Learning via Probabilistic Modeling

Piyush Rai

Introduction to Machine Learning (CS771A)

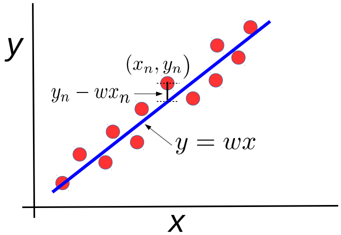
August 14, 2018

Recap: Linear Models and Linear Regression

- Linear model: Each output is a linearly weighted combination of the inputs

$$y_n = \sum_{d=1}^{D} w_d x_{nd} = \boldsymbol{w}^\top \boldsymbol{x}_n, \forall n \quad \Rightarrow \quad \boldsymbol{y} = \mathbf{X}\boldsymbol{w}$$

- 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 $\boldsymbol{w}$ that minimizes the total error/loss of the fit
- This requires optimizing the loss w.r.t. $\boldsymbol{w}$

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

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \sum_{n=1}^{N} (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 = (\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top)^{-1} \sum_{n=1}^{N} y_n \boldsymbol{x}_n = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \boldsymbol{y}$$

- Ridge regression minimizes the ℓ_2 regularized sum of squared errors

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} [\underbrace{\sum_{n=1}^{N} (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2}_{\text{training loss}} + \underbrace{\frac{\lambda}{2} \boldsymbol{w}^\top \boldsymbol{w}}_{\text{regularizer}}] = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_D)^{-1} \mathbf{X}^\top \boldsymbol{y}$$

- Regularization helps prevent overfitting the training data
- The ℓ_2 regularization $\boldsymbol{w}^\top \boldsymbol{w} = \sum_{d=1}^{D} w_d^2$ promotes small individual weights

Recap: Learning as Optimization

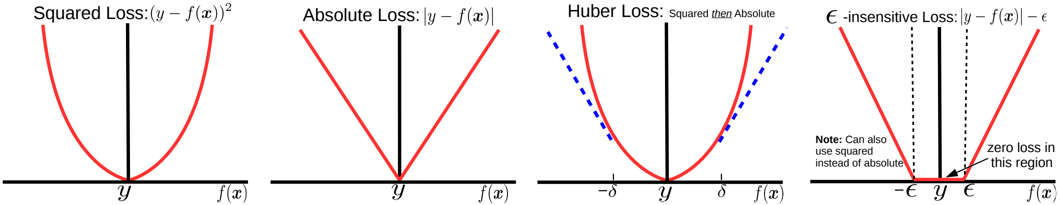
- Supervised learning is essentially a function approximation problem
- Given training data $\{(\boldsymbol{x}_n, y_n)_{n=1}^N\}$, find a function f s.t. $f(\boldsymbol{x}_n) \approx y_n, \forall n$
- In addition, we want f to be simple (i.e., want to regularize it)
- Can learn such a function f by solving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

$$\hat{f} = \arg\min_f \mathcal{L}_{reg}(f) = \arg\min_f \underbrace{\sum_{n=1}^N \ell(y_n, f(\boldsymbol{x}_n))}_{\text{training loss}} + \underbrace{\lambda R(f)}_{\text{regularization}}$$

- Different supervised learning problems differ in the choice of f, $\ell(.,.)$ and $R(.)$
  - f depends on the model (e.g., for linear models $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x}$)
  - $\ell(.,.)$: 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., ℓ_2 regularization)

A Brief Detour: Some Loss Functions

- Some popular loss functions for regression problems[1]



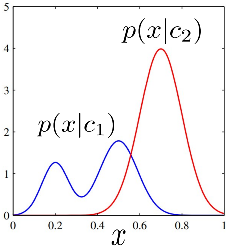
- Absolute/Huber loss preferred if there are outliers in the data
  - Less affected by large errors $|y - f(\boldsymbol{x})|$ as compared to the squared loss
- Overall objective function = loss func + some regularizer (e.g., ℓ_2, ℓ_1), as we saw for ridge reg.
- Some objectives easy to optimize (convex and differentiable), some not so (e.g., non-differentiable)
- Will revisit many of these aspects when we talk about optimization techniques for ML

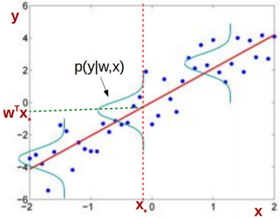
[1] will look at loss functions for classification later when discussing classification in detail

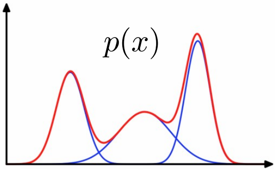
Brief Detour: Inductive Bias of ML Algorithms

- 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







Probabilistic Modeling of Data

- Assume the data $\boldsymbol{y} = \{y_1, y_2, \ldots, y_N\}$ as generated from a probability model

$$y_n \sim p(y|\theta) \qquad \forall n$$

- Each y_n assumed drawn from distribution $p(y|\theta)$, with unknown parameters θ
- We usually assume data to be independently & identically distributed (i.i.d.)

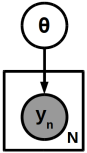


Plate Notation

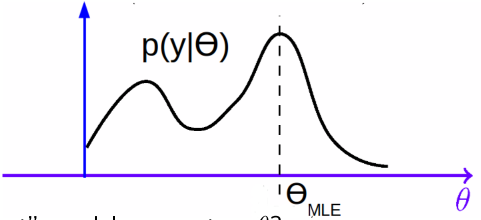
- Some of the things we may be interested in
  - Parameter Estimation: Estimate θ given the observed data $\boldsymbol{y}$
  - Prediction: Compute predictive distribution $p(y_*|\boldsymbol{y})$ for new data (or mean/variance of $p(y_*|\boldsymbol{y})$)
- **Important:** Pretty much any ML problem (sup/unsup) can be formulated like this

Parameter Estimation in Probabilistic Models

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

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

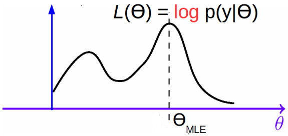
- $p(\boldsymbol{y}|\theta)$ also called the model's likelihood, $p(y_n|\theta)$ is likelihood w.r.t. a single data point
- The likelihood will be a function of the parameters θ



- How do we estimate the "best" model parameters θ?
- One option: Find value of θ that makes observed data most probable (i.e., most *likely*)
  - **Maximize** the likelihood $p(\boldsymbol{y}|\theta)$ w.r.t. θ: 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:

$$\mathcal{L}(\theta) = \log p(\boldsymbol{y} \mid \theta) = \log \prod_{n=1}^{N} p(y_n \mid \theta) = \sum_{n=1}^{N} \log p(y_n \mid \theta)$$

- Maximum Likelihood Estimation (MLE)

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

- Now this becomes an optimization problem w.r.t. θ

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)

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

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

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

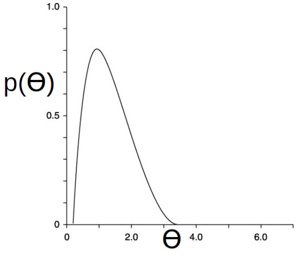
- Here θ is the unknown parameter (probability of head). Want to learn θ using MLE
- Log-likelihood: $\sum_{n=1}^{N} \log p(y_n \mid \theta) = \sum_{n=1}^{N} y_n \log \theta + (1-y_n)\log(1-\theta)$
- Taking derivative of the log-likelihood w.r.t. θ, and setting it to zero gives

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

- $\hat{\theta}_{MLE}$ in this example is simply the fraction of heads!
- What can go wrong with this approach (or MLE in general)?
  - We haven't "regularized" θ. 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

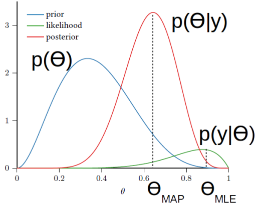


- The prior distribution expresses our *a priori* belief about the unknown θ. Plays two key roles
  - The prior helps us specify that some values of θ are more likely than others
  - The prior also works as a regularizer for θ (we will see this soon)
- Note: A uniform prior distribution is the same as using no prior!

Using a Prior in Parameter Estimation

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

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



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

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

- Note that the prior sort of "pulls" θ_{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)

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

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

- 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

$$\boxed{\hat{\theta}_{MAP} = \arg\min_{\theta} NLL(\theta) - \log p(\theta)}$$

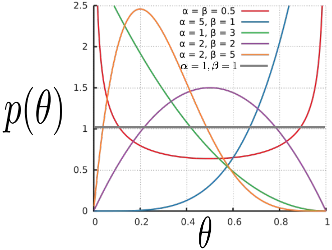
- When $p(\theta)$ is a uniform prior, MAP reduces to MLE

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: Γ is the gamma function. α, β 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. θ, the log posterior probability simplifies to

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

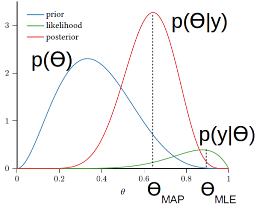
- Taking derivative w.r.t. θ and setting to zero gives

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

- **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 θ. Doesn't capture the uncertainty in θ



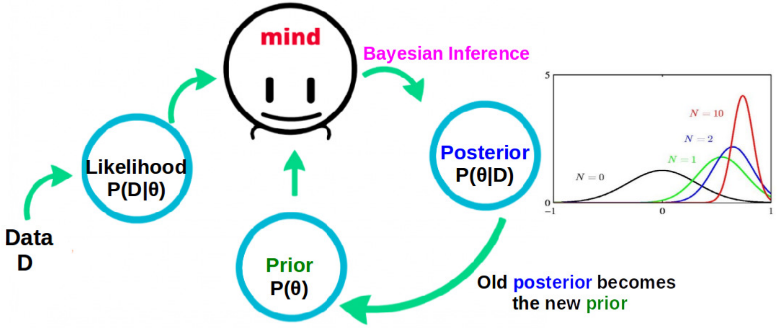
- The Bayes rule (at least in theory) also allows us to **compute** the full posterior

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

- 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 θ keeps getting updated as we see more and more data

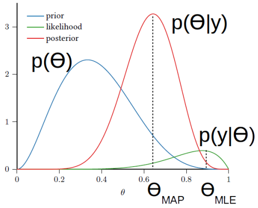
Fully Bayesian Inference: An Example

- 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)



Making Predictions: MLE/MAP/Fully Bayesian

- Once θ 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 $p(\theta|\mathbf{y}) = \text{Beta}(\theta|\alpha + N_1, \beta + N_0)$
- Thus for this example (where observations are assumed to come from a Bernoulli)

$$\text{MLE prediction:} \quad p(y_{N+1} = 1|\mathbf{y}) = \int p(y_{N+1} = 1|\theta)p(\theta|\mathbf{y})d\theta \approx p(y_{N+1} = 1|\theta_{MLE}) = \theta_{MLE} = \frac{N_1}{N}$$

$$\text{MAP prediction:} \quad p(y_{N+1} = 1|\mathbf{y}) = \int p(y_{N+1} = 1|\theta)p(\theta|\mathbf{y})d\theta \approx p(y_{N+1} = 1|\theta_{MAP}) = \theta_{MAP} = \frac{N_1 + \alpha - 1}{N + \alpha + \beta - 2}$$

$$\text{Fully Bayesian:} \quad p(y_{N+1} = 1|\mathbf{y}) = \int p(y_{N+1} = 1|\theta)p(\theta|\mathbf{y})d\theta = \int \theta p(\theta|\mathbf{y})d\theta = \int \theta \text{Beta}(\theta|\alpha + N_1, \beta + N_0)d\theta = \frac{N_1 + \alpha}{N + \alpha + \beta}$$

- Note that the fully Bayesian approach to prediction averages over all possible values of θ, 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