Learning via Probabilistic Modeling

Piyush Rai

Introduction to Machine Learning (CS771A)

August 14, 2018
Recap: Linear Models and Linear Regression

- Linear model: Each output is a linearly weighted combination of the inputs
  \[ y_n = \sum_{d=1}^{D} w_d x_{nd} = w^\top x_n, \forall n \quad \Rightarrow \quad y = Xw \]

- The weights are the parameters of the model
- Can use linear models for doing linear regression. Amounts to fitting a line/plane to the data.

- Finding the best line/plane = finding \( w \) that minimizes the total error/loss of the fit
- This requires optimizing the loss w.r.t. \( w \)
Recap: Least Squares and Ridge Regression

- Least squares and ridge regression are both linear regression models based on squared loss.
- Least squares regression minimizes the simple sum of squared errors:
  \[
  \hat{w} = \arg \min_w \sum_{n=1}^N (y_n - w^\top x_n)^2 = \left( \sum_{n=1}^N x_n x_n^\top \right)^{-1} \sum_{n=1}^N y_n x_n = (X^\top X)^{-1} X^\top y
  \]
- Ridge regression minimizes the $\ell_2$ regularized sum of squared errors:
  \[
  \hat{w} = \arg \min_w \left[ \sum_{n=1}^N (y_n - w^\top x_n)^2 + \frac{\lambda}{2} w^\top w \right] = (X^\top X + \lambda I_D)^{-1} X^\top y
  \]
  Regularization helps prevent overfitting the training data.
- The $\ell_2$ regularization $w^\top w = \sum_{d=1}^D w_d^2$ promotes small individual weights.
Recap: Learning as Optimization

- Supervised learning is essentially a **function approximation** problem.
- Given training data \( \{(x_n, y_n)_{n=1}^{N}\} \), find a function \( f \) s.t. \( f(x_n) \approx y_n, \forall n \).
- In addition, we want \( f \) to be simple (i.e., want to regularize it).
- Can learn such a function \( f \) by solving the following **optimization problem**:

\[
\hat{f} = \arg \min_f \mathcal{L}_{\text{reg}}(f) = \arg \min_f \sum_{n=1}^{N} \ell(y_n, f(x_n)) + \lambda R(f)
\]

- Different supervised learning problems differ in the **choice of** \( f, \ell(.,.) \) and \( R(.) \):
  - \( f \) depends on the model (e.g., for linear models \( f(x) = w^T x \)).
  - \( \ell(.,.) \): loss function that measures the error of model’s prediction (e.g., squared loss).
  - \( R(.) \) denotes the regularizer chosen to make \( f \) simple (e.g., \( \ell_2 \) regularization).
A Brief Detour: Some Loss Functions

Some popular loss functions for regression problems\(^1\)

- **Squared Loss**: \((y - f(x))^2\)
- **Absolute Loss**: \(|y - f(x)|\)
- **Huber Loss**: Squared then Absolute
- **\(\epsilon\)-insensitive Loss**: \(|y - f(x)| - \epsilon\)

- **Absolute/Huber loss preferred if there are outliers in the data**
  - Less affected by large errors \(|y - f(x)|\) as compared to the squared loss

- **Overall objective function** = loss func + some regularizer (e.g., \(\ell_2, \ell_1\)), as we saw for ridge reg.

- Some objectives easy to optimize (convex and differentiable), some not so (e.g., non-differentiable)

- Will revisit many of these aspects when we talk about optimization techniques for ML

---

\(^1\) will look at loss functions for classification later when discussing classification in detail
Brief Detour: Inductive Bias of ML Algorithms

- No ML algorithm is “universally good”
- Should not expect it to work well on all datasets
- Each algorithm makes some assumption about data ("no free lunch")
  - Work best when assumptions are correct. May fail in other case.
- Inductive Bias: Set of assumptions made about outputs of previously unseen inputs
- Learning is impossible without making assumptions!
- Some common examples of such assumptions
  - Classes are separable by a large margin
  - The function is “smooth”
  - Only a few features are relevant for the prediction
Learning via Probabilistic Modeling

\[ p(x|c_1) \quad p(x|c_2) \]

\[ p(x) \]

\[ p(y|w, x) \]

\[ \mathbf{w}^\top \mathbf{x} \]

\[ x \]

\[ y \]
Probabilistic Modeling of Data

- Assume the data \( y = \{y_1, y_2, \ldots, y_N\} \) as generated from a probability model
  \[ y_n \sim p(y|\theta) \quad \forall n \]
- Each \( y_n \) assumed drawn from distribution \( p(y|\theta) \), with unknown parameters \( \theta \)
- We usually assume data to be independently & identically distributed (i.i.d.)

Some of the things we may be interested in

- **Parameter Estimation**: Estimate \( \theta \) given the observed data \( y \)
- **Prediction**: Compute predictive distribution \( p(y_*|y) \) for new data (or mean/variance of \( p(y_*|y) \))

Important: Pretty much any ML problem (sup/unsup) can be formulated like this
Parameter Estimation in Probabilistic Models

- Since data is i.i.d., the probability (or probability density) of observed data \( y = \{y_1, y_2, \ldots, y_N\} \)

\[
p(y|\theta) = p(y_1, y_2, \ldots, y_N|\theta) = \prod_{n=1}^{N} p(y_n|\theta)
\]

- \( p(y|\theta) \) also called the model’s likelihood, \( p(y_n|\theta) \) is likelihood w.r.t. a single data point

- The likelihood will be a function of the parameters \( \theta \)

How do we estimate the “best” model parameters \( \theta \)?

- One option: Find value of \( \theta \) that makes observed data most probable (i.e., most likely)

  - **Maximize** the likelihood \( p(y|\theta) \) w.r.t. \( \theta \): Maximum Likelihood Estimation (MLE)
Maximum Likelihood Estimation (MLE)

- We doing MLE, we typically maximize log-likelihood instead of the likelihood, which is easier (doesn’t affect the estimation because log is monotonic)

  ![Log-likelihood graph]

- Log-likelihood:
  \[
  \mathcal{L}(\theta) = \log p(y | \theta) = \log \prod_{n=1}^{N} p(y_n | \theta) = \sum_{n=1}^{N} \log p(y_n | \theta)
  \]

- Maximum Likelihood Estimation (MLE)
  \[
  \hat{\theta}_{MLE} = \arg \max_{\theta} \mathcal{L}(\theta) = \arg \max_{\theta} \sum_{n=1}^{N} \log p(y_n | \theta)
  \]

- Now this becomes an optimization problem w.r.t. \( \theta \)
Maximum Likelihood Estimation (MLE)

- Maximum Likelihood parameter estimation

$$\hat{\theta}_{MLE} = \arg \max_{\theta} \sum_{n=1}^{N} \log p(y_n \mid \theta) = \arg \min_{\theta} - \sum_{n=1}^{N} \log p(y_n \mid \theta)$$

- We thus also think of it as minimizing the **negative** log-likelihood (NLL)

$$\hat{\theta}_{MLE} = \arg \min_{\theta} NLL(\theta)$$

where $NLL(\theta) = - \sum_{n=1}^{N} \log p(y_n \mid \theta)$

- We can think of the negative log-likelihood as a loss function

- Thus MLE is equivalent to doing empirical risk (training data loss) minimization

- **Important:** This view relates/unifies the optimization and probabilistic modeling approaches

- Something is still missing (we will look at that shortly)
MLE: An Example

- Consider a sequence of $N$ coin toss outcomes (observations)
- Each observation $y_n$ is a binary random variable. Head = 1, Tail = 0
- Since each $y_n$ is binary, let's use a Bernoulli distribution to model it

$$p(y_n \mid \theta) = \theta^{y_n}(1-\theta)^{1-y_n}$$

- Here $\theta$ is the unknown parameter (probability of head). Want to learn $\theta$ using MLE
- Log-likelihood: $\sum_{n=1}^{N} \log p(y_n \mid \theta) = \sum_{n=1}^{N} y_n \log \theta + (1 - y_n) \log(1 - \theta)$
- Taking derivative of the log-likelihood w.r.t. $\theta$, and setting it to zero gives

$$\hat{\theta}_{MLE} = \frac{\sum_{n=1}^{N} y_n}{N}$$

- $\hat{\theta}_{MLE}$ in this example is simply the fraction of heads!

- What can go wrong with this approach (or MLE in general)?
  - We haven’t “regularized” $\theta$. Can do badly (i.e., overfit), e.g., if we don’t have enough data
Prior Distributions

- In probabilistic models, we can specify a prior distribution $p(\theta)$ on parameters.

The prior distribution expresses our *a priori* belief about the unknown $\theta$. Plays two key roles:

- The prior helps us specify that some values of $\theta$ are more likely than others.
- The prior also works as a regularizer for $\theta$ (we will see this soon).

Note: A uniform prior distribution is the same as using no prior!
Using a Prior in Parameter Estimation

- We can **combine** the prior $p(\theta)$ with the **likelihood** $p(y|\theta)$ using **Bayes rule** and define the **posterior distribution** over the parameters $\theta$

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)}$$

- Now, instead of doing MLE which **maximizes the likelihood**, we can find the $\theta$ that is **most likely** given the data, i.e., which maximizes the posterior probability $p(\theta|y)$

$$\hat{\theta}_{MAP} = \arg\max_{\theta} p(\theta|y)$$

- Note that the prior sort of “pulls” $\theta_{MLE}$ toward’s the prior distribution’s mean/mode
Maximum-a-Posteriori (MAP) Estimation

- We will work with the \( \log \) posterior probability (it is easier)

\[
\hat{\theta}_{\text{MAP}} = \arg \max_{\theta} p(\theta | y) = \arg \max_{\theta} \log p(\theta | y)
\]

\[
= \arg \max_{\theta} \log \frac{p(y | \theta) p(\theta)}{p(y)}
\]

\[
= \arg \max_{\theta} \log p(y | \theta) + \log p(\theta)
\]

\[
\hat{\theta}_{\text{MAP}} = \arg \max_{\theta} \sum_{n=1}^{N} \log p(y_n | \theta) + \log p(\theta)
\]

- Same as MLE with an extra \( \log \)-prior-distribution term (acts as a regularizer)

- Can also write the same as the following (equivalent) minimization problem

\[
\hat{\theta}_{\text{MAP}} = \arg \min_{\theta} NLL(\theta) - \log p(\theta)
\]

- When \( p(\theta) \) is a uniform prior, MAP reduces to MLE
MAP: An Example

- Let’s again consider the coin-toss problem (estimating the bias of the coin)
- Each likelihood term is Bernoulli: \( p(y_n|\theta) = \theta^{y_n}(1 - \theta)^{1-y_n} \)
- Since \( \theta \in (0, 1) \), we assume a Beta prior: \( \theta \sim \text{Beta}(\alpha, \beta) \)

\[
p(\theta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} \theta^{\alpha-1}(1 - \theta)^{\beta-1}
\]

- Note: \( \Gamma \) is the gamma function. \( \alpha, \beta \) are called hyperparameters of the prior

For Beta, using \( \alpha = \beta = 1 \) corresponds to using a uniform prior distribution
**MAP: An Example**

- The log posterior probability for the coin-toss model

\[
\sum_{n=1}^{N} \log p(y_n|\theta) + \log p(\theta)
\]

- Ignoring the constants w.r.t. \( \theta \), the log posterior probability simplifies to

\[
\sum_{n=1}^{N} \{y_n \log \theta + (1 - y_n) \log(1 - \theta)\} + (\alpha - 1) \log \theta + (\beta - 1) \log(1 - \theta)
\]

- Taking derivative w.r.t. \( \theta \) and setting to zero gives

\[
\hat{\theta}_{MAP} = \frac{\sum_{n=1}^{N} y_n + \alpha - 1}{N + \alpha + \beta - 2}
\]

- **Note:** For \( \alpha = 1, \beta = 1 \), i.e., \( p(\theta) = \text{Beta}(1,1) \) (which is equivalent to a uniform prior, hence no regularization). Thus, for \( \alpha = 1, \beta = 1 \), we get the same solution as \( \hat{\theta}_{MLE} \)

- **Note:** Hyperparameters of a prior distribution usually have intuitive meaning. E.g., in the coin-toss example, \( \alpha - 1, \beta - 1 \) are like “pseudo-observations” - expected numbers of heads and tails, respectively, before tossing the coin.
Inferring the **Full Posterior** (a.k.a. Fully Bayesian Inference)

- MLE/MAP only give us a **point estimate** of $\theta$. Doesn’t capture the uncertainty in $\theta$

- The Bayes rule (at least in theory) also allows us to **compute** the full posterior

\[
p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)} = \frac{p(y|\theta)p(\theta)}{\int p(y|\theta)p(\theta)d\theta}
\]

- In general, much harder problem than MLE/MAP! Easy if the prior and likelihood are “conjugate” to each other (then the posterior will then have the same “form” as the prior)

- Many pairs of distributions are conjugate to each other (e.g., Beta-Bernoulli, Gaussian is conjugate to itself, etc.). May refer to Wikipedia for a list of conjugate pairs of distributions
Fully Bayesian Inference

- Fully Bayesian inference fits naturally into an “online” learning setting

- Our belief about $\theta$ keeps getting updated as we see more and more data
Let’s again consider the coin-toss example

With Bernoulli likelihood and Beta prior (a conjugate pair), the posterior is also Beta (exercise)

$$\text{Beta}(\alpha + N_1, \beta + N_0)$$

where $N_1$ is the number of heads and $N_0 = N - N_1$ is the number of tails

Can verify the above by simply plugging in the expressions of likelihood and prior into the Bayes rule and identifying the form of resulting posterior (note: this may not always be easy)
Once $\theta$ is learned, we can use it to make predictions about the future observations.

E.g., for the coin-toss example, we can predict the probability of next toss being head.

This can be done using the MLE/MAP estimate, or using the full posterior (harder).

In the coin-toss example, $\theta_{MLE} = \frac{N_1}{N}$, $\theta_{MAP} = \frac{N_1 + \alpha - 1}{N + \alpha + \beta - 2}$, and $p(\theta|y) = \text{Beta}(\theta|\alpha + N_1, \beta + N_0)$.

Thus for this example (where observations are assumed to come from a Bernoulli):

- MLE prediction: $p(y_{N+1} = 1|y) = \int p(y_{N+1} = 1|\theta)p(\theta|y)d\theta \approx p(y_{N+1} = 1|\theta_{MLE}) = \theta_{MLE} = \frac{N_1}{N}$
- MAP prediction: $p(y_{N+1} = 1|y) = \int p(y_{N+1} = 1|\theta)p(\theta|y)d\theta \approx p(y_{N+1} = 1|\theta_{MAP}) = \theta_{MAP} = \frac{N_1 + \alpha - 1}{N + \alpha + \beta - 2}$
- Fully Bayesian: $p(y_{N+1} = 1|y) = \int p(y_{N+1} = 1|\theta)p(\theta|y)d\theta = \int \theta p(\theta|y)d\theta = \int \theta \text{Beta}(\theta|\alpha + N_1, \beta + N_0)d\theta = \frac{N_1 + \alpha}{N + \alpha + \beta}$

- Note that the fully Bayesian approach to prediction averages over all possible values of $\theta$, weighted by their respective posterior probabilities (easy in this example, but a hard problem in general)
Probabilistic Modeling: Summary

- A flexible way to model data by specifying a proper probabilistic model
- Likelihood corresponds to a loss function; prior corresponds to a regularizer
- Can choose likelihoods and priors based on the nature/property of data/parameters
- MLE estimation = unregularized loss function minimization
- MAP estimation = regularized loss function minimization
- Allows us to do **fully Bayesian learning**
  - Allows learning the **full distribution** of the parameters (note that MLE/MAP only give a “single best” answer as a **point estimate** of the parameters)
  - Makes more robust predictions by posterior averaging (rather than using a single point estimate)
  - Many other benefits, such as
    - Estimate of confidence in the model’s prediction (useful for doing **Active Learning**)
    - Can do automatic model selection, hyperparameter estimation, handle missing data, etc.
    - .. and many other benefits (a proper treatment deserves a separate course :)
- MLE/MAP estimation is also related to the optimization view of ML

Intro to Machine Learning  (CS771A)