# Learning via Probabilistic Modeling, Probabilistic Linear Regression

Piyush Rai

Machine Learning (CS771A)

Aug 12, 2016

# Some Announcements

- Homework 1 out tomorrow. Will be due in two weeks.
  - Will cover topics from up to the previous lecture
- Project discussion next week.
- Class TA's finalized. Will soon announce their/mine office hours
- Watch out the class webpage regularly for readings/reference materials
- Please participate on Piazza actively. Share and learn from each other.

# Recap

# Learning as Optimization

- Supervised learning problem with training data $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$
- Goal: Find $f : \boldsymbol{x} \to y$ that fits the training data well and is also "simple"
- The function $f$ is learned by solving the following optimization problem

$$\hat{f} = \arg\min_{f} \sum_{n=1}^{N} \ell(y_n, f(\boldsymbol{x}_n)) + \lambda R(f)$$

# Learning as Optimization

- Supervised learning problem with training data $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$
- Goal: Find $f : \boldsymbol{x} \rightarrow y$ that fits the training data well and is also "simple"
- The function $f$ is learned by solving the following optimization problem

$$\boxed{\hat{f} = \arg\min_{f} \sum_{n=1}^{N} \ell(y_n, f(\boldsymbol{x}_n)) + \lambda R(f)}$$

- The objective is a sum of the empirical training loss and a regularizer term

# Learning as Optimization

- Supervised learning problem with training data $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$
- Goal: Find $f : \boldsymbol{x} \rightarrow y$ that fits the training data well and is also "simple"
- The function $f$ is learned by solving the following optimization problem

$$\hat{f} = \arg\min_f \sum_{n=1}^{N} \ell(y_n, f(\boldsymbol{x}_n)) + \lambda R(f)$$

- The objective is a sum of the empirical training loss and a regularizer term
- $\ell(y_n, f(\boldsymbol{x}_n))$ denotes the **loss function**: Error $f$ makes on example $(\boldsymbol{x}_n, y_n)$
- The regularizer $R(f)$ is a measure of complexity of the function $f$

# Learning as Optimization

- Supervised learning problem with training data $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$
- Goal: Find $f : \boldsymbol{x} \rightarrow y$ that fits the training data well and is also "simple"
- The function $f$ is learned by solving the following optimization problem

$$\hat{f} = \arg\min_f \sum_{n=1}^{N} \ell(y_n, f(\boldsymbol{x}_n)) + \lambda R(f)$$

- The objective is a sum of the empirical training loss and a regularizer term
- $\ell(y_n, f(\boldsymbol{x}_n))$ denotes the **loss function**: Error $f$ makes on example $(\boldsymbol{x}_n, y_n)$
- The regularizer $R(f)$ is a measure of complexity of the function $f$
- This is called **Regularized** Empirical Risk Minimization

# Learning as Optimization

- Supervised learning problem with training data $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$
- Goal: Find $f : \boldsymbol{x} \to y$ that fits the training data well and is also "simple"
- The function $f$ is learned by solving the following optimization problem

$$\hat{f} = \arg\min_f \sum_{n=1}^{N} \ell(y_n, f(\boldsymbol{x}_n)) + \lambda R(f)$$

- The objective is a sum of the empirical training loss and a regularizer term
- $\ell(y_n, f(\boldsymbol{x}_n))$ denotes the **loss function**: Error $f$ makes on example $(\boldsymbol{x}_n, y_n)$
- The regularizer $R(f)$ is a measure of complexity of the function $f$
- This is called **Regularized** Empirical Risk Minimization
- Regularization hyperparameter $\lambda$ controls the amount of regularization

# $\ell_2$ Regularized Linear Regression: Ridge Regression

- Linear regression model $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x}$
- Loss function: squared loss, regularizer: $\ell_2$ norm of $\boldsymbol{w}$
- The resulting Ridge Regression problem is solved as

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

# $\ell_2$ Regularized Linear Regression: Ridge Regression

- Linear regression model $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x}$
- Loss function: squared loss, regularizer: $\ell_2$ norm of $\boldsymbol{w}$
- The resulting Ridge Regression problem is solved as

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

- A nice, convex objective function, has a unique global minima

# $\ell_2$ Regularized Linear Regression: Ridge Regression

- Linear regression model $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x}$
- Loss function: squared loss, regularizer: $\ell_2$ norm of $\boldsymbol{w}$
- The resulting Ridge Regression problem is solved as

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

- A nice, convex objective function, has a unique global minima
- Note: $\lambda = 0$ gives the ordinary least squares solution (no regularization)

# $\ell_2$ Regularized Linear Regression: Ridge Regression

- Linear regression model $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x}$
- Loss function: squared loss, regularizer: $\ell_2$ norm of $\boldsymbol{w}$
- The resulting Ridge Regression problem is solved as

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

- A nice, convex objective function, has a unique global minima
- Note: $\lambda = 0$ gives the ordinary least squares solution (no regularization)
- Can take derivative w.r.t. $\boldsymbol{w}$, set it to zero, and get simple, closed form soln

$$\boxed{\boldsymbol{w} = (\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top + \lambda \mathbf{I}_D)^{-1} \sum_{n=1}^{N} y_n \boldsymbol{x}_n = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_D)^{-1} \mathbf{X}^\top \boldsymbol{y}}$$

# $\ell_2$ Regularized Linear Regression: Ridge Regression

- Linear regression model $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x}$
- Loss function: squared loss, regularizer: $\ell_2$ norm of $\boldsymbol{w}$
- The resulting Ridge Regression problem is solved as

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

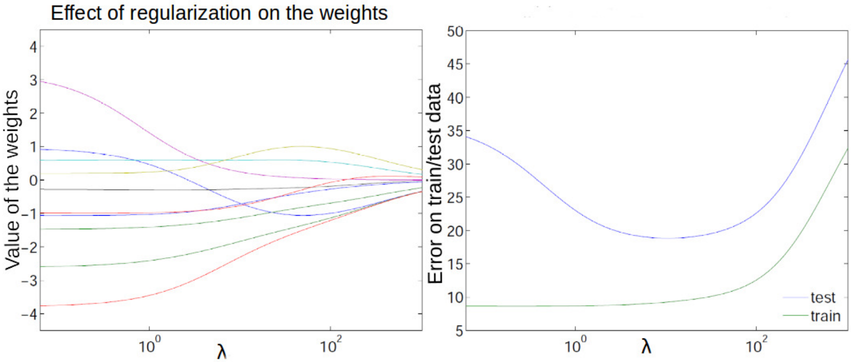
- A nice, convex objective function, has a unique global minima
- Note: $\lambda = 0$ gives the ordinary least squares solution (no regularization)
- Can take derivative w.r.t. $\boldsymbol{w}$, set it to zero, and get simple, closed form soln

$$\boldsymbol{w} = (\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top + \lambda \mathbf{I}_D)^{-1} \sum_{n=1}^{N} y_n \boldsymbol{x}_n = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_D)^{-1} \mathbf{X}^\top \boldsymbol{y}$$

- Can also used iterative methods (e.g., gradient-descent) to optimize the objective function and solve for $\boldsymbol{w}$ (for better efficiency)

# Ridge Regression: Effect of Regularization

- Consider ridge regression on some data with 10 features (thus the weight vector $\boldsymbol{w}$ has 10 components)



# 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 data point $y_n$ is a random variable drawn from distribution $p(y|\theta)$
- $\theta$ denotes the parameters of the probability distribution

# 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 data point $y_n$ is a random variable drawn from distribution $p(y|\theta)$
- $\theta$ denotes the parameters of the probability distribution
- Assume the observations to be independently & identically distributed (i.i.d.)

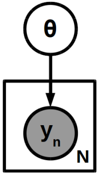



Plate Notation

# 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 data point $y_n$ is a random variable drawn from distribution $p(y|\theta)$
- $\theta$ denotes the parameters of the probability distribution
- Assume the observations to be independently & identically distributed (i.i.d.)

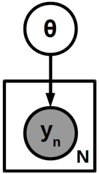



Plate Notation

- We wish to learn the parameters $\theta$ using the data $\boldsymbol{y} = \{y_1, y_2, \ldots, y_N\}$

# 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 data point $y_n$ is a random variable drawn from distribution $p(y|\theta)$
- $\theta$ denotes the parameters of the probability distribution
- Assume the observations to be independently & identically distributed (i.i.d.)

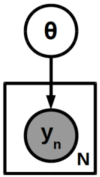



Plate Notation

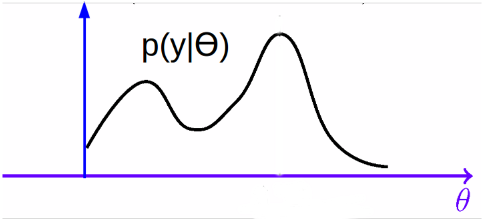
- We wish to learn the parameters $\theta$ using the data $\boldsymbol{y} = \{y_1, y_2, \ldots, y_N\}$
- Almost any learning problem can be formulated like this

# Parameter Estimation in Probabilistic Models

- Since data is i.i.d., the probability of observing 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 likelihood, $p(y_n|\theta)$ is lik. w.r.t. a single data point
- The likelihood will be a function of the parameters

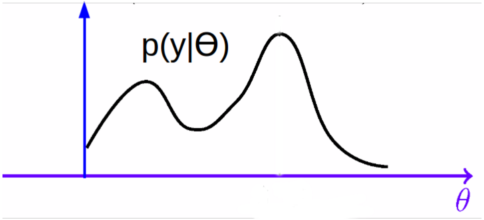


# Parameter Estimation in Probabilistic Models

- Since data is i.i.d., the probability of observing 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 likelihood, $p(y_n|\theta)$ is lik. w.r.t. a single data point
- The likelihood will be a function of the parameters



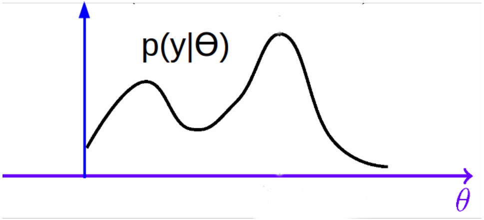

- How do we estimate the "best" model parameters $\theta$?

# Parameter Estimation in Probabilistic Models

- Since data is i.i.d., the probability of observing 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 likelihood, $p(y_n|\theta)$ is lik. w.r.t. a single data point
- The likelihood will be a function of the parameters



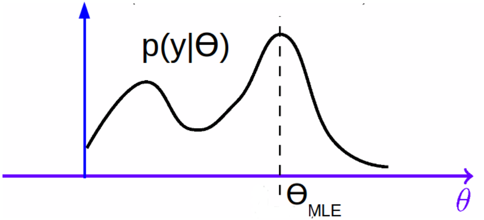

- How do we estimate the "best" model parameters $\theta$?
- One option: Find value of $\theta$ that makes observed data most probable
  - **Maximize** the likelihood $p(\boldsymbol{y}|\theta)$ w.r.t. $\theta$ (Maximum Likelihood Estimation)

# Parameter Estimation in Probabilistic Models

- Since data is i.i.d., the probability of observing data $\boldsymbol{y} = \{y_1, y_2, \ldots, y_N\}$

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

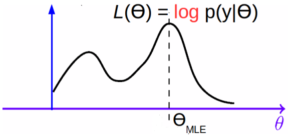
- $p(\boldsymbol{y}|\theta)$ also called the likelihood, $p(y_n|\theta)$ is lik. w.r.t. a single data point
- The likelihood will be a function of the parameters




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

# 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)

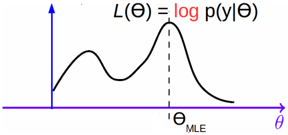



- Log-likelihood:

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

# Maximum Likelihood Estimation (MLE)

- 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:

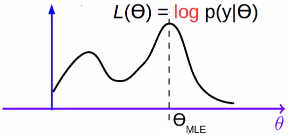
$$\mathcal{L}(\theta) = \log p(\boldsymbol{y} \mid \theta) = \log \prod_{n=1}^{N} p(y_n \mid \theta) = \sum_{n=1}^{N} \log p(y_n \mid \theta)$$

- Maximum Likelihood Estimation (MLE)

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

# 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)

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

- Now this becomes an optimization problem w.r.t. $\theta$

# Maximum Likelihood Estimation (MLE)

- Maximum Likelihood parameter estimation

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

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

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

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

# Maximum Likelihood Estimation (MLE)

- Maximum Likelihood parameter estimation

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

- We can 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

# Maximum Likelihood Estimation (MLE)

- Maximum Likelihood parameter estimation

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

- We can 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 (loss) minimization

# Maximum Likelihood Estimation (MLE)

- Maximum Likelihood parameter estimation

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

- We can 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 (loss) minimization
- This view relates the optimization and probabilistic modeling approaches

# Maximum Likelihood Estimation (MLE)

- Maximum Likelihood parameter estimation

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

- We can 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 (loss) minimization
- This view relates 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 $\theta$ to be probability of a head. Want to learn $\theta$ using MLE

# 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 $\theta$ to be probability of a head. Want to learn $\theta$ using MLE
- Log-likelihood: $\sum_{n=1}^{N} \log p(y_n \mid \theta)$

# 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 $\theta$ to be probability of a head. Want to learn $\theta$ using MLE
- Log-likelihood: $\sum_{n=1}^{N} \log p(y_n \mid \theta) = \sum_{n=1}^{N} y_n \log\theta + (1-y_n)\log(1-\theta)$

# 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 $\theta$ to be probability of a 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

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

# 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 $\theta$ to be probability of a 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!

# 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 $\theta$ to be probability of a 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)?

# 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

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

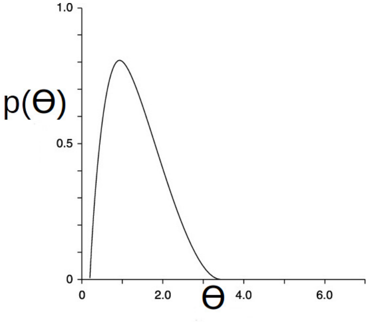
- Here $\theta$ to be probability of a head. Want to learn $\theta$ using MLE
- Log-likelihood: $\sum_{n=1}^{N} \log p(y_n \mid \theta) = \sum_{n=1}^{N} y_n \log\theta + (1-y_n)\log(1-\theta)$
- Taking derivative of the log-likelihood w.r.t. $\theta$, and setting it to zero gives

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

- $\hat{\theta}_{MLE}$ 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) if there are outliers or if we don't have enough data to learn $\theta$ reliably.

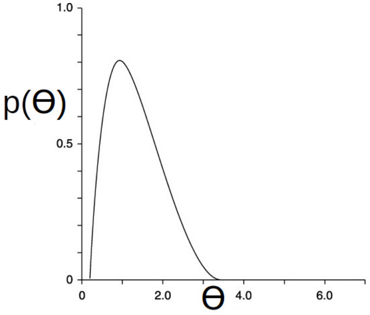
# Prior Distributions

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



# Prior Distributions

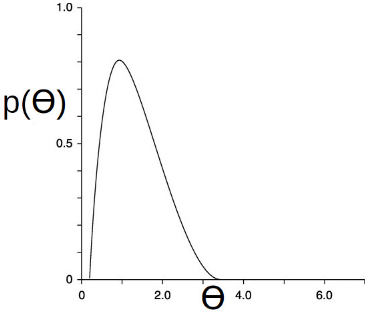
- In probabilistic models, we can specify a prior distribution $p(\theta)$ on parameters




- The prior distribution plays two key roles

# Prior Distributions

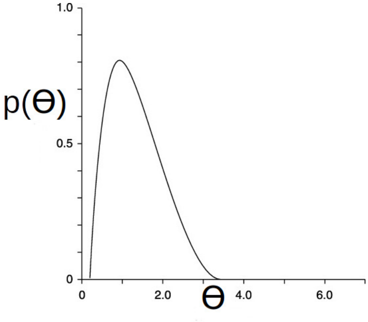
- In probabilistic models, we can specify a prior distribution $p(\theta)$ on parameters




- The prior distribution plays two key roles
  - The prior helps us specify that some values of $\theta$ are more likely than others

# Prior Distributions

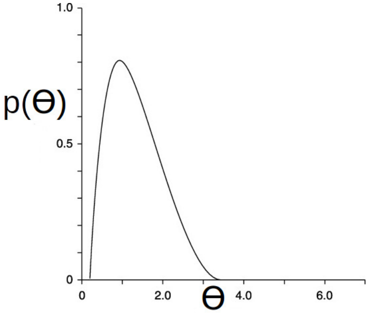
- In probabilistic models, we can specify a prior distribution $p(\theta)$ on parameters




- The prior distribution 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)

# Prior Distributions

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

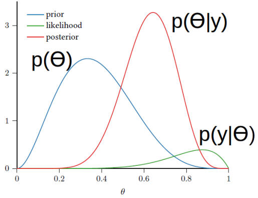



- The prior distribution plays two key roles
  - The prior helps us specify that some values of $\theta$ are more likely than others
  - The prior also works as a regularizer for $\theta$ (we will see this soon)
- Note: A uniform prior distribution is the same as using no prior!

# Using a Prior in Parameter Estimation

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

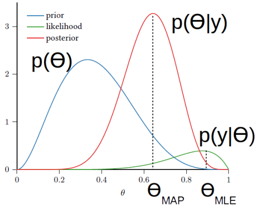
$$p(\theta|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\theta)p(\theta)}{p(\boldsymbol{y})}$$



# 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 $\theta$

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




- Now, instead of doing MLE which maximizes the likelihood, we can find the $\theta$ that maximizes the posterior probability $p(\theta|\boldsymbol{y})$

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

# Maximum-a-Posteriori (MAP) Estimation

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

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

# Maximum-a-Posteriori (MAP) Estimation

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

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

# Maximum-a-Posteriori (MAP) Estimation

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

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

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

# Maximum-a-Posteriori (MAP) Estimation

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

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

# Maximum-a-Posteriori (MAP) Estimation

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

$$\begin{aligned}\hat{\theta}_{MAP} = \arg\max_{\theta} p(\theta|\mathbf{y}) &= \arg\max_{\theta} \log p(\theta|\mathbf{y}) \\ &= \arg\max_{\theta} \log \frac{p(\mathbf{y}|\theta)p(\theta)}{p(\mathbf{y})} \\ &= \arg\max_{\theta} \log p(\mathbf{y}|\theta) + \log p(\theta)\end{aligned}$$

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

# Maximum-a-Posteriori (MAP) Estimation

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

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

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

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

# Maximum-a-Posteriori (MAP) Estimation

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

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

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

- 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

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

# Maximum-a-Posteriori (MAP) Estimation

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

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

$$= \arg\max_{\theta} \log \frac{p(\mathbf{y}|\theta)p(\theta)}{p(\mathbf{y})}$$

$$= \arg\max_{\theta} \log p(\mathbf{y}|\theta) + \log p(\theta)$$

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

- 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

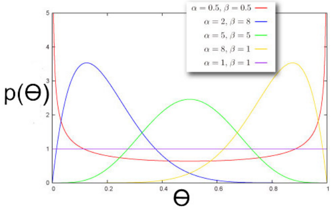
- Let's again consider the coin-toss problem (estimating the bias of the coin)
- Each likelihood term is Bernoulli: $p(y_n|\theta) = \theta^{y_n}(1-\theta)^{1-y_n}$

# 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: $\theta \sim \text{Beta}(\alpha, \beta)$

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

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

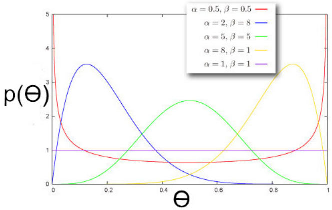


# MAP: An Example

- 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: $\theta \sim \text{Beta}(\alpha, \beta)$

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

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




- 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

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

# 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

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

# 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

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

# 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 regularizer), we get the same solution as $\hat{\theta}_{MLE}$

# 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

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

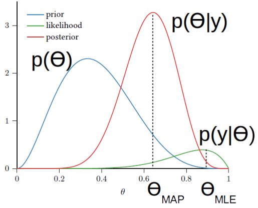
- Taking derivative w.r.t. $\theta$ and setting to zero gives

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

- **Note:** For $\alpha = 1, \beta = 1$, i.e., $p(\theta) = \text{Beta}(1,1)$ (which is equivalent to a uniform prior, hence no regularizer), 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

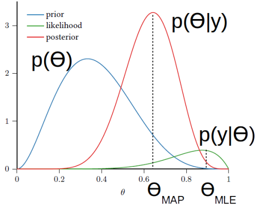
# Inferring the full Posterior (aka Bayesian Inference)

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



# Inferring the full Posterior (aka Bayesian Inference)

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

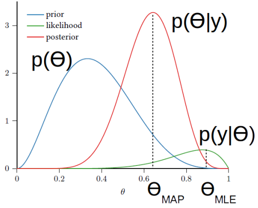



- The Bayes rule (at least in theory) also allows us to **compute** the full posterior distribution of $\theta$

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

# Inferring the full Posterior (aka Bayesian Inference)

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



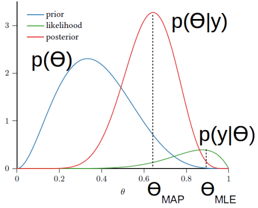

- The Bayes rule (at least in theory) also allows us to **compute** the full posterior distribution of $\theta$

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

- A much harder problem than MLE/MAP! Easy if the prior is "conjugate" to the likelihood (the posterior will then have the same "form" as the prior - basically, the same type of distribution)

# Inferring the full Posterior (aka Bayesian Inference)

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



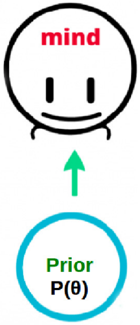

- The Bayes rule (at least in theory) also allows us to **compute** the full posterior distribution of $\theta$

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

- A much harder problem than MLE/MAP! Easy if the prior is "conjugate" to the likelihood (the posterior will then have the same "form" as the prior - basically, the same type of distribution)
- A very nice aspect is that Bayesian inference is naturally "online" (the posterior can be treated as a prior for next batch of data and updated recursively as we see more and more data)

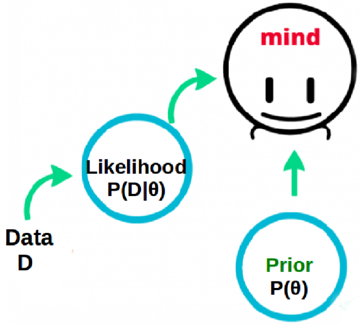
# Bayesian Inference

- Bayesian inference fits naturally into an "online" learning setting



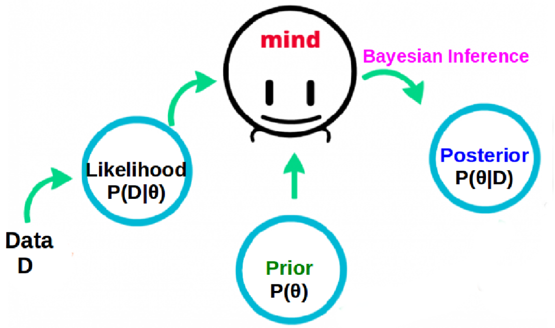
# Bayesian Inference

- Bayesian inference fits naturally into an "online" learning setting



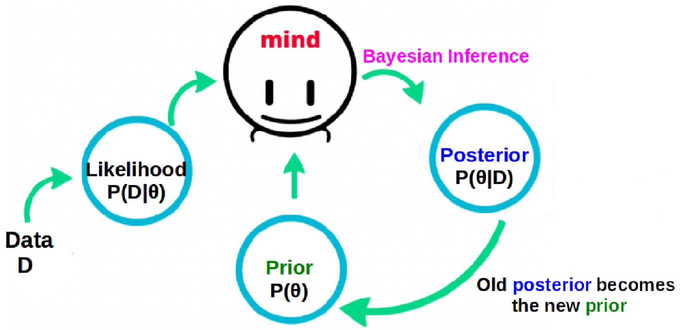
# Bayesian Inference

- Bayesian inference fits naturally into an "online" learning setting



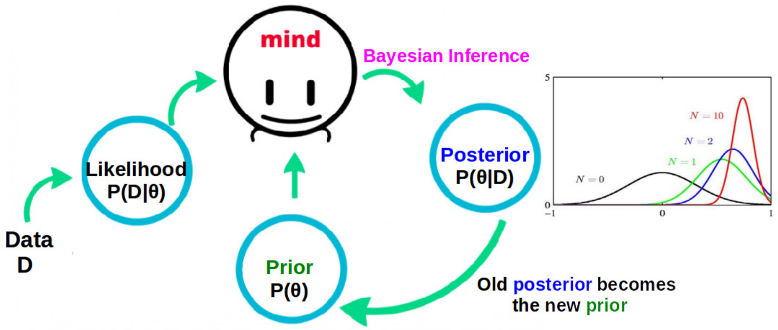
# Bayesian Inference

- Bayesian inference fits naturally into an "online" learning setting



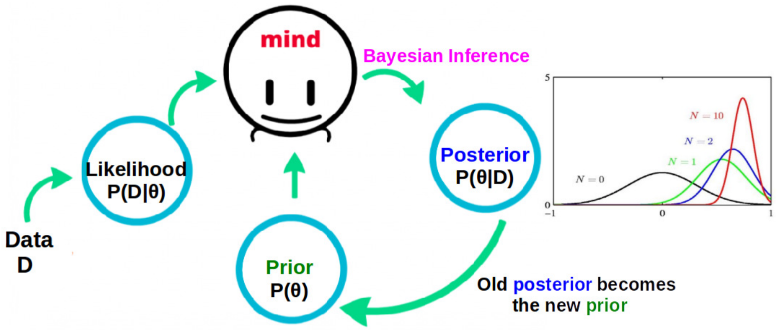
# Bayesian Inference

- Bayesian inference fits naturally into an "online" learning setting



# Bayesian Inference

- Bayesian inference fits naturally into an "online" learning setting




- Our belief about $\theta$ keeps getting updated as we see more and more data

# 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

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

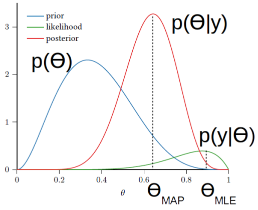
where $N_1$ is the number of heads and $N_0 = N - N_1$ is the number of tails

# 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

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

  where $N_1$ is the number of heads and $N_0 = N - N_1$ is the number of tails
- Exercise: 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/Bayesian

- Once $\theta$ is learned, we can use it to make predictions about the future observations

# Making Predictions: MLE/MAP/Bayesian

- 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

# Making Predictions: MLE/MAP/Bayesian

- 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 by using the MLE/MAP estimate, or by using the full distribution (harder)

# Making Predictions: MLE/MAP/Bayesian

- 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 by using the MLE/MAP estimate, or by using the full distribution (harder)
- In the coin-toss example, $\theta_{MLE} = \frac{N_1}{N}$, $\theta_{MAP} = \frac{N_1+\alpha-1}{N+\alpha+\beta-2}$, and $p(\theta|\boldsymbol{y}) = \text{Beta}(\theta|\alpha + N_1, \beta + N_0)$

# Making Predictions: MLE/MAP/Bayesian

- 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 by using the MLE/MAP estimate, or by using the full distribution (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)

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

# Making Predictions: MLE/MAP/Bayesian

- 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 by using the MLE/MAP estimate, or by using the full distribution (harder)
- In the coin-toss example, $\theta_{MLE} = \frac{N_1}{N}$, $\theta_{MAP} = \frac{N_1+\alpha-1}{N+\alpha+\beta-2}$, and $p(\theta|\boldsymbol{y}) = \text{Beta}(\theta|\alpha + N_1, \beta + N_0)$
- Thus for this example (where observations are assumed to come from a Bernoulli)

$$\begin{aligned} \text{MLE prediction:} \quad p(y_{N+1}|\boldsymbol{y}) &= p(y_{N+1} = 1|\theta_{MLE}) = \theta_{MLE} = \frac{N_1}{N} \\ \text{MAP prediction:} \quad p(y_{N+1}|\boldsymbol{y}) &= p(y_{N+1} = 1|\theta_{MAP}) = \theta_{MAP} = \frac{N_1 + \alpha - 1}{N + \alpha + \beta - 2} \end{aligned}$$

# Making Predictions: MLE/MAP/Bayesian

- 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 by using the MLE/MAP estimate, or by using the full distribution (harder)
- In the coin-toss example, $\theta_{MLE} = \frac{N_1}{N}$, $\theta_{MAP} = \frac{N_1+\alpha-1}{N+\alpha+\beta-2}$, and $p(\theta|\boldsymbol{y}) = \text{Beta}(\theta|\alpha+N_1, \beta+N_0)$
- Thus for this example (where observations are assumed to come from a Bernoulli)

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

# Making Predictions: MLE/MAP/Bayesian

- 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 by using the MLE/MAP estimate, or by using the full distribution (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)

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

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

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

- 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 Linear Regression

# Linear Regression: A Probabilistic View

- Given: $N$ training examples $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$, features: $\boldsymbol{x}_n \in \mathbb{R}^D$, response $y_n \in \mathbb{R}$
- Probabilistic view: responses $y_n$'s are generated from a probabilistic model

# Linear Regression: A Probabilistic View

- Given: $N$ training examples $\{\boldsymbol{x}_n, y_n\}_{n=1}^{N}$, features: $\boldsymbol{x}_n \in \mathbb{R}^D$, response $y_n \in \mathbb{R}$
- Probabilistic view: responses $y_n$'s are generated from a probabilistic model
- Assume a "noisy" linear model with regression weight vector $\boldsymbol{w} \in \mathbb{R}^D$:

$$\boxed{y_n = \boldsymbol{w}^\top \boldsymbol{x}_n + \epsilon_n}$$

- Gaussian noise: $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$, $\sigma^2$: variance of Gaussian noise

# Linear Regression: A Probabilistic View

- Given: $N$ training examples $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$, features: $\boldsymbol{x}_n \in \mathbb{R}^D$, response $y_n \in \mathbb{R}$
- Probabilistic view: responses $y_n$'s are generated from a probabilistic model
- Assume a "noisy" linear model with regression weight vector $\boldsymbol{w} \in \mathbb{R}^D$:

$$\boxed{y_n = \boldsymbol{w}^\top \boldsymbol{x}_n + \epsilon_n}$$

- Gaussian noise: $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$, $\sigma^2$: variance of Gaussian noise
- Thus each $y_n$ can be thought of as drawn from a Gaussian, as follows

$$\boxed{y_n \sim \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}_n, \sigma^2)}$$

# Linear Regression: A Probabilistic View

- Given: $N$ training examples $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$, features: $\boldsymbol{x}_n \in \mathbb{R}^D$, response $y_n \in \mathbb{R}$
- Probabilistic view: responses $y_n$'s are generated from a probabilistic model
- Assume a "noisy" linear model with regression weight vector $\boldsymbol{w} \in \mathbb{R}^D$:

$$\boxed{y_n = \boldsymbol{w}^\top \boldsymbol{x}_n + \epsilon_n}$$

- Gaussian noise: $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$, $\sigma^2$: variance of Gaussian noise
- Thus each $y_n$ can be thought of as drawn from a Gaussian, as follows

$$\boxed{y_n \sim \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}_n, \sigma^2)}$$

- Goal: Learn weight vector $\boldsymbol{w}$ (note: $\sigma^2$ assumed known but can be learned)

# Linear Regression: A Probabilistic View

- Given: $N$ training examples $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$, features: $\boldsymbol{x}_n \in \mathbb{R}^D$, response $y_n \in \mathbb{R}$
- Probabilistic view: responses $y_n$'s are generated from a probabilistic model
- Assume a "noisy" linear model with regression weight vector $\boldsymbol{w} \in \mathbb{R}^D$:

$$\boxed{y_n = \boldsymbol{w}^\top \boldsymbol{x}_n + \epsilon_n}$$

- Gaussian noise: $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$, $\sigma^2$: variance of Gaussian noise
- Thus each $y_n$ can be thought of as drawn from a Gaussian, as follows

$$\boxed{y_n \sim \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}_n, \sigma^2)}$$

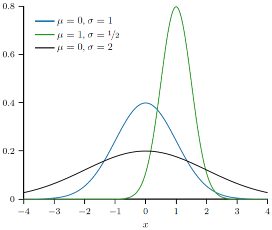
- Goal: Learn weight vector $\boldsymbol{w}$ (note: $\sigma^2$ assumed known but can be learned)
- Let's look at both MLE and MAP estimation for this probabilistic model

# Gaussian Distribution: Brief Review

# Univariate Gaussian Distribution

- Distribution over real-valued scalar r.v. $x$
- Defined by a scalar **mean** $\mu$ and a scalar **variance** $\sigma^2$
- Distribution defined as

$$\mathcal{N}(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

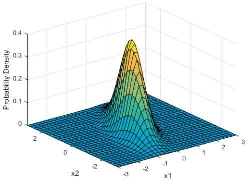



- Mean: $\mathbb{E}[x] = \mu$
- Variance: $\text{var}[x] = \sigma^2$

# Multivariate Gaussian Distribution

- Distribution over a multivariate r.v. vector $\mathbf{x} \in \mathbb{R}^D$ of real numbers
- Defined by a **mean vector** $\boldsymbol{\mu} \in \mathbb{R}^D$ and a $D \times D$ **covariance matrix** $\boldsymbol{\Sigma}$

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$$




- The covariance matrix $\boldsymbol{\Sigma}$ must be symmetric and positive definite
  - All eigenvalues are positive
  - $\boldsymbol{z}^\top \boldsymbol{\Sigma} \boldsymbol{z} > 0$ for any real vector $\boldsymbol{z}$

# MLE for Probabilistic Linear Regression

- Assuming Gaussian distributed responses $y_n$, we have

$$p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}_n, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2}{2\sigma^2}\right\}$$

# MLE for Probabilistic Linear Regression

- Assuming Gaussian distributed responses $y_n$, we have

$$p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}_n, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2}{2\sigma^2}\right\}$$

- Thus the likelihood (assuming i.i.d. responses) or *probability* of data:

$$p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) = \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) = \left(\frac{1}{2\pi\sigma^2}\right)^{\frac{N}{2}} \exp\left\{-\sum_{n=1}^{N} \frac{(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2}{2\sigma^2}\right\}$$

- Note: $\boldsymbol{x}_n$ (features) assumed given/fixed. Only modeling the response $y_n$

# MLE for Probabilistic Linear Regression

- Assuming Gaussian distributed responses $y_n$, we have

$$p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}_n, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2}{2\sigma^2}\right\}$$

- Thus the likelihood (assuming i.i.d. responses) or *probability* of data:

$$p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) = \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) = \left(\frac{1}{2\pi\sigma^2}\right)^{\frac{N}{2}} \exp\left\{-\sum_{n=1}^{N} \frac{(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2}{2\sigma^2}\right\}$$

- Note: $\boldsymbol{x}_n$ (features) assumed given/fixed. Only modeling the response $y_n$
- **Log-likelihood** (ignoring constants w.r.t. $\boldsymbol{w}$)

$$\boxed{\log p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) \propto -\frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2}$$

# MLE for Probabilistic Linear Regression

- Assuming Gaussian distributed responses $y_n$, we have

$$p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}_n, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2}{2\sigma^2}\right\}$$

- Thus the likelihood (assuming i.i.d. responses) or *probability* of data:

$$p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) = \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) = \left(\frac{1}{2\pi\sigma^2}\right)^{\frac{N}{2}} \exp\left\{-\sum_{n=1}^{N} \frac{(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2}{2\sigma^2}\right\}$$

- Note: $\boldsymbol{x}_n$ (features) assumed given/fixed. Only modeling the response $y_n$
- **Log-likelihood** (ignoring constants w.r.t. $\boldsymbol{w}$)

$$\boxed{\log p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) \propto -\frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2}$$

- Note that **negative** log likelihood (NLL) is similar to squared loss function

# MLE for Probabilistic Linear Regression

- Assuming Gaussian distributed responses $y_n$, we have

$$p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}_n, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2}{2\sigma^2}\right\}$$

- Thus the likelihood (assuming i.i.d. responses) or *probability* of data:

$$p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) = \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) = \left(\frac{1}{2\pi\sigma^2}\right)^{\frac{N}{2}} \exp\left\{-\sum_{n=1}^{N} \frac{(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2}{2\sigma^2}\right\}$$

- Note: $\boldsymbol{x}_n$ (features) assumed given/fixed. Only modeling the response $y_n$
- **Log-likelihood** (ignoring constants w.r.t. $\boldsymbol{w}$)

$$\log p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) \propto -\frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2$$

- Note that **negative** log likelihood (NLL) is similar to squared loss function
- MLE will give the same solution as in the (unregularized) least squares

# MAP Estimation for Probabilistic Linear Regression

- We want to regularize our model, so we will use a prior distribution on the weight vector $\boldsymbol{w}$. We will use a **multivariate Gaussian prior** with zero mean

$$p(\boldsymbol{w}) = \mathcal{N}(0, \rho^2 \mathbf{I}_D) \propto \exp\left\{-\frac{\boldsymbol{w}^\top \boldsymbol{w}}{2\rho^2}\right\}$$

# MAP Estimation for Probabilistic Linear Regression

- We want to regularize our model, so we will use a prior distribution on the weight vector $\boldsymbol{w}$. We will use a **multivariate Gaussian prior** with zero mean

$$p(\boldsymbol{w}) = \mathcal{N}(0, \rho^2 \mathbf{I}_D) \propto \exp\left\{-\frac{\boldsymbol{w}^\top \boldsymbol{w}}{2\rho^2}\right\}$$

- The **log-likelihood**, as we have already seen, is given by

$$\log p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) \propto -\frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2$$

# MAP Estimation for Probabilistic Linear Regression

- We want to regularize our model, so we will use a prior distribution on the weight vector $\boldsymbol{w}$. We will use a **multivariate Gaussian prior** with zero mean

$$p(\boldsymbol{w}) = \mathcal{N}(0, \rho^2 \mathbf{I}_D) \propto \exp\left\{-\frac{\boldsymbol{w}^\top \boldsymbol{w}}{2\rho^2}\right\}$$

- The **log-likelihood**, as we have already seen, is given by

$$\log p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) \propto -\frac{1}{2\sigma^2}\sum_{n=1}^{N}(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2$$

- The MAP objective (log-posterior) will be the log-likelihood + $\log p(\boldsymbol{w})$

$$-\frac{1}{2\sigma^2}\sum_{n=1}^{N}(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 - \frac{\boldsymbol{w}^\top \boldsymbol{w}}{2\rho^2}$$

# MAP Estimation for Probabilistic Linear Regression

- We want to regularize our model, so we will use a prior distribution on the weight vector $\boldsymbol{w}$. We will use a **multivariate Gaussian prior** with zero mean

$$p(\boldsymbol{w}) = \mathcal{N}(0, \rho^2 \mathbf{I}_D) \propto \exp\left\{-\frac{\boldsymbol{w}^\top \boldsymbol{w}}{2\rho^2}\right\}$$

- The **log-likelihood**, as we have already seen, is given by

$$\log p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) \propto -\frac{1}{2\sigma^2}\sum_{n=1}^{N}(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2$$

- The MAP objective (log-posterior) will be the log-likelihood + $\log p(\boldsymbol{w})$

$$-\frac{1}{2\sigma^2}\sum_{n=1}^{N}(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 - \frac{\boldsymbol{w}^\top \boldsymbol{w}}{2\rho^2}$$

- Maximizing this is equivalent to minimizing the following w.r.t. $\boldsymbol{w}$

$$\hat{\mathbf{w}}_{MAP} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N}(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 + \frac{\sigma^2}{\rho^2}\boldsymbol{w}^\top \boldsymbol{w}$$

# MAP Estimation for Probabilistic Linear Regression

- We want to regularize our model, so we will use a prior distribution on the weight vector $\boldsymbol{w}$. We will use a **multivariate Gaussian prior** with zero mean

$$p(\boldsymbol{w}) = \mathcal{N}(0, \rho^2 \mathbf{I}_D) \propto \exp\left\{-\frac{\boldsymbol{w}^\top \boldsymbol{w}}{2\rho^2}\right\}$$

- The **log-likelihood**, as we have already seen, is given by

$$\log p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) \propto -\frac{1}{2\sigma^2}\sum_{n=1}^{N}(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2$$

- The MAP objective (log-posterior) will be the log-likelihood + $\log p(\boldsymbol{w})$

$$-\frac{1}{2\sigma^2}\sum_{n=1}^{N}(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 - \frac{\boldsymbol{w}^\top \boldsymbol{w}}{2\rho^2}$$

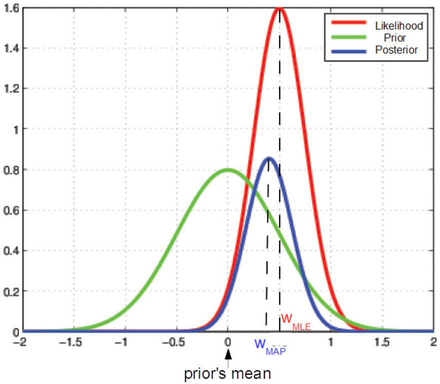
- Maximizing this is equivalent to minimizing the following w.r.t. $\boldsymbol{w}$

$$\hat{\mathbf{w}}_{MAP} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N}(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 + \frac{\sigma^2}{\rho^2}\boldsymbol{w}^\top \boldsymbol{w}$$

- Assuming $\lambda = \frac{\sigma^2}{\rho^2}$ (regularization hyperparam), it's equivalent to regularized (i.e., ridge) regression

# MLE vs MAP Estimation: An Illustration

$\boldsymbol{w}_{MAP}$ is a compromise between prior's mean and $\boldsymbol{w}_{MLE}$




In this case, doing MAP shrinks the estimate of $\boldsymbol{w}$ towards the prior's mean

# MLE vs MAP for Linear Regression: Summary

- MLE solution:

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

- MAP solution:

$$\hat{\mathbf{w}}_{MAP} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N} (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 + \frac{\sigma^2}{\rho^2} \boldsymbol{w}^\top \boldsymbol{w}$$

# MLE vs MAP for Linear Regression: Summary

- MLE solution:

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

- MAP solution:

$$\boxed{\hat{\mathbf{w}}_{MAP} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N} (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 + \frac{\sigma^2}{\rho^2} \boldsymbol{w}^\top \boldsymbol{w}}$$

- **Some Take-home messages:**

# MLE vs MAP for Linear Regression: Summary

- MLE solution:

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

- MAP solution:

$$\hat{\mathbf{w}}_{MAP} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N} (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 + \frac{\sigma^2}{\rho^2} \boldsymbol{w}^\top \boldsymbol{w}$$

- **Some Take-home messages:**
  - MLE estimation of a parameter leads to unregularized solutions

# MLE vs MAP for Linear Regression: Summary

- MLE solution:

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

- MAP solution:

$$\hat{\mathbf{w}}_{MAP} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N} (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 + \frac{\sigma^2}{\rho^2} \boldsymbol{w}^\top \boldsymbol{w}$$

- **Some Take-home messages:**
  - MLE estimation of a parameter leads to unregularized solutions
  - MAP estimation of a parameter leads to regularized solutions

# MLE vs MAP for Linear Regression: Summary

- MLE solution:

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

- MAP solution:

$$\hat{\mathbf{w}}_{MAP} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N} (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 + \frac{\sigma^2}{\rho^2} \boldsymbol{w}^\top \boldsymbol{w}$$

- **Some Take-home messages:**
  - MLE estimation of a parameter leads to unregularized solutions
  - MAP estimation of a parameter leads to regularized solutions
  - A Gaussian likelihood model corresponds to using squared loss

# MLE vs MAP for Linear Regression: Summary

- MLE solution:

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

- MAP solution:

$$\hat{\mathbf{w}}_{MAP} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N} (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 + \frac{\sigma^2}{\rho^2} \boldsymbol{w}^\top \boldsymbol{w}$$

- **Some Take-home messages:**
  - MLE estimation of a parameter leads to unregularized solutions
  - MAP estimation of a parameter leads to regularized solutions
  - A Gaussian likelihood model corresponds to using squared loss
  - A Gaussian prior on parameters acts as an $\ell_2$ regularizer

# MLE vs MAP for Linear Regression: Summary

- MLE solution:

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

- MAP solution:

$$\hat{\mathbf{w}}_{MAP} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N} (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 + \frac{\sigma^2}{\rho^2} \boldsymbol{w}^\top \boldsymbol{w}$$

- **Some Take-home messages:**
  - MLE estimation of a parameter leads to unregularized solutions
  - MAP estimation of a parameter leads to regularized solutions
  - A Gaussian likelihood model corresponds to using squared loss
  - A Gaussian prior on parameters acts as an $\ell_2$ regularizer
  - Other likelihoods/priors can be chosen. E.g., using a Laplace likelihood model can give more robustness to outliers than Gaussian likelihood

# MLE vs MAP for Linear Regression: Summary

- MLE solution:

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

- MAP solution:

$$\hat{\mathbf{w}}_{MAP} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N} (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 + \frac{\sigma^2}{\rho^2} \boldsymbol{w}^\top \boldsymbol{w}$$

- **Some Take-home messages:**
  - MLE estimation of a parameter leads to unregularized solutions
  - MAP estimation of a parameter leads to regularized solutions
  - A Gaussian likelihood model corresponds to using squared loss
  - A Gaussian prior on parameters acts as an $\ell_2$ regularizer
  - Other likelihoods/priors can be chosen. E.g., using a Laplace likelihood model can give more robustness to outliers than Gaussian likelihood
  - Note: Full Bayesian inference can be performed as well (not a focus of this course though)

# Probabilistic Modeling: Summary

- A flexible way to model data by specifying a proper probabilistic model

# Probabilistic Modeling: Summary

- A flexible way to model data by specifying a proper probabilistic model
- Can choose likelihoods and priors based on the nature/property of data

# Probabilistic Modeling: Summary

- A flexible way to model data by specifying a proper probabilistic model
- Can choose likelihoods and priors based on the nature/property of data
- Allows us to do **Bayesian learning**

# Probabilistic Modeling: Summary

- A flexible way to model data by specifying a proper probabilistic model
- Can choose likelihoods and priors based on the nature/property of data
- Allows us to do **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)

# Probabilistic Modeling: Summary

- A flexible way to model data by specifying a proper probabilistic model
- Can choose likelihoods and priors based on the nature/property of data
- Allows us to do **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)

# Probabilistic Modeling: Summary

- A flexible way to model data by specifying a proper probabilistic model
- Can choose likelihoods and priors based on the nature/property of data
- Allows us to do **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)
  - Allows getting an estimate of confidence in the model's prediction (useful for doing Active Learning)

# Probabilistic Modeling: Summary

- A flexible way to model data by specifying a proper probabilistic model
- Can choose likelihoods and priors based on the nature/property of data
- Allows us to do **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)
  - Allows getting an estimate of confidence in the model's prediction (useful for doing Active Learning)
  - Allows learning the size/complexity of the model from data (no tuning)

# Probabilistic Modeling: Summary

- A flexible way to model data by specifying a proper probabilistic model
- Can choose likelihoods and priors based on the nature/property of data
- Allows us to do **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)
  - Allows getting an estimate of confidence in the model's prediction (useful for doing Active Learning)
  - Allows learning the size/complexity of the model from data (no tuning)
  - Allows learning the hyperparameters from data (no tuning)

# Probabilistic Modeling: Summary

- A flexible way to model data by specifying a proper probabilistic model
- Can choose likelihoods and priors based on the nature/property of data
- Allows us to do **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)
  - Allows getting an estimate of confidence in the model's prediction (useful for doing Active Learning)
  - Allows learning the size/complexity of the model from data (no tuning)
  - Allows learning the hyperparameters from data (no tuning)
  - Allows learning in the presence of missing data

# Probabilistic Modeling: Summary

- A flexible way to model data by specifying a proper probabilistic model
- Can choose likelihoods and priors based on the nature/property of data
- Allows us to do **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)
  - Allows getting an estimate of confidence in the model's prediction (useful for doing Active Learning)
  - Allows learning the size/complexity of the model from data (no tuning)
  - Allows learning the hyperparameters from data (no tuning)
  - Allows learning in the presence of missing data
  - .. and many other benefits (a proper treatment deserves a separate course :) )

# Probabilistic Modeling: Summary

- A flexible way to model data by specifying a proper probabilistic model
- Can choose likelihoods and priors based on the nature/property of data
- Allows us to do **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)
  - Allows getting an estimate of confidence in the model's prediction (useful for doing Active Learning)
  - Allows learning the size/complexity of the model from data (no tuning)
  - Allows learning the hyperparameters from data (no tuning)
  - Allows learning in the presence of missing data
  - .. and many other benefits (a proper treatment deserves a separate course :) )
- MLE/MAP estimation is also related to the optimization view of ML

# Next Class:
# Probabilistic Models for Classification
# (Logistic Regression)