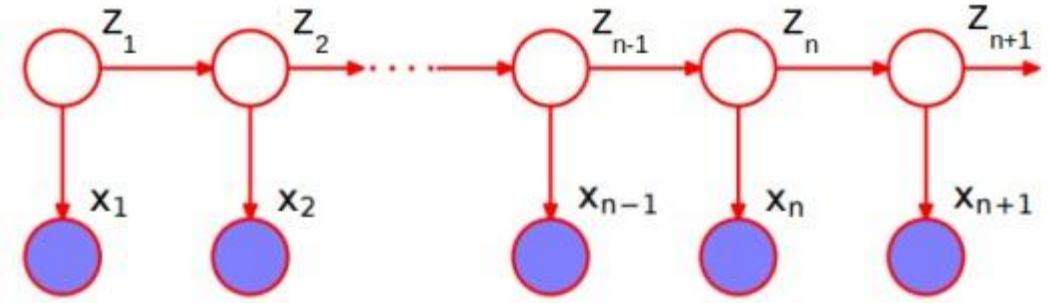
- Now compute the approximation as

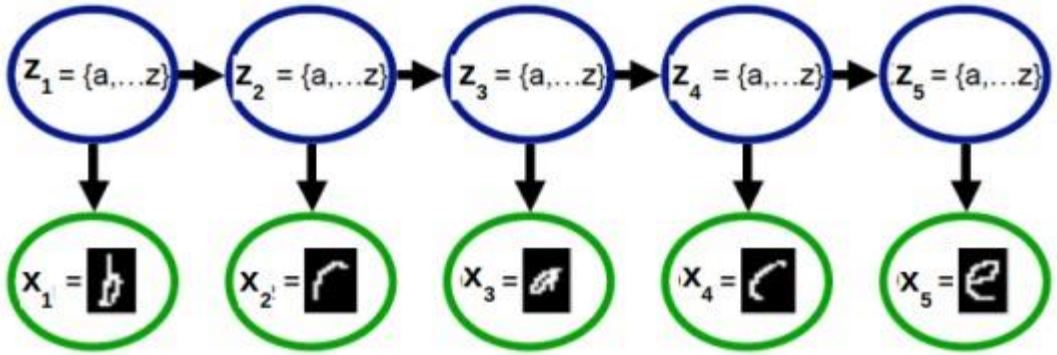
$$p(\pi(\tilde{\mathcal{D}}) = \boldsymbol{\theta}|\tilde{\mathcal{D}} \sim \boldsymbol{\theta}^*) \approx \frac{1}{S}\sum_{s=1}^{S} \delta(\boldsymbol{\theta} = \pi(\widetilde{\mathcal{D}^{(s)}}))$$

Probabilistic Models for Sequential Data

Latent Variable Models for Sequential Data

- Task: Given a sequence of observations, infer the latent state of each observation





Observation model

$$\boldsymbol{x}_n|\boldsymbol{z}_n \sim p(\boldsymbol{x}_n|\boldsymbol{z}_n) \quad \text{(i.i.d. draws of } \boldsymbol{x}_n \text{ given } \boldsymbol{z}_n\text{)}$$

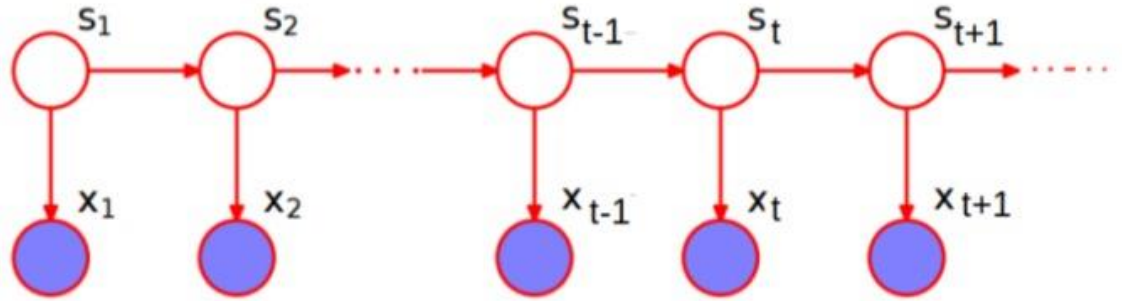
State-transition model

$$\boldsymbol{z}_n|\boldsymbol{z}_{n-1} \sim p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) \quad \text{(first-order dependence b/w } \boldsymbol{z}_n\text{'s)}$$

- If $\boldsymbol{z_n}$'s are discrete, we have a hidden Markov model (HMM) $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1} = \ell) = \text{multinoulli}(\boldsymbol{\pi}_\ell)$
- If $\boldsymbol{z_n}$'s are real-valued, we have a state-space model (SSM) $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) = \mathcal{N}(\mathbf{A}\boldsymbol{z}_{n-1}, \mathbf{I}_K)$

State-Space Models

- In the most general form, the state-transition and observation models of an SSM



Using '**s**' instead of '**z**' to refer to states

Using 't' to denote the 'time-step'

HMM is similar to SSM except the state-transition model is a discrete distribution

g_t, h_t can be linear or nonlinear functions

$$\boldsymbol{s}_t|\boldsymbol{s}_{t-1} = g_t(\boldsymbol{s}_{t-1}) + \epsilon_t \quad \text{(must be a cont. dist. over } \boldsymbol{s}_t\text{)}$$

$$\boldsymbol{x}_t|\boldsymbol{s}_t = h_t(\boldsymbol{s}_t) + \delta_t \quad \text{(can be any dist. over } \boldsymbol{x}_t\text{)}$$

- Assuming Gaussian noise in the state-transition and observation models

This is a Gaussian SSM

$$\boldsymbol{s}_t|\boldsymbol{s}_{t-1} \sim \mathcal{N}(\boldsymbol{s}_t|g_t(\boldsymbol{s}_{t-1}), \mathbf{Q}_t)$$

$$\boldsymbol{x}_t|\boldsymbol{s}_t \sim \mathcal{N}(\boldsymbol{x}_t|h_t(\boldsymbol{s}_t), \mathbf{R}_t)$$

If g_t, h_t, Q_t, R_t are independent of t then it is called a stationary model

g_t, h_t, Q_t, R_t may be known or can be learned

State-Space Models: A Simple Example

- Consider the linear Gaussian SSM

$$\boldsymbol{s}_t|\boldsymbol{s}_{t-1} = \mathbf{A}_t\boldsymbol{s}_{t-1} + \epsilon_t$$

$$\boldsymbol{x}_t|\boldsymbol{s}_t = \mathbf{B}_t\boldsymbol{s}_t + \delta_t$$

- Suppose $\boldsymbol{x}_t \in \mathbb{R}^2$ denotes the (noisy) observed 2D location of an object
- Suppose $\boldsymbol{s}_t \in \mathbb{R}^6$ denotes the "state" vector

$$\boldsymbol{s}_t = [\text{pos1}, \text{vel1}, \text{accel1}, \text{pos2}, \text{vel2}, \text{accel2}]$$

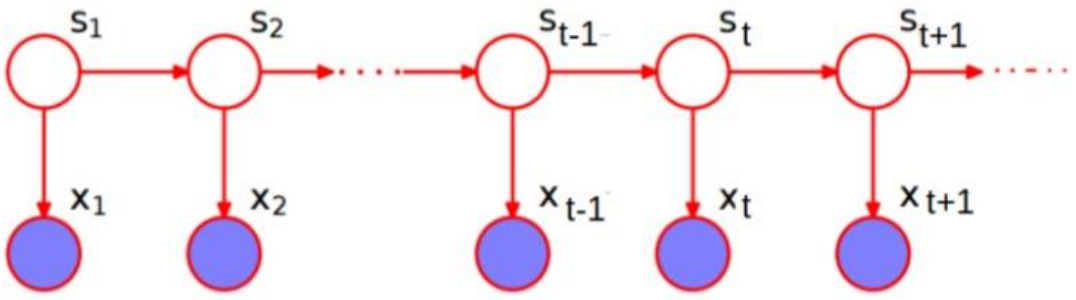
- Here is an example SSM for this problem with pre-defined $\mathbf{A_t}$ and $\mathbf{B_t}$ matrices

$$\mathbf{s}_t = \underbrace{\begin{bmatrix} 1 & \Delta t & \frac{1}{2}(\Delta t)^2 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & e^{-\alpha\Delta t} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & \frac{1}{2}(\Delta t)^2 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & e^{-\alpha\Delta t} \end{bmatrix}}_{\mathbf{A}_t} \mathbf{s}_{t-1} + \epsilon_t$$

$$\mathbf{x}_t = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}}_{\mathbf{B}_t} \mathbf{s}_t + \delta_t$$

Typical Inference Task for Gaussian SSM

- One of the key tasks: Given sequence $x_1, x_2, \ldots, x_T$, infer latent $s_1, s_2, \ldots, s_T$



- Usually two ways of inferring the latent states
  - Infer $p(s_t|x_1, x_2, \ldots, x_t)$: Called the "filtering" problem

Turns out to be another Gaussian

$$p(s_t|x_1, x_2, \ldots, x_t) \propto \underbrace{p(x_t|s_t)}_{\mathcal{N}(x_t|Bs_t,R)} \int \underbrace{p(s_t|s_{t-1})}_{\mathcal{N}(s_t|As_{t-1},Q)} p(s_{t-1}|x_1, x_2, \ldots, x_{t-1}) ds_{t-1}$$

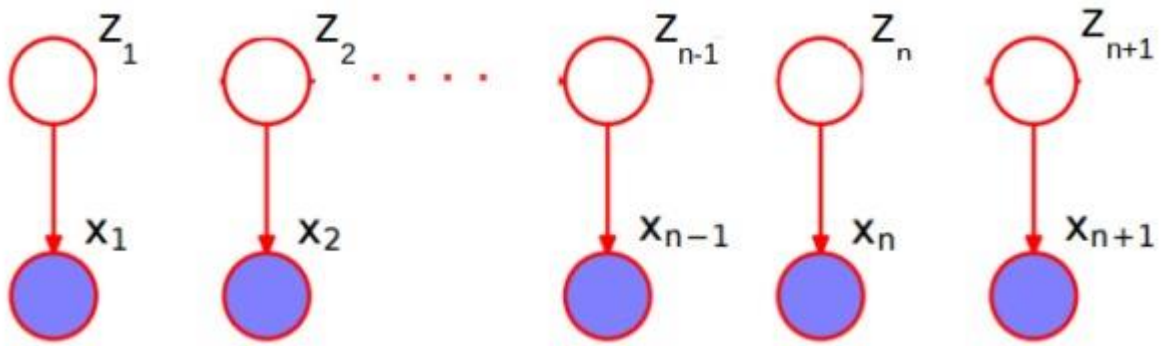
A Gaussian

Kalman Filtering is a popular algorithm for a linear Gaussian SSM

  - Infer $p(s_t|x_1, x_2, \ldots, x_t, \ldots, x_T)$: Called the "smoothing" problem
- Some other tasks one can solve for using an SSM
  - Predicting future states $p(s_{t+h}|x_1, x_2, \ldots, x_t)$ for $h \geq 1$, given observations thus far
  - Predicting future observations $p(x_{t+h}|x_1, x_2, \ldots, x_t)$ for $h \geq 1$, given observations thus far

A Special Case

- What if we have i.i.d. latent states, i.e.,. $p(z_n|z_{n-1}) = p(z_n)$?



- Discrete case (HMM) becomes a simple mixture model $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1} = \ell) = p(\boldsymbol{z}_n) = \text{multinoulli}(\boldsymbol{\pi})$
- Real-valued case (SSM) becomes a PPCA model $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) = p(\boldsymbol{z}_n) = \mathcal{N}(\mathbf{0}, \mathbf{I}_K)$ or $\mathcal{N}(\boldsymbol{\mu}, \Psi)$
- Inference algos for HMM/SSM are thus very similar to that of mixture models/PPCA
  - Only main difference is how the latent variables $\boldsymbol{z}_n$'s are inferred since they aren't i.i.d.
  - E.g., if using EM, only E step needs to change (Bishop Chap 13 has EM for HMM and SSM)