Advanced probabilistic methods

Lecture 7: Model selection, Edward introduction

Pekka Marttinen

Aalto University

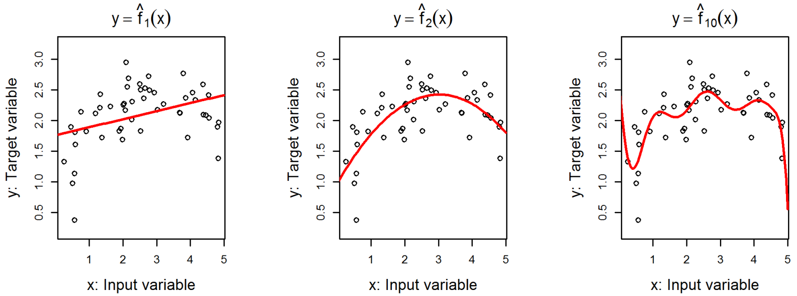
March, 2019

Lecture 7 overview

- Bayesian model selection
  - marginal likelihood
  - BIC, Laplace approximation, VB lower bound
- Predictive model selection
  - AIC, (DIC, WAIC, etc.)
  - Cross-validation
- Introduction to probabilistic programming using Edward
- Lecture based on (suggested reading):
  - Barber: Ch. 12 (Bayesian model selection)
  - Hastie et al. *The Elements of Statistical Learning*, (available at *http://statweb.stanford.edu/~tibs/ElemStatLearn/)*: Ch. 7.1, 7.2, 7.4, 7.5, 7.10 (for AIC and CV)

Model selection

- Possible goal may be to learn
  - **the most useful model**, for example the one that best predicts future observations
  - **the most probable model**, for example when comparing between scientific hypotheses and different hypotheses correspond to different models



Bayesian model selection

- Consider m models M_i with parameters θ_i and associated priors,

$$p(x, \theta_i | M_i) = p(x|\theta_i, M_i) p(\theta_i | M_i), \quad i \in 1, \dots, m,$$

- We can compute the **model posterior probabilities**

$$p(M_i|x) = \frac{p(x|M_i)p(M_i)}{p(x)},$$

where

$$p(x|M_i) = \int p(x|\theta_i, M_i) p(\theta_i|M_i) d\theta_i \quad \text{and}$$
$$p(x) = \sum_{i=1}^{m} p(x|M_i)p(M_i)$$

Bayes factors

- For comparing two models, we compute the **Bayes' factor**

$$\underbrace{\frac{p(M_i|x)}{p(M_j|x)}}_{\text{Posterior odds}} = \underbrace{\frac{p(x|M_i)}{p(x|M_j)}}_{\text{Bayes' factor}} \times \underbrace{\frac{p(M_i)}{p(M_j)}}_{\text{Prior odds}},$$

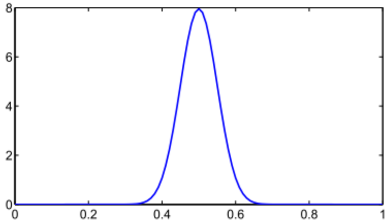
- Bayes factor is the ratio of **marginal likelihoods** $p(D|M_i)$ and it tells how much more seeing the data D has increased the probability of model M_i as opposed to model M_j.

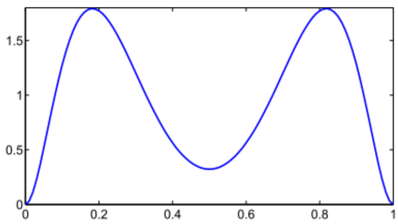
Bayes factor example (1/3)

- **Problem:** given N throws of a coin, determine whether a coin is biased or unbiased.
- Let θ denote the probability of heads. The probability of observing h heads and $N-h$ tails in a sequence of N throws is

$$p(H=h) = \binom{N}{h}\theta^h(1-\theta)^{N-h}$$

- The difference between models is encoded in the prior distribution of θ (**Left**: fair coin, **Right**: biased coin)





Bayes factor example (2/3)

- M_{fair} ('Coin is fair') corresponds to prior

$$\begin{aligned} p(\theta|M_{fair}) &= Beta(\theta|a,b) \\ &= B(a,b)^{-1}\theta^{a-1}(1-\theta)^{b-1} \end{aligned}$$

  where $a = b = 50$.

- Probability of h heads in N throws of the coin is given by

$$\begin{aligned} p(x|M_{fair}) &= \int p(x|\theta, M_{fair})p(\theta|M_{fair})d\theta \\ &= \binom{N}{h} B(a,b)^{-1} \int \theta^h (1-\theta)^{N-h}\theta^{a-1}(1-\theta)^{b-1}d\theta \\ &= \binom{N}{h} B(a,b)^{-1} \int \theta^{h+a-1}(1-\theta)^{N-h+b-1}d\theta \\ &= \binom{N}{h} B(a,b)^{-1} B(h+a, N-h+b) \end{aligned}$$

Bayes factor example (3/3)

- M_{biased} ('Coin is biased') corresponds to assuming

$$p(\theta|M_2) = 0.5 \times Beta(\theta|3,10) + 0.5 \times Beta(\theta|10,3)$$

- We get

$$p(x|M_2) = \frac{1}{2}\binom{N}{h}\left\{\frac{B(h+3, N-h+10)}{B(3,10)} + \frac{B(h+10, N-h+3)}{B(10,3)}\right\}$$

- For example with $h = 50$ and $N = 70$, we get

$$BF_{fair,biased} = \frac{p(x|M_{fair})}{p(x|M_{biased})} = 0.087.$$

This indicates that if the models are *a priori* equally likely, after seeing the data, M_{biased} is about 11 times more probable than M_{fair}.

Laplace approximation for marginal likelihood

- Laplace approximation for $p(x|M)$

$$\log p(x|M) \approx \log p(x|\widehat{\theta}, M) + \log p(\widehat{\theta}|M) + \frac{D}{2}\log(2\pi) - \frac{1}{2}\log|H_{\widehat{\theta}}|,$$

where

$$\widehat{\theta} = \arg\max_{\theta} p(x|\theta, M)p(\theta|M)$$

is the MAP estimate and $H_{\widehat{\theta}}$ is the Hessian (second derivative for univariate θ) of

$$f(\theta) = -\log\left[p(x|\theta, M)p(\theta|M)\right]$$

at $\widehat{\theta}$.

BIC approximation for marginal likelihood*

- BIC approximation[1]

$$\mathrm{BIC}(M) = \log p(x|\widehat{\theta}, M) - \frac{D}{2}\log N$$

  is obtained from the Laplace approximation by assuming $p(\theta) = const$, $H \approx NI_D$, and $N \to \infty$.

- Note that we can compute the approximate Bayes factor using

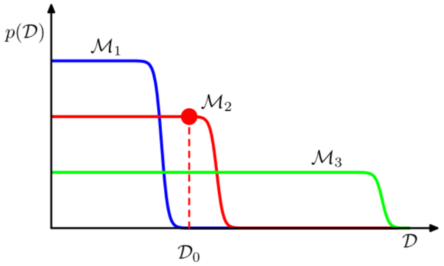
$$\mathrm{BF}_{12} = \frac{\exp(\mathrm{BIC}(M_1))}{\exp(\mathrm{BIC}(M_2))},$$

  or similarly by plugging in exponentiated Laplace approximation (Laplace is better, both to be used with caution, especially with small N).

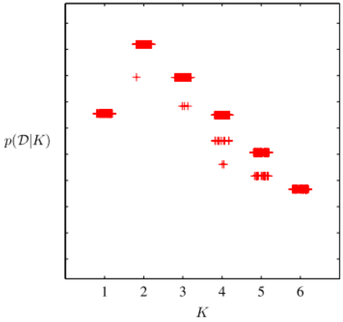
[1] Sometimes there is -2 in the front.

Bayesian model selection and Occam's razor

- When complexity of M increases, $p(x|\widehat{\theta}, M)$ always increases
- On the other hand, $p(x|M)$ **is the highest for the simplest model that can explain the data** (=Occam's razor principle)
- **Left:** illustration with model complexity increasing from M_1 to M_3
- **Right:** $p(x|K)$ for the number K of GMM components for the 'Old Faithful' data (approximated using VB lower bound)



Bishop, Fig. 3.13



Bishop, Fig. 10.7

Selecting models for prediction, concepts (1/2)

- X: input variables, Y: target variable, $\widehat{f}(X)$: prediction model estimated from a training data $\mathcal{T}$.
- Loss function measures the (lack of) accuracy of prediction
- Squared loss

$$L(Y, \widehat{f}(X)) = (Y - \widehat{f}(X))^2$$

- Loss based on log-likelihood

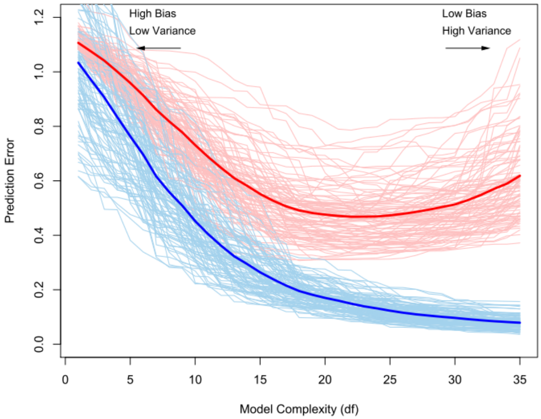
$$L(Y, \theta(X)) = -2 \log p(Y | \theta(X)),$$

where $\theta(X)$ is a parameter of the prediction model.

Selecting models for prediction, concepts (2/2)

$$\mathrm{Err}_{\mathcal{T}} = E\left[L(Y, \widehat{f}(X))|\mathcal{T}\right] \qquad \text{(test/prediction/generalization error)}$$

$$\overline{\mathrm{err}} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \widehat{f}(x_i)) \qquad \text{(training error)}$$

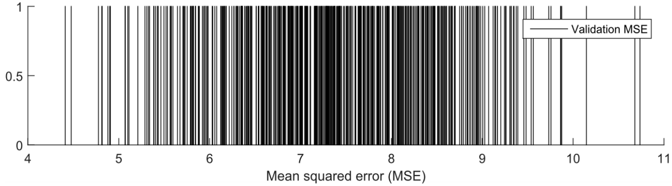


Predictive model selection criteria

- Predictive model selection criteria aim to approximate **expected prediction accuracy** in a new data set, either
  - analytically (e.g. AIC, DIC, WAIC), or
  - by efficient sample re-use (e.g. cross-validation)
- Hence, they aim to find a model that is **suitable for prediction**.
- Asymptotically, AIC and leave-one-out cross validation are equivalent.

Example (validation vs. test error)*

- Data $(\mathbf{x}_i, y_i)$ is simulated using $y_i = \sum_{i=1}^{30} w_i x_i + \epsilon_i$, where $w_i \sim U(-1, 1)$, and $\epsilon_i \sim N(0, 0.1^2)$.
- 500 candidate models created by randomly selecting 10 covariates out of 30, and fitting a linear model with the selected covariates.
- $n_{Train} = 300$ and $n_{Valid} = 60$. Validation MSEs for different models:



- **Question:** What is your best guess for the test set MSE for the best model?

AIC, basic idea*

- It can be shown that for large N

$$-2 \cdot E\left[\log p(\widetilde{y}|\widehat{\theta})\right] \approx -\frac{2}{N}\log p(y|\widehat{\theta}) + 2 \cdot \frac{d}{N},$$

  where $\widetilde{y}$ is an unobserved future observation and

$$\log p(y|\widehat{\theta}) = \sum_{i=1}^{N} \log p(y_i|\widehat{\theta})$$

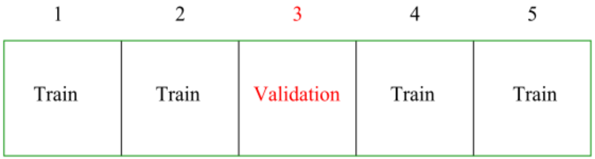
  is the log-likelihood.
- This gives rise to:

$$\text{AIC} = -\frac{2}{N}\log p(y|\widehat{\theta}) + 2 \cdot \frac{d}{N}$$

  (the smallest AIC is the best)
- **Main point**: AIC is one possible analytical approximation for the expected prediction accuracy measured using log probability of future data[2].

[2] For more Bayesian variants, see, e.g., Gelman *et al.* Stat. Comput. (2014)

Cross-Validation (CV), basic idea*



- Let $\kappa : \{1, \dots, N\} \longmapsto \{1, \dots, K\}$ denotes the fold to which observation i belongs. Then

$$CV(\widehat{f}) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \widehat{f}^{-\kappa(i)}(x_i)),$$

  where $\widehat{f}^{-\kappa(i)}$ is the predictor model trained without fold $\kappa(i)$.
- CV yields an estimate of the expected prediction error $E\left[L(Y, \widehat{f}(X))\right]$.

A wrong way to do cross-validation*

- A (wrong!) strategy for building a classifier with a large number of predictors
  1. Pre-screening of the predictors: find a subset of predictors with strong univariate correlation with the class label
  2. Using the set of predictors from pre-screening, build a multivariate classifier
  3. Use cross-validation to estimate the unknown tuning parameter and to estimate the prediction error of the final model
- **Question:** what's the problem?

The correct way*

- The correct way for building a classifier with a large number of predictors
  1. Divide the samples into K folds
  2. For each fold $k = 1, \ldots, K$
     - Find a subset of predictors with strong univariate correlation with the class labels, using all samples except those in fold k.
     - Build a multivariate classifier using this set of predictors (excluding fold k)
     - Use the classifier to predict the class labels for the samples in fold k
- The class labels of the test fold should not be used at any point before predicting them in CV!

Remarks

- Bayesian model selection
  - asymptotically consistent
  - suitable when trying to find the "true" model from a set of distinct interpretable alternatives
  - heavy penalty on complexity $\longrightarrow$ may produce too sparse models for prediction
  - may be sensitive to the prior on the parameters
- Predictive model selection
  - no consistency guarantees
  - no need to assume a true model
  - less penalty for model complexity $\longrightarrow$ more complex models that may be suitable for prediction
- In practice people seem to use the two ways interchangeably for both goals: prediction and comparing hypotheses.

Model selection, summary

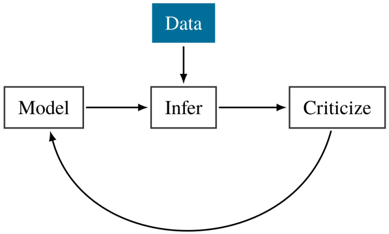
- There are two **different goals** for model selection: learning the correct model or selecting a model for prediction
- The **Bayesian model selection** gives probabilities of different models and may be more suitable if the goal is to learn the correct model.
- **Predictive model selection** criteria may be better if the goal is to use the model for prediction.
- BIC approximates Bayesian model selection, AIC and CV are related to predictive model selection.

Probabilistic programming using Edward

- **Edward:** a Python library for probabilistic modeling, inference, and criticism.
- **Probabilistic program**:
  - define the model and the inference separately as a program.
  - inference is done automatically
- Rapid experimentation with different models and/or inference algorithms!

Edward example

- Bayesian linear regression
  http://edwardlib.org/tutorials/supervised-regression
  1. **Simulate** data
  2. Define a **model**: Bayesian linear regression
  3. Specify and run **inference**
  4. **Visualize** model outputs
- Below we'll have a closer look at steps 2 and 3



Box's loop (Blei, 2014)

Edward example, model definition (1/2)

- Model definition (as written 'by hand')

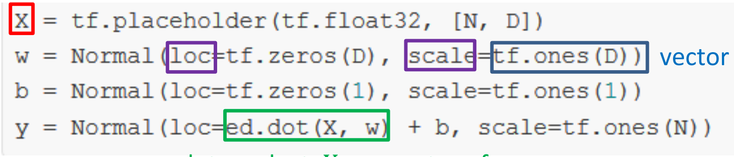
$$p(\mathbf{w}) = N(\mathbf{w}|\mathbf{0}, \sigma_w^2\mathbf{I}), \quad \sigma_w = 1$$
$$p(b) = N(b|0, \sigma_b^2), \quad \sigma_b = 1$$
$$p(\mathbf{y}|\mathbf{w}, b, \mathbf{X}) = \prod_{n=1}^{N} N(y_n|\mathbf{w}^\top\mathbf{x}_n + b, \sigma_y^2), \quad \sigma_y = 1$$

- Same in Edward

```
X = tf.placeholder(tf.float32, [N, D])
w = Normal(loc=tf.zeros(D), scale=tf.ones(D))
b = Normal(loc=tf.zeros(1), scale=tf.ones(1))
y = Normal(loc=ed.dot(X, w) + b, scale=tf.ones(N))
```

Edward example, model definition (2/2)

a 'placeholder' for N*D input data



```
X = tf.placeholder(tf.float32, [N, D])
w = Normal(loc=tf.zeros(D), scale=tf.ones(D))
b = Normal(loc=tf.zeros(1), scale=tf.ones(1))
y = Normal(loc=ed.dot(X, w) + b, scale=tf.ones(N))
```

vector of ones

dot product, Xw, a vector of length N, with elements $w^T x_n$

The normal distribution is parameterized by loc (mean) and scale (standard deviation)

Edward example, inference definition (1/2)

- Specify mean-field VB approximation (by hand)

$$q(\mathbf{w}) = \prod_{d=1}^{D} q(w_d) = \prod_{d=1}^{D} N(w_d|\mu_d, \tau_d^2),$$
$$q(b) = N(b|\mu_b, \tau_b^2),$$

μ_d, τ_d, μ_b, and τ_b are the **variational parameters**.

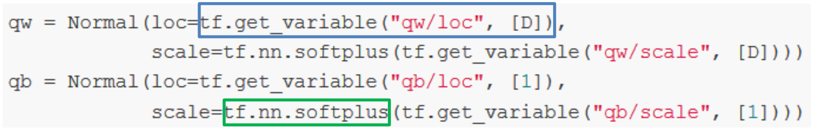
- In Edward

```
qw = Normal(loc=tf.get_variable("qw/loc", [D]),
            scale=tf.nn.softplus(tf.get_variable("qw/scale", [D])))
qb = Normal(loc=tf.get_variable("qb/loc", [1]),
            scale=tf.nn.softplus(tf.get_variable("qb/scale", [1])))
```

```
inference = ed.KLqp({w: qw, b: qb}, data={X: X_train, y: y_train})
inference.run(n_samples=5, n_iter=250)
```

Edward example, inference definition (2/2)

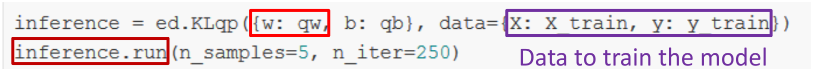
Variational parameters are represented as **Tensorflow Variables**. Here, the name of the D-vector is ’qw/loc’



```
qw = Normal(loc=tf.get_variable("qw/loc", [D]),
            scale=tf.nn.softplus(tf.get_variable("qw/scale", [D])))
qb = Normal(loc=tf.get_variable("qb/loc", [1]),
            scale=tf.nn.softplus(tf.get_variable("qb/scale", [1])))
```

A transformation to ensure STD stays positive

Random variable w’s distribution is approximated with factor qw.



```
inference = ed.KLqp({w: qw, b: qb}, data={X: X_train, y: y_train})
inference.run(n_samples=5, n_iter=250)
```

Data to train the model

Inference finds values for Tensorflow variables (in this case the variational parameters) which minimize KL(q|p)

Edward example, Tensorflow Variables

- Running inference optimizes the ELBO with respect to all Tensorflow Variables
  - here, the variational parameters
- We could use this, for example, to learn also the level of regularization (standard deviation of **w**), just replace

```
w = Normal(loc=tf.zeros(D), scale=tf.ones(D))
```

with

```
w_prior_std = tf.nn.softplus(tf.Variable(tf.constant(1.0)))
w = Normal(loc=tf.zeros(D), scale=w_prior_std * tf.ones(D))
```

Edward example, specifying data for inference

- Input X is defined as a Tensorflow **placeholder**

```
X = tf.placeholder(tf.float32, [N, D])
```

- Output y is defined as a **random variable** with a Normal distribution

```
y = Normal(loc=ed.dot(X, w) + b, scale=tf.ones(N))
```

- But both are given as data to the inference algorithm

```
inference = ed.KLqp({w: qw, b: qb}, data={X: X_train, y: y_train})
```

- Only y is modeled conditional on X, but X is not modeled itself!

Edward, further information

- Tran et al. (2017a). *Edward: A library for probabilistic modeling, inference, and criticism*. arXiv:1610.09787
    - The basics of Edward.
- Tran et al. (2017b). *Deep probabilistic programming*. ICLR 2017.
    - Example codes for many models, e.g., variational auto-encoders, Bayesian neural networks, language models, etc.
- Ranganath et al. (2014). *Black Box variational inference*. AISTATS 2014.
    - Explains the black-box variational inference used in Edward.
    - More on this next week.
- *http://edwardlib.org/*
- **Edward2**:
  https://github.com/tensorflow/probability/blob/master/tensorflow_probability/python/edward2/Upgrading_From_Edward_To_Edward2.md