Bayesian Linear Regression (Hyperparameter Estimation, Sparse Priors), Bayesian Logistic Regression

Piyush Rai

Topics in Probabilistic Modeling and Inference (CS698X)

Jan 21, 2019
Recap: Bayesian Linear Regression

- Assume Gaussian likelihood:  
  
  \[ p(y|X, w, \beta) = \prod_{n=1}^{N} \mathcal{N}(y_n|w^\top x_n, \beta^{-1}) = \mathcal{N}(y|Xw, \beta^{-1}I_N) \]

- Assume zero-mean spherical Gaussian prior:  
  
  \[ p(w|\lambda) = \prod_{d=1}^{D} \mathcal{N}(w_d|0, \lambda^{-1}) = \mathcal{N}(w|0, \lambda^{-1}I_D) \]
Recap: Bayesian Linear Regression

- Assume Gaussian likelihood: \( p(y|X, w, \beta) = \prod_{n=1}^{N} \mathcal{N}(y_n|w^\top x_n, \beta^{-1}) = \mathcal{N}(y|Xw, \beta^{-1}I_N) \)

- Assume zero-mean spherical Gaussian prior: \( p(w|\lambda) = \prod_{d=1}^{D} \mathcal{N}(w_d|0, \lambda^{-1}) = \mathcal{N}(w|0, \lambda^{-1}I_D) \)

- Assuming hyperparameters as fixed, the posterior is Gaussian

\[
p(w|y, X, \beta, \lambda) = \mathcal{N}(\mu_N, \Sigma_N)
\]

Gives both predictive mean and predictive variance (imp: pred-var is different for each input)
Recap: Bayesian Linear Regression

- Assume Gaussian likelihood: \( p(y|X, w, \beta) = \prod_{n=1}^{N} \mathcal{N}(y_n|w^\top x_n, \beta^{-1}) = \mathcal{N}(y|Xw, \beta^{-1}I_N) \)
- Assume zero-mean spherical Gaussian prior: \( p(w|\lambda) = \prod_{d=1}^{D} \mathcal{N}(w_d|0, \lambda^{-1}) = \mathcal{N}(w|0, \lambda^{-1}I_D) \)
- Assuming hyperparameters as fixed, the posterior is Gaussian
  \[
  p(w|y, X, \beta, \lambda) = \mathcal{N}(\mu_N, \Sigma_N)
  \]
  \[
  \Sigma_N = (\beta \sum_{n=1}^{N} x_n x_n^\top + \lambda I_D)^{-1} = (\beta X^\top X + \lambda I_D)^{-1} \quad \text{(posterior's covariance matrix)}
  \]
Recap: Bayesian Linear Regression

- Assume Gaussian likelihood: \( p(y|X, w, \beta) = \prod_{n=1}^{N} \mathcal{N}(y_n|w^\top x_n, \beta^{-1}) = \mathcal{N}(y|Xw, \beta^{-1}I_N) \)

- Assume zero-mean spherical Gaussian prior: \( p(w|\lambda) = \prod_{d=1}^{D} \mathcal{N}(w_d|0, \lambda^{-1}) = \mathcal{N}(w|0, \lambda^{-1}I_D) \)

- Assuming hyperparameters as fixed, the posterior is Gaussian

\[
p(w|y, X, \beta, \lambda) = \mathcal{N}(\mu_N, \Sigma_N)
\]

\[
\Sigma_N = (\beta \sum_{n=1}^{N} x_n x_n^\top + \lambda I_D)^{-1} = (\beta X^\top X + \lambda I_D)^{-1} \quad \text{(posterior's covariance matrix)}
\]

\[
\mu_N = \Sigma_N \left[ \beta \sum_{n=1}^{N} y_n x_n \right] = \Sigma_N \left[ \beta X^\top y \right] = (X^\top X + \frac{\lambda}{\beta} I_D)^{-1} X^\top y \quad \text{(posterior's mean)}
\]

The posterior predictive distribution is also Gaussian

\[
p(y^*|x^*, X, y, \beta, \lambda) = \int p(y^*|w, x^*, \beta) p(w|y, X, \beta, \lambda) \, dw = \mathcal{N}(\mu_{N^*}, \Sigma_{N^*})
\]

Gives both predictive mean and predictive variance (imp: pred-var is different for each input)
Recap: Bayesian Linear Regression

- Assume Gaussian likelihood: \( p(y|X, w, \beta) = \prod_{n=1}^{N} \mathcal{N}(y_n|w^\top x_n, \beta^{-1}) = \mathcal{N}(y|Xw, \beta^{-1}I_N) \)

- Assume zero-mean spherical Gaussian prior: \( p(w|\lambda) = \prod_{d=1}^{D} \mathcal{N}(w_d|0, \lambda^{-1}) = \mathcal{N}(w|0, \lambda^{-1}I_D) \)

- Assuming hyperparameters as fixed, the posterior is Gaussian

\[
p(w|y, X, \beta, \lambda) = \mathcal{N}(\mu_N, \Sigma_N)
\]

\[
\Sigma_N = (\beta \sum_{n=1}^{N} x_n x_n^\top + \lambda I_D)^{-1} = (\beta X^\top X + \lambda I_D)^{-1} \quad \text{(posterior’s covariance matrix)}
\]

\[
\mu_N = \Sigma_N \left( \beta \sum_{n=1}^{N} y_n x_n \right) = \Sigma_N \left( \beta X^\top y \right) = (X^\top X + \frac{\lambda}{\beta} I_D)^{-1} X^\top y \quad \text{(posterior’s mean)}
\]

- The posterior predictive distribution is also Gaussian

\[
p(y_*|x_*, X, y, \beta, \lambda) = \int p(y_*|w, x_*, \beta)p(w|y, X, \beta, \lambda)dw = \mathcal{N}(\mu_N^\top x_*, \beta^{-1} + x_*^\top \Sigma_N x_*)
\]
Recap: Bayesian Linear Regression

- Assume Gaussian likelihood: 
  \[ p(y|X, w, \beta) = \prod_{n=1}^{N} \mathcal{N}(y_n|w^\top x_n, \beta^{-1}) = \mathcal{N}(y|Xw, \beta^{-1}I_N) \]

- Assume zero-mean spherical Gaussian prior: 
  \[ p(w|\lambda) = \prod_{d=1}^{D} \mathcal{N}(w_d|0, \lambda^{-1}) = \mathcal{N}(w|0, \lambda^{-1}I_D) \]

- Assuming hyperparameters as fixed, the posterior is Gaussian 
  \[
  p(w|y, X, \beta, \lambda) = \mathcal{N}(\mu_N, \Sigma_N) \\
  \Sigma_N = (\beta \sum_{n=1}^{N} x_n x_n^\top + \lambda I_D)^{-1} = (\beta X^\top X + \lambda I_D)^{-1} \quad \text{(posterior's covariance matrix)}
  \\
  \mu_N = \Sigma_N \left( \beta \sum_{n=1}^{N} y_n x_n \right) = \Sigma_N \left[ \beta X^\top y \right] = (X^\top X + \frac{\lambda}{\beta} I_D)^{-1} X^\top y \quad \text{(posterior's mean)}
  
  \]

- The posterior predictive distribution is also Gaussian 
  \[
  p(y_*|x_*, X, y, \beta, \lambda) = \int p(y_*|w, x_*, \beta)p(w|y, X, \beta, \lambda)dw = \mathcal{N}(\mu_N^\top x_*, \beta^{-1} + x_*^\top \Sigma_N x_*)
  
  \]

- Gives both predictive mean and predictive variance (imp: pred-var is different for each input)
A Visualization of Uncertainty in Bayesian Linear Regression

- Posterior $p(w|X, y)$ and lines ($w_0$ intercept, $w_1$ slope) corresponding to some random $w$’s
A Visualization of Uncertainty in Bayesian Linear Regression

- Posterior $p(w|X, y)$ and lines ($w_0$ intercept, $w_1$ slope) corresponding to some random $w$'s

- A visualization of the posterior predictive of a Bayesian linear regression model
A Visualization of Uncertainty (Contd)

- We can similarly visualize a Bayesian nonlinear regression model.

- Figures below: Green curve is the true function and blue circles are observations \((x_n, y_n)\).

- Posterior of the nonlinear regression model: Some curves drawn from the posterior.

![Diagram of posterior predictive distribution with red curve as predictive mean and shaded region for predictive uncertainty.](image)
A Visualization of Uncertainty (Contd)

- We can similarly visualize a Bayesian nonlinear regression model.

- Figures below: Green curve is the true function and blue circles are observations \((x_n, y_n)\).

- Posterior of the nonlinear regression model: Some curves drawn from the posterior.

- Posterior predictive: Red curve is predictive mean, shaded region denotes predictive uncertainty.
Estimating Hyperparameters for Bayesian Linear Regression
Learning Hyperparameters in Probabilistic Models

- Can treat hyperparams as just a bunch of additional unknowns
- Can be learned using a suitable inference algorithm (point estimation or fully Bayesian)
Learning Hyperparameters in Probabilistic Models

- Can treat hyperparams as just a bunch of additional unknowns
- Can be learned using a suitable inference algorithm (point estimation or fully Bayesian)
- Example: For the linear regression model, the full set of parameters would be \((w, \lambda, \beta)\)

![Diagram showing the relationship between hyperparameters and the model](image.png)

Infering the above is usually intractable (rare to have conjugacy). Requires approximations.

What priors (or "hyperpriors") to choose for \(\beta\) and \(\lambda\)?
What about the hyperparameters of those priors?
Learning Hyperparameters in Probabilistic Models

- Can treat hyperparams as just a bunch of additional unknowns
- Can be learned using a suitable inference algorithm (point estimation or fully Bayesian)
- Example: For the linear regression model, the full set of parameters would be \((w, \lambda, \beta)\)

Can assume priors on all these parameters and infer their “joint” posterior distribution

\[
p(w, \beta, \lambda | X, y) = \frac{p(y | X, w, \beta, \lambda)p(w, \beta, \lambda)}{p(y | X)} = \frac{p(y | X, w, \beta, \lambda)p(w | \lambda)p(\beta)p(\lambda)}{\int p(y | X, w, \beta)p(w | \lambda)p(\beta)p(\lambda) \, dw \, d\lambda \, d\beta}
\]
Learning Hyperparameters in Probabilistic Models

- Can treat hyperparams as just a bunch of additional unknowns
- Can be learned using a suitable inference algorithm (point estimation or fully Bayesian)
- Example: For the linear regression model, the full set of parameters would be \((w, \lambda, \beta)\)

Can assume priors on all these parameters and infer their “joint” posterior distribution

\[
p(w, \beta, \lambda | X, y) = \frac{p(y|X, w, \beta, \lambda)p(w, \lambda, \beta)}{p(y|X)} = \frac{p(y|X, w, \beta, \lambda)p(w|\lambda)p(\beta)p(\lambda)}{\int p(y|X, w, \beta)p(w|\lambda)p(\beta)p(\lambda) \, dw \, d\lambda \, d\beta}
\]

- Infering the above is usually intractable (rare to have conjugacy).
Learning Hyperparameters in Probabilistic Models

- Can treat hyperparams as just a bunch of additional unknowns
- Can be learned using a suitable inference algorithm (point estimation or fully Bayesian)
- Example: For the linear regression model, the full set of parameters would be \((w, \lambda, \beta)\)

\[
p(w, \beta, \lambda | X, y) = \frac{p(y|X, w, \beta, \lambda)p(w, \lambda, \beta)}{p(y|X)} = \frac{p(y|X, w, \beta, \lambda)p(w|\lambda)p(\beta)p(\lambda)}{\int p(y|X, w, \beta)p(w|\lambda)p(\beta)p(\lambda) \, dw \, d\lambda \, d\beta}
\]

- Can assume priors on all these parameters and infer their “joint” posterior distribution
- Infering the above is usually intractable (rare to have conjugacy). Requires approximations.
Learning Hyperparameters in Probabilistic Models

- Can treat hyperparams as just a bunch of additional unknowns
- Can be learned using a suitable inference algorithm (point estimation or fully Bayesian)
- Example: For the linear regression model, the full set of parameters would be \((w, \lambda, \beta)\)

\[
p(w, \beta, \lambda|X, y) = \frac{p(y|X, w, \beta, \lambda)p(w, \lambda, \beta)}{p(y|X)} = \frac{p(y|X, w, \beta, \lambda)p(w|\lambda)p(\beta)p(\lambda)}{\int p(y|X, w, \beta)p(w|\lambda)p(\beta)p(\lambda) \, dw \, d\lambda \, d\beta}
\]

- Can assume priors on all these parameters and infer their “joint” posterior distribution

- Infering the above is usually intractable (rare to have conjugacy). Requires approximations. Also,
Learning Hyperparameters in Probabilistic Models

- Can treat hyperparams as just a bunch of additional unknowns
- Can be learned using a suitable inference algorithm (point estimation or fully Bayesian)
- Example: For the linear regression model, the full set of parameters would be \((w, \lambda, \beta)\)

Can assume priors on all these parameters and infer their “joint” posterior distribution

\[
p(w, \beta, \lambda | X, y) = \frac{p(y|X, w, \beta, \lambda)p(w, \beta, \lambda)}{p(y|X)} = \frac{p(y|X, w, \beta, \lambda)p(w|\lambda)p(\beta)p(\lambda)}{\int p(y|X, w, \beta)p(w|\lambda)p(\beta)p(\lambda) \, dw \, d\lambda \, d\beta}
\]

Infering the above is usually intractable (rare to have conjugacy). Requires approximations. Also,

- What priors (or “hyperpriors”) to choose for \(\beta\) and \(\lambda\)?
Learning Hyperparameters in Probabilistic Models

- Can treat hyperparams as just a bunch of additional unknowns
- Can be learned using a suitable inference algorithm (point estimation or fully Bayesian)
- Example: For the linear regression model, the full set of parameters would be \((w, \lambda, \beta)\)

\[
\begin{align*}
p(w, \lambda, \beta | X, y) &= \frac{p(y|X, w, \beta, \lambda)p(w, \lambda, \beta)}{p(y|X)} = \frac{p(y|X, w, \beta, \lambda)p(w|\lambda)p(\beta)p(\lambda)}{\int p(y|X, w, \beta)p(w|\lambda)p(\beta)p(\lambda) \, dw \, d\lambda \, d\beta} \\
\end{align*}
\]

- Can assume priors on all these parameters and infer their “joint” posterior distribution

- Infering the above is usually intractable (rare to have conjugacy). Requires approximations. Also,
  - What priors (or “hyperpriors”) to choose for \(\beta\) and \(\lambda\)?
  - What about the hyperparameters of those priors?
One popular way to estimate hyperparameters is by maximizing the marginal likelihood:

\[ p(y|X, \beta, \lambda) = \int p(y|X, w, \beta) p(w|\lambda) \, dw \]

The "optimal" hyperparameters in this case can be then found by:

\[ \hat{\beta}, \hat{\lambda} = \arg \max_{\beta, \lambda} \log p(y|X, \beta, \lambda) \]

This is called MLE-II or (log) evidence maximization. Akin to doing MLE to estimate the hyperparameters where the "main" parameter (in this case \( w \)) has been integrated out from the model's likelihood function.

Note: If the likelihood and prior are conjugate then marginal likelihood is available in closed form.
Learning Hyperparameters via Point Estimation

- One popular way to estimate hyperparameters is by maximizing the _marginal likelihood_.

- For our linear regression model, this quantity (a function of the hyperparams) will be

\[
p(y|\mathbf{X}, \beta, \lambda) = \int p(y|\mathbf{X}, \mathbf{w}, \beta)p(\mathbf{w}|\lambda)d\mathbf{w}
\]

The "optimal" hyperparameters in this case can then be found by

\[
\hat{\beta}, \hat{\lambda} = \arg \max_{\beta, \lambda} \log p(y|\mathbf{X}, \beta, \lambda)
\]

This is called MLE-II or (log) evidence maximization.

Akin to doing MLE to estimate the hyperparameters where the "main" parameter (in this case \(\mathbf{w}\)) has been integrated out from the model's likelihood function.

Note: If the likelihood and prior are conjugate then marginal likelihood is available in closed form.
One popular way to estimate hyperparameters is by maximizing the marginal likelihood.

For our linear regression model, this quantity (a function of the hyperparams) will be:

$$ p(y|X, \beta, \lambda) = \int p(y|X, w, \beta)p(w|\lambda)dw $$

The “optimal” hyperparameters in this case can be then found by:

$$ \hat{\beta}, \hat{\lambda} = \arg \max_{\beta, \lambda} \log p(y|X, \beta, \lambda) $$
Learning Hyperparameters via Point Estimation

One popular way to estimate hyperparameters is by maximizing the marginal likelihood.

For our linear regression model, this quantity (a function of the hyperparams) will be

\[ p(y|X, \beta, \lambda) = \int p(y|X, w, \beta)p(w|\lambda)dw \]

The “optimal” hyperparameters in this case can be then found by

\[ \hat{\beta}, \hat{\lambda} = \arg \max_{\beta, \lambda} \log p(y|X, \beta, \lambda) \]

This is called MLE-II or (log) evidence maximization.
Learning Hyperparameters via Point Estimation

One popular way to estimate hyperparameters is by maximizing the marginal likelihood.

For our linear regression model, this quantity (a function of the hyperparams) will be

\[
p(y|X, \beta, \lambda) = \int p(y|X, w, \beta)p(w|\lambda)dw
\]

The “optimal” hyperparameters in this case can be then found by

\[
\hat{\beta}, \hat{\lambda} = \arg\max_{\beta, \lambda} \log p(y|X, \beta, \lambda)
\]

This is called MLE-II or (log) evidence maximization.

Akin to doing MLE to estimate the hyperparameters where the “main” parameter (in this case \(w\)) has been integrated out from the model’s likelihood function.
One popular way to estimate hyperparameters is by maximizing the marginal likelihood.

For our linear regression model, this quantity (a function of the hyperparams) will be

\[ p(y|X, \beta, \lambda) = \int p(y|X, w, \beta)p(w|\lambda)dw \]

The “optimal” hyperparameters in this case can be then found by

\[ \hat{\beta}, \hat{\lambda} = \arg\max_{\beta, \lambda} \log p(y|X, \beta, \lambda) \]

This is called MLE-II or (log) evidence maximization.

Akin to doing MLE to estimate the hyperparameters where the “main” parameter (in this case \( w \)) has been integrated out from the model’s likelihood function.

Note: If the likelihood and prior are conjugate then marginal likelihood is available in closed form.
What is MLE-II Doing?

- For linear regression case, would ideally like the posterior over all unknowns, i.e., \( p(w, \lambda, \beta|X, y) \)

\[
p(w, \beta, \lambda|X, y) = p(w|X, y, \beta, \lambda)p(\beta, \lambda|X, y) \quad \text{(from product rule)}
\]

Note that \( p(w|X, y, \beta, \lambda) \) is easy if \( \lambda, \beta \) are known

However \( p(\beta, \lambda|X, y) = p(y|X, \beta, \alpha)p(\beta)p(\lambda) \) is hard (lack of conjugacy, intractable denominator)

Let's approximate it by a point function \( \delta \) at the mode of \( p(\beta, \lambda|X, y) \)

\[
p(\beta, \lambda|X, y) \approx \delta(\hat{\beta}, \hat{\lambda})
\]

where \( \hat{\beta}, \hat{\lambda} = \text{arg max}_{\beta, \lambda} p(\beta, \lambda|X, y) = \text{arg max}_{\beta, \lambda} p(y|X, \beta, \lambda)p(\beta)p(\lambda) \)

Moreover, if \( p(\beta), p(\lambda) \) are uniform/uninformative priors then

\[
\hat{\beta}, \hat{\lambda} = \text{arg max}_{\beta, \lambda} p(y|X, \beta, \lambda)p(\beta)p(\lambda)
\]

Thus MLE-II is approximating the posterior of hyperparams by their point estimate assuming uniform priors (therefore we don't need to worry about a prior over the hyperparams).
What is MLE-II Doing?

- For linear regression case, would ideally like the posterior over all unknowns, i.e., $p(w, \lambda, \beta | X, y)$

$$p(w, \beta, \lambda | X, y) = p(w | X, y, \beta, \lambda)p(\beta, \lambda | X, y) \quad \text{ (from product rule)}$$

- Note that $p(w | X, y, \beta, \lambda)$ is easy if $\lambda, \beta$ are known
What is MLE-II Doing?

For linear regression case, would ideally like the posterior over all unknowns, i.e., $p(w, \lambda, \beta|X, y)$

$$p(w, \beta, \lambda|X, y) = p(w|X, y, \beta, \lambda)p(\beta, \lambda|X, y) \quad \text{from product rule}$$

Note that $p(w|X, y, \beta, \lambda)$ is easy if $\lambda, \beta$ are known

However $p(\beta, \lambda|X, y) = \frac{p(y|X, \beta, \alpha)p(\beta)p(\lambda)}{p(y|X)}$ is hard
What is MLE-II Doing?

- For linear regression case, would ideally like the posterior over all unknowns, i.e., 
  \[ p(w, \lambda, \beta | X, y) \]

  \[
p(w, \beta, \lambda | X, y) = p(w | X, y, \beta, \lambda) p(\beta, \lambda | X, y) \quad \text{(from product rule)}
  \]

- Note that \( p(w | X, y, \beta, \lambda) \) is easy if \( \lambda, \beta \) are known

- However \( p(\beta, \lambda | X, y) = \frac{p(y | X, \beta, \alpha)p(\beta)p(\lambda)}{p(y | X)} \) is hard (lack of conjugacy, intractable denominator)
What is MLE-II Doing?

- For linear regression case, would ideally like the posterior over all unknowns, i.e., $p(w, \lambda, \beta | X, y)$

$$p(w, \beta, \lambda | X, y) = p(w | X, y, \beta, \lambda) p(\beta, \lambda | X, y) \quad \text{(from product rule)}$$

- Note that $p(w | X, y, \beta, \lambda)$ is easy if $\lambda, \beta$ are known

- However $p(\beta, \lambda | X, y) = \frac{p(y | x, \beta, \alpha) p(\beta) p(\lambda)}{p(y | X)}$ is hard (lack of conjugacy, intractable denominator)

- Let's approximate it by a point function $\delta$ at the mode of $p(\beta, \lambda | X, y)$

$$p(\beta, \lambda | X, y) \approx \delta(\hat{\beta}, \hat{\lambda})$$
What is MLE-II Doing?

- For linear regression case, would ideally like the posterior over all unknowns, i.e., \( p(w, \lambda, \beta | X, y) \)

\[
p(w, \beta, \lambda | X, y) = p(w | X, y, \beta, \lambda)p(\beta, \lambda | X, y) \quad \text{(from product rule)}
\]

- Note that \( p(w | X, y, \beta, \lambda) \) is easy if \( \lambda, \beta \) are known

- However \( p(\beta, \lambda | X, y) = \frac{p(y | X, \beta, \alpha)p(\beta)p(\lambda)}{p(y | X)} \) is hard (lack of conjugacy, intractable denominator)

- Let’s approximate it by a point function \( \delta \) at the mode of \( p(\beta, \lambda | X, y) \)

\[
p(\beta, \lambda | X, y) \approx \delta(\hat{\beta}, \hat{\lambda}) \quad \text{where} \quad \hat{\beta}, \hat{\lambda} = \arg \max_{\beta, \lambda} p(\beta, \lambda | X, y) = \arg \max_{\beta, \lambda} p(y | X, \beta, \lambda)p(\lambda)p(\beta)
\]
What is MLE-II Doing?

- For linear regression case, would ideally like the posterior over all unknowns, i.e., \( p(w, \lambda, \beta | X, y) \)

\[
p(w, \beta, \lambda | X, y) = p(w | X, y, \beta, \lambda) p(\beta, \lambda | X, y) \quad \text{(from product rule)}
\]

- Note that \( p(w | X, y, \beta, \lambda) \) is easy if \( \lambda, \beta \) are known

- However \( p(\beta, \lambda | X, y) = \frac{p(y | X, \beta, \lambda) p(\beta) p(\lambda)}{p(y | X)} \) is hard (lack of conjugacy, intractable denominator)

- Let’s approximate it by a point function \( \delta \) at the mode of \( p(\beta, \lambda | X, y) \)

\[
p(\beta, \lambda | X, y) \approx \delta(\hat{\beta}, \hat{\lambda}) \quad \text{where} \quad \hat{\beta}, \hat{\lambda} = \arg \max_{\beta, \lambda} p(\beta, \lambda | X, y) = \arg \max_{\beta, \lambda} p(y | X, \beta, \lambda) p(\lambda) p(\beta)
\]

- Moreover, if \( p(\beta), p(\lambda) \) are uniform/uninformative priors then

\[
\hat{\beta}, \hat{\lambda} = \arg \max_{\beta, \lambda} p(y | X, \beta, \lambda)
\]
What is MLE-II Doing?

- For linear regression case, would ideally like the posterior over all unknowns, i.e., \( p(w, \lambda, \beta | X, y) \)

\[
p(w, \beta, \lambda | X, y) = p(w | X, y, \beta, \lambda)p(\beta, \lambda | X, y) \quad \text{(from product rule)}
\]

- Note that \( p(w | X, y, \beta, \lambda) \) is easy if \( \lambda, \beta \) are known

- However \( p(\beta, \lambda | X, y) = \frac{p(y | X, \beta, \lambda)p(\beta)p(\lambda)}{p(y | X)} \) is hard (lack of conjugacy, intractable denominator)

- Let’s approximate it by a point function \( \delta \) at the mode of \( p(\beta, \lambda | X, y) \)

\[
p(\beta, \lambda | X, y) \approx \delta(\hat{\beta}, \hat{\lambda}) \quad \text{where} \quad \hat{\beta}, \hat{\lambda} = \arg \max_{\beta, \lambda} p(\beta, \lambda | X, y) = \arg \max_{\beta, \lambda} p(y | X, \beta, \lambda)p(\lambda)p(\beta)
\]

- Moreover, if \( p(\beta), p(\lambda) \) are uniform/uninformative priors then

\[
\hat{\beta}, \hat{\lambda} = \arg \max_{\beta, \lambda} p(y | X, \beta, \lambda)
\]

- Thus MLE-II is approximating the posterior of hyperparams by their point estimate assuming uniform priors (therefore we don’t need to worry about a prior over the hyperparams)
For the linear regression case, the marginal likelihood is defined as

\[ p(y|X, \beta, \lambda) = \int p(y|X, w, \beta)p(w|\lambda) dw \]

MLE-II maximizes \( \log p(y|X, \beta, \lambda) \) w.r.t. \( \beta \) and \( \lambda \) to estimate these hyperparams. This objective doesn't have a closed form solution, solved using iterative/alternating optimization. PRML Chapter 3 contains the iterative update equations.

Note: Can also do "MAP-II" using a suitable prior on these hyperparams (e.g., gamma). Note: Can also use different \( \lambda \) for each \( w \).
MLE-II for Linear Regression

- For the linear regression case, the marginal likelihood is defined as

\[ p(y|X, \beta, \lambda) = \int p(y|X, w, \beta)p(w|\lambda)dw \]

- Since \( p(y|X, w, \beta) = \mathcal{N}(y|Xw, \beta^{-1}I_N) \) and \( p(w|\lambda) = \mathcal{N}(w|0, \lambda^{-1}I_D) \), the marginal likelihood

\[ p(y|X, \beta, \lambda) = \mathcal{N}(y|0, \beta^{-1}I + \lambda^{-1}XX^\top) \]
MLE-II for Linear Regression

For the linear regression case, the marginal likelihood is defined as

$$p(y|X, \beta, \lambda) = \int p(y|X, w, \beta)p(w|\lambda)dw$$

Since $p(y|X, w, \beta) = \mathcal{N}(y|Xw, \beta^{-1}I_N)$ and $p(w|\lambda) = \mathcal{N}(w|0, \lambda^{-1}I_D)$, the marginal likelihood

$$p(y|X, \beta, \lambda) = \mathcal{N}(y|0, \beta^{-1}I + \lambda^{-1}XX^\top)$$

$$= \frac{1}{(2\pi)^{N/2} |\beta^{-1}I + \lambda^{-1}XX^\top|^{-1/2}} \exp\left(-\frac{1}{2}y^\top (\beta^{-1}I + \lambda^{-1}XX^\top)^{-1}y\right)$$

MLE-II maximizes log $p(y|X, \beta, \lambda)$ w.r.t. $\beta$ and $\lambda$ to estimate these hyperparams.

This objective doesn't have a closed form solution. Solved using iterative/alternating optimization.

PRML Chapter 3 contains the iterative update equations.

Note: Can also do "MAP-II" using a suitable prior on these hyperparams (e.g., gamma).

Note: Can also use different $\lambda_d$ for each $w_d$. 

Prob. Mod. & Inference - CS698X (Piyush Rai, IITK) Bayesian Linear Regression (Hyperparameter Estimation, Sparse Priors), Bayesian Logistic Regression
MLE-II for Linear Regression

- For the linear regression case, the marginal likelihood is defined as

\[ p(y|X, \beta, \lambda) = \int p(y|X, w, \beta)p(w|\lambda)dw \]

- Since \( p(y|X, w, \beta) = \mathcal{N}(y|Xw, \beta^{-1}I_N) \) and \( p(w|\lambda) = \mathcal{N}(w|0, \lambda^{-1}I_D) \), the marginal likelihood

\[
p(y|X, \beta, \lambda) = \mathcal{N}(y|0, \beta^{-1}I + \lambda^{-1}XX^\top) \\
= \frac{1}{(2\pi)^{N/2} |\beta^{-1}I + \lambda^{-1}XX^\top|^{-1/2}} \exp\left(-\frac{1}{2} y^\top(\beta^{-1}I + \lambda^{-1}XX^\top)^{-1}y\right)
\]

- MLE-II maximizes \( \log p(y|X, \beta, \lambda) \) w.r.t. \( \beta \) and \( \lambda \) to estimate these hyperparams
MLE-II for Linear Regression

- For the linear regression case, the marginal likelihood is defined as

\[ p(y|X, \beta, \lambda) = \int p(y|X, w, \beta)p(w|\lambda)dw \]

- Since \( p(y|X, w, \beta) = \mathcal{N}(y|Xw, \beta^{-1}I_N) \) and \( p(w|\lambda) = \mathcal{N}(w|0, \lambda^{-1}I_D) \), the marginal likelihood

\[
p(y|X, \beta, \lambda) = \mathcal{N}(y|0, \beta^{-1}I + \lambda^{-1}XX^\top)
\]

\[
= \frac{1}{(2\pi)^{N/2}|\beta^{-1}I + \lambda^{-1}XX^\top|^{-1/2}} \exp\left(-\frac{1}{2}y^\top(\beta^{-1}I + \lambda^{-1}XX^\top)^{-1}y\right)
\]

- MLE-II maximizes \( \log p(y|X, \beta, \lambda) \) w.r.t. \( \beta \) and \( \lambda \) to estimate these hyperparams

  - This objective doesn’t have a closed form solution
MLE-II for Linear Regression

For the linear regression case, the marginal likelihood is defined as

\[ p(y|X, \beta, \lambda) = \int p(y|X, w, \beta)p(w|\lambda)dw \]

Since \( p(y|X, w, \beta) = \mathcal{N}(y|Xw, \beta^{-1}I_N) \) and \( p(w|\lambda) = \mathcal{N}(w|0, \lambda^{-1}I_D) \), the marginal likelihood

\[
p(y|X, \beta, \lambda) = \mathcal{N}(y|0, \beta^{-1}I + \lambda^{-1}XX^\top) \]
\[= \frac{1}{(2\pi)^{N/2}|\beta^{-1}I + \lambda^{-1}XX^\top|^{-1/2}} \exp\left(-\frac{1}{2}y^\top(\beta^{-1}I + \lambda^{-1}XX^\top)^{-1}y\right)\]

MLE-II maximizes \( \log p(y|X, \beta, \lambda) \) w.r.t. \( \beta \) and \( \lambda \) to estimate these hyperparams

- This objective doesn't have a closed form solution
- Solved using iterative/alternating optimization
MLE-II for Linear Regression

- For the linear regression case, the marginal likelihood is defined as
  \[
  p(y|X, \beta, \lambda) = \int p(y|X, w, \beta) p(w|\lambda) dw
  \]

- Since \( p(y|X, w, \beta) = \mathcal{N}(y|Xw, \beta^{-1}I_N) \) and \( p(w|\lambda) = \mathcal{N}(w|0, \lambda^{-1}I_D) \), the marginal likelihood
  \[
  p(y|X, \beta, \lambda) = \mathcal{N}(y|0, \beta^{-1}I + \lambda^{-1}XX^T)
  \]
  \[
  = \frac{1}{(2\pi)^{N/2}|\beta^{-1}I + \lambda^{-1}XX^T|^{-1/2}} \exp\left(-\frac{1}{2}y^T(\beta^{-1}I + \lambda^{-1}XX^T)^{-1}y\right)
  \]

- MLE-II maximizes log \( p(y|X, \beta, \lambda) \) w.r.t. \( \beta \) and \( \lambda \) to estimate these hyperparams
  - This objective doesn’t have a closed form solution
  - Solved using iterative/alternating optimization
  - PRML Chapter 3 contains the iterative update equations
MLE-II for Linear Regression

- For the linear regression case, the marginal likelihood is defined as

\[ p(y|X, \beta, \lambda) = \int p(y|X, w, \beta)p(w|\lambda)dw \]

- Since \( p(y|X, w, \beta) = \mathcal{N}(y|Xw, \beta^{-1}I_N) \) and \( p(w|\lambda) = \mathcal{N}(w|0, \lambda^{-1}I_D) \), the marginal likelihood

\[
p(y|X, \beta, \lambda) = \mathcal{N}(y|0, \beta^{-1}I + \lambda^{-1}XX^\top) \\
= \frac{1}{(2\pi)^{N/2} |\beta^{-1}I + \lambda^{-1}XX^\top|^{-1/2}} \exp(-\frac{1}{2}y^\top (\beta^{-1}I + \lambda^{-1}XX^\top)^{-1}y)
\]

- MLE-II maximizes log \( p(y|X, \beta, \lambda) \) w.r.t. \( \beta \) and \( \lambda \) to estimate these hyperparams
  - This objective doesn’t have a closed form solution
  - Solved using iterative/alternating optimization
  - PRML Chapter 3 contains the iterative update equations

- Note: Can also do “MAP-II” using a suitable prior on these hyperparams (e.g., gamma)
MLE-II for Linear Regression

For the linear regression case, the marginal likelihood is defined as

$$p(y|X, \beta, \lambda) = \int p(y|X, w, \beta) p(w|\lambda) dw$$

Since $p(y|X, w, \beta) = \mathcal{N}(y|Xw, \beta^{-1}I_N)$ and $p(w|\lambda) = \mathcal{N}(w|0, \lambda^{-1}I_D)$, the marginal likelihood

$$p(y|X, \beta, \lambda) = \mathcal{N}(y|0, \beta^{-1}I + \lambda^{-1}XX^\top)$$

$$= \frac{1}{(2\pi)^{N/2} |\beta^{-1}I + \lambda^{-1}XX^\top|^{-1/2}} \exp\left(-\frac{1}{2} y^\top (\beta^{-1}I + \lambda^{-1}XX^\top)^{-1} y\right)$$

MLE-II maximizes $\log p(y|X, \beta, \lambda)$ w.r.t. $\beta$ and $\lambda$ to estimate these hyperparams

- This objective doesn’t have a closed form solution
- Solved using iterative/alternating optimization
- PRML Chapter 3 contains the iterative update equations

Note: Can also do “MAP-II” using a suitable prior on these hyperparams (e.g., gamma)

Note: Can also use different $\lambda_d$ for each $w_d$
Using MLE-II Estimates for Making Prediction

With the MLE-II approximation $p(\beta, \lambda | X, y) \approx \delta(\hat{\beta}, \hat{\lambda})$, the posterior over unknowns

$$p(w, \beta, \lambda | X, y) = p(w | X, y, \beta, \lambda)p(\beta, \lambda | X, y)$$
Using MLE-II Estimates for Making Prediction

With the MLE-II approximation $p(\beta, \lambda | X, y) \approx \delta(\hat{\beta}, \hat{\lambda})$, the posterior over unknowns

$$p(w, \beta, \lambda | X, y) = p(w | X, y, \beta, \lambda)p(\beta, \lambda | X, y) \approx p(w | X, y, \hat{\beta}, \hat{\lambda})$$
Using MLE-II Estimates for Making Prediction

- With the MLE-II approximation \( p(\beta, \lambda | \mathbf{X}, \mathbf{y}) \approx \delta(\hat{\beta}, \hat{\lambda}) \), the posterior over unknowns
  \[
p(w, \beta, \lambda | \mathbf{X}, \mathbf{y}) = p(w | \mathbf{X}, \mathbf{y}, \beta, \lambda) p(\beta, \lambda | \mathbf{X}, \mathbf{y}) \approx p(w | \mathbf{X}, \mathbf{y}, \hat{\beta}, \hat{\lambda})
  \]

- The posterior predictive distribution can also be approximated as
  \[
p(y_* | x_*, \mathbf{X}, \mathbf{y}) = \int p(y_* | x_*, w, \beta) p(w, \beta, \lambda | \mathbf{X}, \mathbf{y}) \, dw \, d\beta \, d\lambda
  \]
Using MLE-II Estimates for Making Prediction

- With the MLE-II approximation \( p(\beta, \lambda | X, y) \approx \delta(\hat{\beta}, \hat{\lambda}) \), the posterior over unknowns
  \[
p(w, \beta, \lambda | X, y) = p(w | X, y, \beta, \lambda)p(\beta, \lambda | X, y) \approx p(w | X, y, \hat{\beta}, \hat{\lambda})
  \]

- The posterior predictive distribution can also be approximated as
  \[
p(y_* | x_*, X, y) = \int p(y_* | x_*, w, \beta)p(w, \beta, \lambda | X, y) \, dw \, d\beta \, d\lambda \\
  = \int p(y_* | x_*, w, \beta)p(w | X, y, \beta, \lambda)p(\beta, \lambda | X, y) \, d\beta \, d\lambda \, dw
  \]
Using MLE-II Estimates for Making Prediction

- With the MLE-II approximation \( p(\beta, \lambda | X, y) \approx \delta(\hat{\beta}, \hat{\lambda}) \), the posterior over unknowns

\[
p(w, \beta, \lambda | X, y) = p(w | X, y, \beta, \lambda)p(\beta, \lambda | X, y) \approx p(w | X, y, \hat{\beta}, \hat{\lambda})
\]

- The posterior predictive distribution can also be approximated as

\[
p(y_* | x_*, X, y) = \int p(y_* | x_*, w, \beta)p(w, \beta, \lambda | X, y) \, dw \, d\beta \, d\lambda
\]

\[
= \int p(y_* | x_*, w, \beta)p(w | X, y, \beta, \lambda)p(\beta, \lambda | X, y) \, d\beta \, d\lambda \, dw
\]

\[
\approx \int p(y_* | x_*, w, \beta)p(w | X, y, \hat{\beta}, \hat{\lambda}) \, dw
\]
Using MLE-II Estimates for Making Prediction

With the MLE-II approximation $p(\beta, \lambda|X, y) \approx \delta(\hat{\beta}, \hat{\lambda})$, the posterior over unknowns

$$p(w, \beta, \lambda|X, y) = p(w|X, y, \beta, \lambda)p(\beta, \lambda|X, y) \approx p(w|X, y, \hat{\beta}, \hat{\lambda})$$

The posterior predictive distribution can also be approximated as

$$p(y_*|x_*, X, y) = \int p(y_*|x_*, w, \beta)p(w, \beta, \lambda|X, y) \, dw \, d\beta \, d\lambda$$

$$= \int p(y_*|x_*, w, \beta)p(w|X, y, \beta, \lambda)p(\beta, \lambda|X, y)\, d\beta \, d\lambda \, dw$$

$$\approx \int p(y_*|x_*, w, \beta)p(w|X, y, \hat{\beta}, \hat{\lambda}) \, dw$$

This is also the same as the usual posterior predictive distribution we have seen earlier, except we are treating the hyperparams $\hat{\beta}, \hat{\lambda}$ fixed at their MLE-II based estimates.
Modeling Sparse Weights
Many probabilistic models consist of weights that are given zero-mean Gaussian priors, e.g.,

\[ \mu(x) = \sum_{d=1}^{D} w_d x_d \]  
(mean of a prob. lin reg model)

\[ \mu(x) = \sum_{n=1}^{N} w_n k(x_n, x) \]  
(mean of a prob. kernel based nonlin reg model)
Modeling Sparse Weights

Many probabilistic models consist of weights that are given zero-mean Gaussian priors, e.g.,

\[
\mu(x) = \sum_{d=1}^{D} w_d x_d \quad \text{(mean of a prob. lin reg model)}
\]

\[
\mu(x) = \sum_{n=1}^{N} w_n k(x_n, x) \quad \text{(mean of a prob. kernel based nonlin reg model)}
\]

A zero-mean prior is of the form \( p(w_d) = \mathcal{N}(0, \lambda^{-1}) \) or \( p(w_d) = \mathcal{N}(0, \lambda_d^{-1}) \)
Modeling Sparse Weights

Many probabilistic models consist of weights that are given zero-mean Gaussian priors, e.g.,

\[ \mu(x) = \sum_{d=1}^{D} w_d x_d \] (mean of a prob. lin reg model)

\[ \mu(x) = \sum_{n=1}^{N} w_n k(x_n, x) \] (mean of a prob. kernel based nonlin reg model)

A zero-mean prior is of the form \( p(w_d) = \mathcal{N}(0, \lambda^{-1}) \) or \( p(w_d) = \mathcal{N}(0, \lambda_d^{-1}) \)

Precision \( \lambda \) or \( \lambda_d \) specifies our belief about how close to zero \( w_d \) is (like regularization hyperparam)
Modeling Sparse Weights

- Many probabilistic models consist of weights that are given zero-mean Gaussian priors, e.g.,

\[
\mu(x) = \sum_{d=1}^{D} w_d x_d \quad \text{(mean of a prob. lin reg model)}
\]

\[
\mu(x) = \sum_{n=1}^{N} w_n k(x_n, x) \quad \text{(mean of a prob. kernel based nonlin reg model)}
\]

- A zero-mean prior is of the form \( p(w_d) = \mathcal{N}(0, \lambda^{-1}) \) or \( p(w_d) = \mathcal{N}(0, \lambda_d^{-1}) \)

- Precision \( \lambda \) or \( \lambda_d \) specifies our belief about how close to zero \( w_d \) is (like regularization hyperparam)

- However, such a prior usually gives small weights but not very strong sparsity

Putting a gamma prior on precision can give sparsity (will soon see why)

Sparsity of weights will be a very useful thing to have in many models, e.g.,

- For linear model, this helps learn relevance of each feature \( x_d \)
- For kernel based model, this helps learn the relevance of each input \( x_n \) (Relevance Vector Machine)
Modeling Sparse Weights

Many probabilistic models consist of weights that are given zero-mean Gaussian priors, e.g.,

$$\mu(x) = \sum_{d=1}^{D} w_d x_d$$  \hspace{1cm} \text{(mean of a prob. lin reg model)}

$$\mu(x) = \sum_{n=1}^{N} w_n k(x_n, x)$$  \hspace{1cm} \text{(mean of a prob. kernel based nonlin reg model)}

A zero-mean prior is of the form $p(w_d) = \mathcal{N}(0, \lambda^{-1})$ or $p(w_d) = \mathcal{N}(0, \lambda_d^{-1})$

Precision $\lambda$ or $\lambda_d$ specifies our belief about how close to zero $w_d$ is (like regularization hyperparam)

However, such a prior usually gives small weights but not very strong sparsity

Putting a gamma prior on precision can give sparsity (will soon see why)
Modeling Sparse Weights

Many probabilistic models consist of weights that are given zero-mean Gaussian priors, e.g.,

$$\mu(x) = \sum_{d=1}^{D} w_d x_d$$

(mean of a prob. lin reg model)

$$\mu(x) = \sum_{n=1}^{N} w_n k(x_n, x)$$

(mean of a prob. kernel based nonlin reg model)

A zero-mean prior is of the form $p(w_d) = \mathcal{N}(0, \lambda^{-1})$ or $p(w_d) = \mathcal{N}(0, \lambda_d^{-1})$

Precision $\lambda$ or $\lambda_d$ specifies our belief about how close to zero $w_d$ is (like regularization hyperparam)

However, such a prior usually gives small weights but not very strong sparsity

Putting a gamma prior on precision can give sparsity (will soon see why)

Sparsity of weights will be a very useful thing to have in many models, e.g.,
Modeling Sparse Weights

Many probabilistic models consist of weights that are given zero-mean Gaussian priors, e.g.,

\[ \mu(x) = \sum_{d=1}^{D} w_d x_d \]  
(mean of a prob. lin reg model)

\[ \mu(x) = \sum_{n=1}^{N} w_n k(x_n, x) \]  
(mean of a prob. kernel based nonlin reg model)

A zero-mean prior is of the form \( p(w_d) = \mathcal{N}(0, \lambda^{-1}) \) or \( p(w_d) = \mathcal{N}(0, \lambda_d^{-1}) \)

Precision \( \lambda \) or \( \lambda_d \) specifies our belief about how close to zero \( w_d \) is (like regularization hyperparam)

However, such a prior usually gives small weights but not very strong sparsity

Putting a gamma prior on precision can give sparsity (will soon see why)

Sparsity of weights will be a very useful thing to have in many models, e.g.,

- For linear model, this helps learn relevance of each feature \( x_d \)
Modeling Sparse Weights

- Many probabilistic models consist of weights that are given zero-mean Gaussian priors, e.g.,

\[
\mu(x) = \sum_{d=1}^{D} w_d x_d \quad \text{(mean of a prob. lin reg model)}
\]

\[
\mu(x) = \sum_{n=1}^{N} w_n k(x_n, x) \quad \text{(mean of a prob. kernel based nonlin reg model)}
\]

- A zero-mean prior is of the form \( p(w_d) = \mathcal{N}(0, \lambda^{-1}) \) or \( p(w_d) = \mathcal{N}(0, \lambda_d^{-1}) \)

- Precision \( \lambda \) or \( \lambda_d \) specifies our belief about how close to zero \( w_d \) is (like regularization hyperparam)

- However, such a prior usually gives small weights but not very strong sparsity

- Putting a gamma prior on precision can give sparsity (will soon see why)

- Sparsity of weights will be a very useful thing to have in many models, e.g.,
  - For linear model, this helps learn relevance of each feature \( x_d \)
  - For kernel based model, this helps learn the relevance of each input \( x_n \) (Relevance Vector Machine)
Consider linear regression with prior $p(w_d | \lambda_d) = \mathcal{N}(0, \lambda_d^{-1})$ on each weight.

Let’s treat precision $\lambda_d$ as unknown and use a gamma (shape = $a$, rate = $b$) prior on it:

$$p(\lambda_d) = \text{Gamma}(a, b) = \frac{b^a}{\Gamma(a)} \lambda_d^{a-1} \exp(-b\lambda_d)$$
Sparsity via a Hierarchical Prior

- Consider linear regression with prior $p(w_d | \lambda_d) = \mathcal{N}(0, \lambda_d^{-1})$ on each weight.

- Let’s treat precision $\lambda_d$ as unknown and use a gamma (shape = $a$, rate = $b$) prior on it.

  $$p(\lambda_d) = \text{Gamma}(a, b) = \frac{b^a}{\Gamma(a)} \lambda_d^{a-1} \exp(-b\lambda_d)$$

- Marginalizing the precision leads to a Student-t prior on each $w_d$

  $$p(w_d) = \int p(w_d | \lambda_d)p(\lambda_d)d\lambda_d = \frac{b^a\Gamma(a + 1/2)}{\sqrt{2\pi}\Gamma(a)} \left(b + w_d^2/2\right)^{-(a+1/2)}$$
Sparsity via a Hierarchical Prior

- Consider linear regression with prior $p(w_d | \lambda_d) = \mathcal{N}(0, \lambda_d^{-1})$ on each weight.
- Let’s treat precision $\lambda_d$ as unknown and use a gamma (shape = $a$, rate = $b$) prior on it:
  
  $$p(\lambda_d) = \text{Gamma}(a, b) = \frac{b^a}{\Gamma(a)} \lambda_d^{a-1} \exp(-b\lambda_d)$$

- Marginalizing the precision leads to a Student-t prior on each $w_d$:
  
  $$p(w_d) = \int p(w_d | \lambda_d)p(\lambda_d)d\lambda_d = \frac{b^a\Gamma(a + 1/2)}{\sqrt{2\pi}\Gamma(a)}(b + w_d^2/2)^{-(a+1/2)}$$

- Note: Can make the prior an uninformative prior by setting $a$ and $b$ to be very small (e.g., $10^{-4}$).
Sparsity via a Hierarchical Prior

- Consider linear regression with prior $p(w_d | \lambda_d) = \mathcal{N}(0, \lambda_d^{-1})$ on each weight.
- Let’s treat precision $\lambda_d$ as unknown and use a gamma (shape = $a$, rate = $b$) prior on it:
  
  $$p(\lambda_d) = \text{Gamma}(a, b) = \frac{b^a}{\Gamma(a)} \lambda_d^{a-1} \exp(-b\lambda_d)$$

- Marginalizing the precision leads to a Student-t prior on each $w_d$
  
  $$p(w_d) = \int p(w_d | \lambda_d) p(\lambda_d) d\lambda_d = \frac{b^a \Gamma(a + 1/2)}{\sqrt{2\pi} \Gamma(a)} (b + w_d^2/2)^{-(a+1/2)}$$

- Note: Can make the prior an uninformative prior by setting $a$ and $b$ to be very small (e.g., $10^{-4}$).
- Note: Some other priors on $\lambda_d$ (e.g., exponential distribution) also result in sparse priors on $w_d$. 
Bayesian Linear Regression with Sparse Prior on Weights

- Posterior inference for $w$ not straightforward since $p(w) = \prod_{d=1}^{D} p(w_d)$ is no longer Gaussian
- Approximate inference is usually needed for inferring the full posterior
Posterior inference for $\mathbf{w}$ not straightforward since $p(\mathbf{w}) = \prod_{d=1}^{D} p(w_d)$ is no longer Gaussian.

Approximate inference is usually needed for inferring the full posterior.

Many approaches exist (which we will see later).

Working with such sparse priors is known as Sparse Bayesian Learning.

Used in many models where we want to have sparsity in the weights (very few non-zero weights).

Note: We will later look at other ways of getting sparsity (e.g., spike-and-slab priors defined by binary switch variables for each weight).
Bayesian Linear Regression with Sparse Prior on Weights

- Posterior inference for \( \mathbf{w} \) not straightforward since \( p(\mathbf{w}) = \prod_{d=1}^{D} p(w_d) \) is no longer Gaussian.
- Approximate inference is usually needed for inferring the full posterior.
- Many approaches exist (which we will see later).
- Such approaches are mostly in form of alternating estimation of \( \mathbf{w} \) and \( \lambda \).
Bayesian Linear Regression with Sparse Prior on Weights

- Posterior inference for $\mathbf{w}$ not straightforward since $p(\mathbf{w}) = \prod_{d=1}^{D} p(w_d)$ is no longer Gaussian
- Approximate inference is usually needed for inferring the full posterior
- Many approaches exist (which we will see later)
- Such approaches are mostly in form of alternating estimation of $\mathbf{w}$ and $\lambda$
  - Estimate $\lambda_d$ given $w_d$, estimate $w_d$ given $\lambda_d$
Bayesian Linear Regression with Sparse Prior on Weights

- Posterior inference for $\mathbf{w}$ not straightforward since $p(\mathbf{w}) = \prod_{d=1}^{D} p(w_d)$ is no longer Gaussian.
- Approximate inference is usually needed for inferring the full posterior.
- Many approaches exist (which we will see later).
- Such approaches are mostly in form of alternating estimation of $\mathbf{w}$ and $\lambda$.
  - Estimate $\lambda_d$ given $w_d$, estimate $w_d$ given $\lambda_d$.
- Popular approaches: EM, Gibbs sampling, variational inference, etc.
Bayesian Linear Regression with Sparse Prior on Weights

- Posterior inference for $\mathbf{w}$ not straightforward since $p(\mathbf{w}) = \prod_{d=1}^{D} p(w_d)$ is no longer Gaussian.
- Approximate inference is usually needed for inferring the full posterior.
- Many approaches exist (which we will see later).
- Such approaches are mostly in form of alternating estimation of $\mathbf{w}$ and $\lambda$.
  - Estimate $\lambda_d$ given $w_d$, estimate $w_d$ given $\lambda_d$.
- Popular approaches: EM, Gibbs sampling, variational inference, etc.
- Working with such sparse priors is known as **Sparse Bayesian Learning**.
Bayesian Linear Regression with Sparse Prior on Weights

- Posterior inference for $\mathbf{w}$ not straightforward since $p(\mathbf{w}) = \prod_{d=1}^{D} p(w_d)$ is no longer Gaussian
- Approximate inference is usually needed for inferring the full posterior
- Many approaches exist (which we will see later)
- Such approaches are mostly in form of alternating estimation of $\mathbf{w}$ and $\lambda$
  - Estimate $\lambda_d$ given $w_d$, estimate $w_d$ given $\lambda_d$
- Popular approaches: EM, Gibbs sampling, variational inference, etc
- Working with such sparse priors is known as **Sparse Bayesian Learning**
  - Used in many models where we want to have sparsity in the weights (very few non-zero weights)
Bayesian Linear Regression with Sparse Prior on Weights

- Posterior inference for $\mathbf{w}$ not straightforward since $p(\mathbf{w}) = \prod_{d=1}^{D} p(w_d)$ is no longer Gaussian
- Approximate inference is usually needed for inferring the full posterior
- Many approaches exist (which we will see later)
- Such approaches are mostly in form of alternating estimation of $\mathbf{w}$ and $\lambda$
  - Estimate $\lambda_d$ given $w_d$, estimate $w_d$ given $\lambda_d$
- Popular approaches: EM, Gibbs sampling, variational inference, etc
- Working with such sparse priors is known as Sparse Bayesian Learning
  - Used in many models where we want to have sparsity in the weights (very few non-zero weights)
- Note: We will later look at other ways of getting sparsity (e.g., spike-and-slab priors defined by binary switch variables for each weight)
Bayesian Logistic Regression

(..a simple, single-parameter, yet non-conjugate model)
The goal is to learn $p(y|x)$. Here $p(y|x)$ will be a discrete distribution (e.g., Bernoulli, multinoulli).

**Discriminative Classification:** Model and learn $p(y|x)$ directly.

This approach does not model the distribution of the inputs $x$.

**Generative Classification:** Model and learn $p(y|x)$ "indirectly" as $p(y|x) = p(y)p(x|y)p(x)$.

Called generative because, via $p(x|y)$, we model how the inputs $x$ of each class are generated.

The approach requires first learning class-marginal $p(y)$ and class-conditional distributions $p(x|y)$.

Usually harder to learn than discriminative but also has some advantages (more on this later).

Both approaches can be given a non-Bayesian or Bayesian treatment.

The Bayesian treatment won't rely on point estimates but infer the posterior over unknowns.
The goal is to learn $p(y|x)$. Here $p(y|x)$ will be a discrete distribution (e.g., Bernoulli, multinoulli).

Usually two approaches to learn $p(y|x)$: Discriminative Classification and Generative Classification.
Probabilistic Models for Classification

- The goal is to learn $p(y|x)$. Here $p(y|x)$ will be a discrete distribution (e.g., Bernoulli, multinoulli).
- Usually two approaches to learn $p(y|x)$: **Discriminative Classification** and **Generative Classification**.
- **Discriminative Classification**: Model and learn $p(y|x)$ directly.
  - This approach does not model the distribution of the inputs $x$.
- **Generative Classification**: Model and learn $p(y|x)$ "indirectly" as $p(y|x) = p(y)p(x|y)p(x)$.
  - Called generative because, via $p(x|y)$, we model how the inputs $x$ of each class are generated.
  - The approach requires first learning class-marginal $p(y)$ and class-conditional distributions $p(x|y)$.
    - Usually harder to learn than discriminative but also has some advantages (more on this later).
  - Both approaches can be given a non-Bayesian or Bayesian treatment.
    - The Bayesian treatment won’t rely on point estimates but infer the posterior over unknowns.
Probabilistic Models for Classification

- The goal is to learn $p(y|x)$. Here $p(y|x)$ will be a discrete distribution (e.g., Bernoulli, multinoulli)
- Usually two approaches to learn $p(y|x)$: **Discriminative Classification** and **Generative Classification**
- **Discriminative Classification**: Model and learn $p(y|x)$ directly
  - This approach does not model the distribution of the inputs $x$
The goal is to learn \( p(y|x) \). Here \( p(y|x) \) will be a discrete distribution (e.g., Bernoulli, multinoulli).

Usually two approaches to learn \( p(y|x) \): Discriminative Classification and Generative Classification.

**Discriminative Classification:** Model and learn \( p(y|x) \) **directly**
- This approach does not model the distribution of the inputs \( x \).

**Generative Classification:** Model and learn \( p(y|x) \) "indirectly"
The goal is to learn $p(y|x)$. Here $p(y|x)$ will be a discrete distribution (e.g., Bernoulli, multinoulli).

Usually two approaches to learn $p(y|x)$: **Discriminative Classification** and **Generative Classification**.

**Discriminative Classification**: Model and learn $p(y|x)$ *directly*

This approach does not model the distribution of the inputs $x$.

**Generative Classification**: Model and learn $p(y|x)$ “indirectly” as $p(y|x) = \frac{p(y)p(x|y)}{p(x)}$
The goal is to learn $p(y|x)$. Here $p(y|x)$ will be a discrete distribution (e.g., Bernoulli, multinoulli).

Usually two approaches to learn $p(y|x)$: **Discriminative Classification** and **Generative Classification**.

**Discriminative Classification**: Model and learn $p(y|x)$ directly
- This approach does not model the distribution of the inputs $x$.

**Generative Classification**: Model and learn $p(y|x)$ “indirectly” as $p(y|x) = \frac{p(y)p(x|y)}{p(x)}$
- Called generative because, via $p(x|y)$, we model how the inputs $x$ of each class are generated.
The goal is to learn $p(y|x)$. Here $p(y|x)$ will be a discrete distribution (e.g., Bernoulli, multinoulli).

Usually two approaches to learn $p(y|x)$: **Discriminative Classification** and **Generative Classification**.

**Discriminative Classification:** Model and learn $p(y|x)$ **directly**

- This approach does not model the distribution of the inputs $x$.

**Generative Classification:** Model and learn $p(y|x)$ “indirectly” as $p(y|x) = \frac{p(y)p(x|y)}{p(x)}$

- Called generative because, via $p(x|y)$, we model how the inputs $x$ of each class are generated.
- The approach requires first learning **class-marginal** $p(y)$ and **class-conditional** distributions $p(x|y)$. 

Probabilistic Models for Classification
The goal is to learn $p(y|x)$. Here $p(y|x)$ will be a discrete distribution (e.g., Bernoulli, multinoulli).

Usually two approaches to learn $p(y|x)$: **Discriminative Classification** and **Generative Classification**.

- **Discriminative Classification**: Model and learn $p(y|x)$ directly.
  - This approach does not model the distribution of the inputs $x$.

- **Generative Classification**: Model and learn $p(y|x)$ “indirectly” as $p(y|x) = \frac{p(y)p(x|y)}{p(x)}$.
  - Called generative because, via $p(x|y)$, we model how the inputs $x$ of each class are generated.
  - The approach requires first learning **class-marginal** $p(y)$ and **class-conditional** distributions $p(x|y)$.
  - Usually harder to learn than discriminative but also has some advantages (more on this later).
The goal is to learn $p(y|x)$. Here $p(y|x)$ will be a discrete distribution (e.g., Bernoulli, multinoulli).

Usually two approaches to learn $p(y|x)$: **Discriminative Classification** and **Generative Classification**

**Discriminative Classification**: Model and learn $p(y|x)$ directly

- This approach does not model the distribution of the inputs $x$

**Generative Classification**: Model and learn $p(y|x)$ “indirectly” as $p(y|x) = \frac{p(y)p(x|y)}{p(x)}$

- Called generative because, via $p(x|y)$, we model how the inputs $x$ of each class are generated
- The approach requires first learning **class-marginal** $p(y)$ and **class-conditional** distributions $p(x|y)$
- Usually harder to learn than discriminative but also has some advantages (more on this later)

Both approaches can be given a non-Bayesian or Bayesian treatment
The goal is to learn $p(y|x)$. Here $p(y|x)$ will be a discrete distribution (e.g., Bernoulli, multinoulli).

Usually two approaches to learn $p(y|x)$: **Discriminative** Classification and **Generative** Classification.

**Discriminative Classification**: Model and learn $p(y|x)$ directly.
- This approach does not model the distribution of the inputs $x$.

**Generative Classification**: Model and learn $p(y|x)$ “indirectly” as $p(y|x) = \frac{p(y)p(x|y)}{p(x)}$.
- Called generative because, via $p(x|y)$, we model how the inputs $x$ of each class are generated.
- The approach requires first learning **class-marginal** $p(y)$ and **class-conditional** distributions $p(x|y)$.
- Usually harder to learn than discriminative but also has some advantages (more on this later).

Both approaches can be given a non-Bayesian or Bayesian treatment.
- The Bayesian treatment won’t rely on point estimates but infer the posterior over unknowns.
Discriminative Classification via Logistic Regression

- **Logistic Regression** (LR) is an example of discriminative binary classification, i.e., $y \in \{0, 1\}$.

Logistic Regression models $x$ to $y$ relationship using the sigmoid function:

$$p(y = 1 | x, w) = \frac{1}{1 + \exp(-w^\top x)}$$

where $w \in \mathbb{R}^D$ is the weight vector.

Also note that $p(y = 0 | x, w) = 1 - p(y = 1 | x, w)$.

A large positive (negative) "score" $w^\top x$ means large probability of label being 1 (0).

Is sigmoid the only way to convert the score into a probability? No, while LR does that, there exist models that define $p(y = 1 | x, w)$ in other ways. E.g. Probit Regression:

$$p(y = 1 | x, w) = \Phi(w^\top x)$$

(where $\Phi$ denotes the CDF of $N(0, 1)$).
Discriminative Classification via Logistic Regression

- **Logistic Regression** (LR) is an example of discriminative binary classification, i.e., $y \in \{0, 1\}$.
- Logistic Regression models $x$ to $y$ relationship using the sigmoid function

\[
p(y = 1 | x, w) = \mu = \sigma(w^\top x) = \frac{1}{1 + \exp(-w^\top x)} = \exp(w^\top x) \frac{1}{1 + \exp(w^\top x)}
\]

where $w \in \mathbb{R}^D$ is the weight vector.

Also note that $p(y = 0 | x, w) = 1 - \mu$.

A large positive (negative) "score" $w^\top x$ means large probability of label being 1 (0).

Is sigmoid the only way to convert the score into a probability? No, while LR does that, there exist models that define $\mu$ in other ways. E.g. Probit Regression

\[
\mu = p(y = 1 | x, w) = \Phi(w^\top x)
\]

(where $\Phi$ denotes the CDF of $N(0, 1)$)
Discriminative Classification via Logistic Regression

- **Logistic Regression** (LR) is an example of discriminative binary classification, i.e., \( y \in \{0, 1\} \)
- Logistic Regression models \( x \) to \( y \) relationship using the **sigmoid function**

\[
p(y = 1|x, w) = \frac{1}{1 + \exp(-w^\top x)}
\]

where \( w \in \mathbb{R}^D \) is the weight vector.

Also note that \( p(y = 0|x, w) = 1 - \mu \).

A large positive (negative) “score” \( w^\top x \) means large probability of label being 1 (0).

Is sigmoid the only way to convert the score into a probability? No, while LR does that, there exist models that define \( \mu \) in other ways. E.g. Probit Regression

\[
\mu = p(y = 1|x, w) = \Phi(w^\top x)
\]

(where \( \Phi \) denotes the CDF of \( N(0, 1) \))
Discriminative Classification via Logistic Regression

- **Logistic Regression** (LR) is an example of discriminative binary classification, i.e., \( y \in \{0, 1\} \)
- Logistic Regression models \( x \) to \( y \) relationship using the sigmoid function
  \[
P(y = 1|x, \mathbf{w}) = \mu
  \]
  
  where \( \mathbf{w} \in \mathbb{R}^D \) is the weight vector.

  Also note that \( p(y = 0|x, \mathbf{w}) = 1 - \mu \)

  A large positive (negative) "score" \( \mathbf{w}^\top x \) means large probability of label being 1 (0)

  Is sigmoid the only way to convert the score into a probability?
  No, while LR does that, there exist models that define \( \mu \) in other ways. E.g. Probit Regression
  \[
  \mu = p(y = 1|x, \mathbf{w}) = \Phi(\mathbf{w}^\top x)
  \]
  (where \( \Phi \) denotes the CDF of \( N(0, 1) \))
Logistic Regression (LR) is an example of discriminative binary classification, i.e., $y \in \{0, 1\}$.

Logistic Regression models $x$ to $y$ relationship using the sigmoid function:

$$p(y = 1|x, w) = \mu = \sigma(w^\top x)$$
Discriminative Classification via Logistic Regression

- **Logistic Regression** (LR) is an example of discriminative binary classification, i.e., $y \in \{0, 1\}$
- Logistic Regression models $x$ to $y$ relationship using the sigmoid function
  \[ p(y = 1|x, w) = \mu = \sigma(w^\top x) = \frac{1}{1 + \exp(-w^\top x)} \]

A large positive (negative) "score" $w^\top x$ means large probability of label being 1 (0)

Is sigmoid the only way to convert the score into a probability? No, while LR does that, there exist models that define $\mu$ in other ways. E.g. Probit Regression

\[ \mu = p(y = 1|x, w) = \Phi(w^\top x) \]

(where $\Phi$ denotes the CDF of $N(0, 1)$)
Discriminative Classification via Logistic Regression

- **Logistic Regression** (LR) is an example of discriminative binary classification, i.e., \( y \in \{0, 1\} \).
- Logistic Regression models \( x \) to \( y \) relationship using the sigmoid function

\[
p(y = 1|x, w) = \mu = \sigma(w^T x) = \frac{1}{1 + \exp(-w^T x)} = \frac{\exp(w^T x)}{1 + \exp(w^T x)}
\]

where \( w \in \mathbb{R}^D \) is the weight vector.

Also note that \( p(y = 0|x, w) = 1 - \mu \).

A large positive (negative) "score" \( w^T x \) means large probability of label being 1 (0).

Is sigmoid the only way to convert the score into a probability? No, while LR does that, there exist models that define \( \mu \) in other ways. E.g. Probit Regression

\[
\mu = \Phi(w^T x)
\]

(where \( \Phi \) denotes the CDF of \( N(0, 1) \)).
Discriminative Classification via Logistic Regression

- **Logistic Regression** (LR) is an example of discriminative binary classification, i.e., $y \in \{0, 1\}$

Logistic Regression models $x$ to $y$ relationship using the sigmoid function

$$p(y = 1|x, w) = \mu = \sigma(w^\top x) = \frac{1}{1 + \exp(-w^\top x)} = \frac{\exp(w^\top x)}{1 + \exp(w^\top x)}$$

where $w \in \mathbb{R}^D$ is the weight vector.
Discriminative Classification via Logistic Regression

- **Logistic Regression** (LR) is an example of discriminative binary classification, i.e., \( y \in \{0, 1\} \)
- Logistic Regression models the relationship between \( x \) and \( y \) using the sigmoid function:
  \[
p(y = 1|x, w) = \mu = \sigma(w^\top x) = \frac{1}{1 + \exp(-w^\top x)} = \frac{\exp(w^\top x)}{1 + \exp(w^\top x)}
\]
  where \( w \in \mathbb{R}^D \) is the weight vector. Also note that \( p(y = 0|x, w) = 1 - \mu \)
Discriminative Classification via Logistic Regression

- **Logistic Regression** (LR) is an example of discriminative binary classification, i.e., $y \in \{0, 1\}$
- Logistic Regression models $x$ to $y$ relationship using the sigmoid function
  \[ p(y = 1|x, w) = \mu = \sigma(w^\top x) = \frac{1}{1 + \exp(-w^\top x)} = \frac{\exp(w^\top x)}{1 + \exp(w^\top x)} \]
  where $w \in \mathbb{R}^D$ is the weight vector. Also note that $p(y = 0|x, w) = 1 - \mu$
Discriminative Classification via Logistic Regression

- **Logistic Regression** (LR) is an example of discriminative binary classification, i.e., $y \in \{0, 1\}$
- Logistic Regression models $x$ to $y$ relationship using the sigmoid function
  $$p(y = 1|x, w) = \mu = \sigma(w^T x) = \frac{1}{1 + \exp(-w^T x)} = \frac{\exp(w^T x)}{1 + \exp(w^T x)}$$
  where $w \in \mathbb{R}^D$ is the weight vector. Also note that $p(y = 0|x, w) = 1 - \mu$

- A large positive (negative) "score" $w^T x$ means large probability of label being 1 (0)
**Discriminative Classification via Logistic Regression**

- **Logistic Regression** (LR) is an example of discriminative **binary** classification, i.e., $y \in \{0, 1\}$
- Logistic Regression models $x$ to $y$ relationship using the **sigmoid function**

$$p(y = 1|x, w) = \mu = \sigma(w^\top x) = \frac{1}{1 + \exp(-w^\top x)} = \frac{\exp(w^\top x)}{1 + \exp(w^\top x)}$$

where $w \in \mathbb{R}^D$ is the weight vector. Also note that $p(y = 0|x, w) = 1 - \mu$

- A large positive (negative) “score” $w^\top x$ means large probability of label being 1 (0)
- Is sigmoid the only way to convert the score into a probability?
Logistic Regression (LR) is an example of discriminative binary classification, i.e., \( y \in \{0, 1\} \)

Logistic Regression models \( x \) to \( y \) relationship using the sigmoid function

\[
p(y = 1 | x, w) = \mu = \sigma(w^T x) = \frac{1}{1 + \exp(-w^T x)} = \frac{\exp(w^T x)}{1 + \exp(w^T x)}
\]

where \( w \in \mathbb{R}^D \) is the weight vector. Also note that \( p(y = 0 | x, w) = 1 - \mu \)

A large positive (negative) “score” \( w^T x \) means large probability of label being 1 (0)

Is sigmoid the only way to convert the score into a probability?

- No, while LR does that, there exist models that define \( \mu \) in other ways
Discriminative Classification via Logistic Regression

- **Logistic Regression** (LR) is an example of discriminative binary classification, i.e., $y \in \{0, 1\}$
- Logistic Regression models $x$ to $y$ relationship using the sigmoid function
  \[
  p(y = 1|x, w) = \mu = \sigma(w^\top x) = \frac{1}{1 + \exp(-w^\top x)} = \frac{\exp(w^\top x)}{1 + \exp(w^\top x)}
  \]
  where $w \in \mathbb{R}^D$ is the weight vector. Also note that $p(y = 0|x, w) = 1 - \mu$

- A large positive (negative) “score” $w^\top x$ means large probability of label being 1 (0)
- Is sigmoid the only way to convert the score into a probability?
  - No, while LR does that, there exist models that define $\mu$ in other ways. E.g. **Probit Regression**
    \[
    \mu = p(y = 1|x, w) = \Phi(w^\top x)
    \]
Discriminative Classification via Logistic Regression

- **Logistic Regression** (LR) is an example of discriminative binary classification, i.e., $y \in \{0, 1\}$
- Logistic Regression models $x$ to $y$ relationship using the sigmoid function
  \[
  p(y = 1|x, w) = \mu = \sigma(w^\top x) = \frac{1}{1 + \exp(-w^\top x)} = \frac{\exp(w^\top x)}{1 + \exp(w^\top x)}
  \]
  where $w \in \mathbb{R}^D$ is the weight vector. Also note that $p(y = 0|x, w) = 1 - \mu$

A large positive (negative) "score" $w^\top x$ means large probability of label being 1 (0)

- Is sigmoid the only way to convert the score into a probability?
  - No, while LR does that, there exist models that define $\mu$ in other ways. E.g. **Probit Regression**
    \[
    \mu = p(y = 1|x, w) = \Phi(w^\top x) \quad \text{(where } \Phi \text{ denotes the CDF of } \mathcal{N}(0, 1))
    \]
Logistic Regression

The LR classification rule is

\[ p(y = 1|x, \mathbf{w}) = \mu = \sigma(\mathbf{w}^\top \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x})} = \frac{\exp(\mathbf{w}^\top \mathbf{x})}{1 + \exp(\mathbf{w}^\top \mathbf{x})} \]
Logistic Regression

- The LR classification rule is

\[
p(y = 1 | x, \mathbf{w}) = \mu = \sigma(\mathbf{w}^\top x) = \frac{1}{1 + \exp(-\mathbf{w}^\top x)} = \frac{\exp(\mathbf{w}^\top x)}{1 + \exp(\mathbf{w}^\top x)}
\]

\[
p(y = 0 | x, \mathbf{w}) = 1 - \mu = 1 - \sigma(\mathbf{w}^\top x) = \frac{1}{1 + \exp(\mathbf{w}^\top x)}
\]
Logistic Regression

- The LR classification rule is
  \[ p(y = 1|x, w) = \mu = \sigma(w^T x) = \frac{1}{1 + \exp(-w^T x)} = \frac{\exp(w^T x)}{1 + \exp(w^T x)} \]
  \[ p(y = 0|x, w) = 1 - \mu = 1 - \sigma(w^T x) = \frac{1}{1 + \exp(w^T x)} \]

- This implies a Bernoulli likelihood model for the labels
  \[ p(y|x, w) = \text{Bernoulli}(\sigma(w^T x)) \]
Logistic Regression

The LR classification rule is

\[
p(y = 1|x, w) = \mu = \sigma(w^Tx) = \frac{1}{1 + \exp(-w^Tx)} = \frac{\exp(w^Tx)}{1 + \exp(w^Tx)}
\]

\[
p(y = 0|x, w) = 1 - \mu = 1 - \sigma(w^Tx) = \frac{1}{1 + \exp(w^Tx)}
\]

This implies a Bernoulli likelihood model for the labels

\[
p(y|x, w) = \text{Bernoulli}(\sigma(w^Tx)) = \left[ \frac{\exp(w^Tx)}{1 + \exp(w^Tx)} \right]^y \left[ \frac{1}{1 + \exp(w^Tx)} \right]^{(1-y)}
\]
Logistic Regression

- The LR classification rule is

\[ p(y = 1|x, w) = \mu = \sigma(w^T x) = \frac{1}{1 + \exp(-w^T x)} = \frac{\exp(w^T x)}{1 + \exp(w^T x)} \]

\[ p(y = 0|x, w) = 1 - \mu = 1 - \sigma(w^T x) = \frac{1}{1 + \exp(w^T x)} \]

- This implies a Bernoulli likelihood model for the labels

\[ p(y|x, w) = \text{Bernoulli}(\sigma(w^T x)) = \left[ \frac{\exp(w^T x)}{1 + \exp(w^T x)} \right]^y \left[ \frac{1}{1 + \exp(w^T x)} \right]^{1-y} \]

- Given \( N \) observations \((X, y) = \{x_n, y_n\}_{n=1}^N\), we can do point estimation for \( w \) by maximizing the log-likelihood (or minimizing the negative log-likelihood).
Logistic Regression

- The LR classification rule is
  \[ p(y = 1|x, w) = \mu = \sigma(w^T x) = \frac{1}{1 + \exp(-w^T x)} = \frac{\exp(w^T x)}{1 + \exp(w^T x)} \]
  \[ p(y = 0|x, w) = 1 - \mu = 1 - \sigma(w^T x) = \frac{1}{1 + \exp(w^T x)} \]

- This implies a Bernoulli likelihood model for the labels
  \[ p(y|x, w) = \text{Bernoulli}(\sigma(w^T x)) = \left[ \frac{\exp(w^T x)}{1 + \exp(w^T x)} \right]^y \left[ \frac{1}{1 + \exp(w^T x)} \right]^{1-y} \]

- Given \( N \) observations \((X, y) = \{x_n, y_n\}_{n=1}^N\), we can do point estimation for \( w \) by maximizing the log-likelihood (or minimizing the negative log-likelihood). This is basically MLE.
  \[ w_{MLE} = \arg \max_w \sum_{n=1}^N \log p(y_n|x_n, w) \]
Logistic Regression

The LR classification rule is

\[ p(y = 1| x, w) = \mu = \sigma(w^T x) = \frac{1}{1 + \exp(-w^T x)} = \frac{\exp(w^T x)}{1 + \exp(w^T x)} \]

\[ p(y = 0| x, w) = 1 - \mu = 1 - \sigma(w^T x) = \frac{1}{1 + \exp(w^T x)} \]

This implies a Bernoulli likelihood model for the labels

\[ p(y|x, w) = \text{Bernoulli}(\sigma(w^T x)) = \left[ \frac{\exp(w^T x)}{1 + \exp(w^T x)} \right]^y \left[ \frac{1}{1 + \exp(w^T x)} \right]^{1-y} \]

Given \( N \) observations \((X, y) = \{x_n, y_n\}_{n=1}^N\), we can do point estimation for \( w \) by maximizing the log-likelihood (or minimizing the negative log-likelihood). This is basically MLE.

\[ w_{MLE} = \arg \max_w \sum_{n=1}^N \log p(y_n|x_n, w) = \arg \min_w -\sum_{n=1}^N \log p(y_n|x_n, w) \]
Logistic Regression

- The LR classification rule is
  \[ p(y = 1 | x, w) = \mu = \sigma(w^T x) = \frac{1}{1 + \exp(-w^T x)} = \frac{\exp(w^T x)}{1 + \exp(w^T x)} \]
  \[ p(y = 0 | x, w) = 1 - \mu = 1 - \sigma(w^T x) = \frac{1}{1 + \exp(w^T x)} \]

- This implies a Bernoulli likelihood model for the labels
  \[ p(y | x, w) = \text{Bernoulli}(\sigma(w^T x)) = \left[ \frac{\exp(w^T x)}{1 + \exp(w^T x)} \right]^y \left[ \frac{1}{1 + \exp(w^T x)} \right]^{1-y} \]

- Given \( N \) observations \((X, y) = \{x_n, y_n\}_{n=1}^N\), we can do point estimation for \( w \) by maximizing the log-likelihood (or minimizing the negative log-likelihood). This is basically MLE.
  \[ w_{MLE} = \text{arg max}_w \sum_{n=1}^N \log p(y_n | x_n, w) = \text{arg min}_w - \sum_{n=1}^N \log p(y_n | x_n, w) = \text{arg min}_w \text{NLL}(w) \]
Logistic Regression

The LR classification rule is

\[ p(y = 1|x, w) = \mu = \sigma(w^T x) = \frac{1}{1 + \exp(-w^T x)} = \frac{\exp(w^T x)}{1 + \exp(w^T x)} \]

\[ p(y = 0|x, w) = 1 - \mu = 1 - \sigma(w^T x) = \frac{1}{1 + \exp(w^T x)} \]

This implies a Bernoulli likelihood model for the labels

\[ p(y|x, w) = \text{Bernoulli}(\sigma(w^T x)) = \left[ \frac{\exp(w^T x)}{1 + \exp(w^T x)} \right]^y \left[ \frac{1}{1 + \exp(w^T x)} \right]^{1-y} \]

Given \( N \) observations \((X, y) = \{x_n, y_n\}_{n=1}^N\), we can do point estimation for \( w \) by maximizing the log-likelihood (or minimizing the negative log-likelihood). This is basically MLE.

\[ w_{MLE} = \arg \max_w \sum_{n=1}^N \log p(y_n|x_n, w) = \arg \min_w \sum_{n=1}^N \log p(y_n|x_n, w) = \arg \min_w NLL(w) \]

Logistic Regression

- The LR classification rule is

\[ p(y = 1| x, w) = \mu = \sigma(w^T x) = \frac{1}{1 + \exp(-w^T x)} = \frac{\exp(w^T x)}{1 + \exp(w^T x)} \]

\[ p(y = 0| x, w) = 1 - \mu = 1 - \sigma(w^T x) = \frac{1}{1 + \exp(w^T x)} \]

- This implies a Bernoulli likelihood model for the labels

\[ p(y|x, w) = \text{Bernoulli}(\sigma(w^T x)) = \left[ \frac{\exp(w^T x)}{1 + \exp(w^T x)} \right]^y \left[ \frac{1}{1 + \exp(w^T x)} \right]^{1-y} \]

- Given \( N \) observations \( (X, y) = \{x_n, y_n\}_{n=1}^N \), we can do point estimation for \( w \) by maximizing the log-likelihood (or minimizing the negative log-likelihood). This is basically MLE.

\[ w_{MLE} = \arg \max_w \sum_{n=1}^N \log p(y_n|x_n, w) = \arg \min_w -\sum_{n=1}^N \log p(y_n|x_n, w) = \arg \min_w \text{NLL}(w) \]


- Can also add a regularizer on \( w \) to prevent overfitting. This corresponds to doing MAP estimation with a prior on \( w \), i.e., \( w_{MAP} = \arg \max_w [\sum_{n=1}^N \log p(y_n|x_n, w) + \log p(w)] \).
MLE/MAP only gives a point estimate. We would like to infer the full posterior over $w$. 

Recall that the likelihood model is Bernoulli:

$$p(y | x, w) = \text{Bernoulli}(\sigma(w^\top x)) = \begin{cases} \exp(w^\top x) & y = 1 \\ 1 & y = 0, \frac{1}{1 + \exp(w^\top x)} \end{cases}$$

Just like the Bayesian linear regression case, let’s use a Gaussian prior on $w$:

$$p(w) = N(0, \lambda^{-1}I_D) \propto \exp\left(-\frac{\lambda}{2}w^\top w\right)$$

Given $N$ observations $(X, y) =$ \{ $x_n, y_n$ \}$_{n=1}^N$, where $X$ is $N \times D$ and $y$ is $N \times 1$, the posterior over $w$ is:

$$p(w | X, y) = \frac{p(y | X, w)p(w)}{\int p(y | X, w)p(w) dw} = \prod_{n=1}^N p(y_n | x_n, w)p(w)$$

The denominator is intractable in general (logistic-Bernoulli and Gaussian are not conjugate). Can’t get a closed form expression for $p(w | X, y)$. Must approximate it! Several ways to do it, e.g., MCMC, variational inference, Laplace approximation (next class).
Bayesian Logistic Regression

- MLE/MAP only gives a point estimate. We would like to infer the full posterior over \( w \).
- Recall that the likelihood model is Bernoulli

\[
p(y|x, w) = \text{Bernoulli}(\sigma(w^T x)) = \left[ \frac{\exp(w^T x)}{1 + \exp(w^T x)} \right]^y \left[ \frac{1}{1 + \exp(w^T x)} \right]^{1-y}
\]
Bayesian Logistic Regression

- MLE/MAP only gives a point estimate. We would like to infer the full posterior over $\mathbf{w}$
- Recall that the likelihood model is Bernoulli

$$p(y|x, \mathbf{w}) = \text{Bernoulli}(\sigma(\mathbf{w}^\top \mathbf{x})) = \left[ \frac{\exp(\mathbf{w}^\top \mathbf{x})}{1 + \exp(\mathbf{w}^\top \mathbf{x})} \right]^y \left[ \frac{1}{1 + \exp(\mathbf{w}^\top \mathbf{x})} \right]^{(1-y)}$$

- Just like the Bayesian linear regression case, let’s use a Gaussian prior on $\mathbf{w}$

$$p(\mathbf{w}) = \mathcal{N}(0, \lambda^{-1} \mathbf{I}_D) \propto \exp(-\frac{\lambda}{2} \mathbf{w}^\top \mathbf{w})$$
MLE/MAP only gives a point estimate. We would like to infer the full posterior over \( w \).

Recall that the likelihood model is Bernoulli:

\[
p(y|x, w) = \text{Bernoulli}(\sigma(w^T x)) = \left[ \frac{\exp(w^T x)}{1 + \exp(w^T x)} \right]^y \left[ \frac{1}{1 + \exp(w^T x)} \right]^{1-y}
\]

Just like the Bayesian linear regression case, let’s use a Gaussian prior on \( w \):

\[
p(w) = \mathcal{N}(0, \lambda^{-1} I_D) \propto \exp\left(-\frac{\lambda}{2} w^T w\right)
\]

Given \( N \) observations \((X, y) = \{x_n, y_n\}_{n=1}^N\), where \( X \) is \( N \times D \) and \( y \) is \( N \times 1 \), the posterior over \( w \):

\[
p(w|X, y) = \frac{p(y|X, w)p(w)}{\int p(y|X, w)p(w)dw}
\]
Bayesian Logistic Regression

- MLE/MAP only gives a point estimate. We would like to infer the full posterior over $\mathbf{w}$
- Recall that the **likelihood model** is Bernoulli

\[
p(y|x, \mathbf{w}) = \text{Bernoulli}(\sigma(\mathbf{w}^\top \mathbf{x})) = \left[ \frac{\exp(\mathbf{w}^\top \mathbf{x})}{1 + \exp(\mathbf{w}^\top \mathbf{x})} \right]^y \left[ \frac{1}{1 + \exp(\mathbf{w}^\top \mathbf{x})} \right]^{(1-y)}
\]

- Just like the Bayesian linear regression case, let’s use a Gaussian **prior** on $\mathbf{w}$

\[
p(\mathbf{w}) = \mathcal{N}(0, \lambda^{-1} I_D) \propto \exp\left(-\frac{\lambda}{2} \mathbf{w}^\top \mathbf{w}\right)
\]

- Given $N$ observations $(\mathbf{X}, \mathbf{y}) = \{\mathbf{x}_n, y_n\}_{n=1}^N$, where $\mathbf{X}$ is $N \times D$ and $\mathbf{y}$ is $N \times 1$, the posterior over $\mathbf{w}$

\[
p(\mathbf{w}|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w})}{\int p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w})d\mathbf{w}} = \frac{\prod_{n=1}^N p(y_n|x_n, \mathbf{w})p(\mathbf{w})}{\int \prod_{n=1}^N p(y_n|x_n, \mathbf{w})p(\mathbf{w})d\mathbf{w}}
\]
Bayesian Logistic Regression

- MLE/MAP only gives a point estimate. We would like to infer the full posterior over $w$.
- Recall that the likelihood model is Bernoulli:

$$p(y|x, w) = \text{Bernoulli}(\sigma(w^\top x)) = \left[ \frac{\exp(w^\top x)}{1 + \exp(w^\top x)} \right]^y \left[ \frac{1}{1 + \exp(w^\top x)} \right]^{(1-y)}$$

- Just like the Bayesian linear regression case, let’s use a Gaussian prior on $w$:

$$p(w) = \mathcal{N}(0, \lambda^{-1}I_D) \propto \exp\left(-\frac{\lambda}{2}w^\top w\right)$$

- Given $N$ observations $(X, y) = \{x_n, y_n\}_{n=1}^N$, where $X$ is $N \times D$ and $y$ is $N \times 1$, the posterior over $w$:

$$p(w|X, y) = \frac{p(y|X, w)p(w)}{\int p(y|X, w)p(w)dw} = \frac{\prod_{n=1}^N p(y_n|x_n, w)p(w)}{\int \prod_{n=1}^N p(y_n|x_n, w)p(w)dw}$$

- The denominator is intractable in general (logistic-Bernoulli and Gaussian are not conjugate).
Bayesian Logistic Regression

- MLE/MAP only gives a point estimate. We would like to infer the full posterior over $w$
- Recall that the likelihood model is Bernoulli
  \[
p(y|x, w) = \text{Bernoulli}(\sigma(w^\top x)) = \left[ \frac{\exp(w^\top x)}{1 + \exp(w^\top x)} \right]^y \left[ \frac{1}{1 + \exp(w^\top x)} \right]^{(1-y)}\]
- Just like the Bayesian linear regression case, let’s use a Gaussian prior on $w$
  \[
p(w) = \mathcal{N}(0, \lambda^{-1}I_D) \propto \exp(-\frac{\lambda}{2}w^\top w)\]
- Given $N$ observations $(X, y) = \{x_n, y_n\}_{n=1}^N$, where $X$ is $N \times D$ and $y$ is $N \times 1$, the posterior over $w$
  \[
p(w|X, y) = \frac{p(y|X, w)p(w)}{\int p(y|X, w)p(w)dw} = \frac{\prod_{n=1}^N p(y_n|x_n, w)p(w)}{\int \prod_{n=1}^N p(y_n|x_n, w)p(w)dw}\]
- The denominator is intractable in general (logistic-Bernoulli and Gaussian are not conjugate)
  - Can’t get a closed form expression for $p(w|X, y)$. Must approximate it!
Bayesian Logistic Regression

- MLE/MAP only gives a point estimate. We would like to infer the full posterior over $w$
- Recall that the likelihood model is Bernoulli

$$p(y|x, w) = \text{Bernoulli}(\sigma(w^\top x)) = \left[ \frac{\exp(w^\top x)}{1 + \exp(w^\top x)} \right]^y \left[ \frac{1}{1 + \exp(w^\top x)} \right]^{1-y}$$

- Just like the Bayesian linear regression case, let’s use a Gaussian prior on $w$

$$p(w) = \mathcal{N}(0, \lambda^{-1} I_D) \propto \exp\left(-\frac{\lambda}{2} w^\top w\right)$$

- Given $N$ observations $(X, y) = \{x_n, y_n\}_{n=1}^N$, where $X$ is $N \times D$ and $y$ is $N \times 1$, the posterior over $w$

$$p(w|X, y) = \frac{p(y|X, w)p(w)}{\int p(y|X, w)p(w)dw} = \frac{\prod_{n=1}^N p(y_n|x_n, w)p(w)}{\int \prod_{n=1}^N p(y_n|x_n, w)p(w)dw}$$

- The denominator is intractable in general (logistic-Bernoulli and Gaussian are not conjugate)

  - Can’t get a closed form expression for $p(w|X, y)$. Must approximate it!
  - Several ways to do it, e.g., MCMC, variational inference, Laplace approximation (next class)
Next Class

- Laplace approximation
- Computing posterior and posterior predictive for logistic regression
- Properties/benefits of Bayesian logistic regression
- Bayesian approach to generative classification