Logistic Regression

Logistic Regression: The Model

- A model for doing *probabilistic* binary classification
- Predicts *label probabilities* rather than a hard value of the label

$$
\begin{aligned}
p(y_n = 1|\boldsymbol{x}_n, \boldsymbol{w}) &= \mu_n \\
p(y_n = 0|\boldsymbol{x}_n, \boldsymbol{w}) &= 1 - \mu_n
\end{aligned}
$$

Logistic Regression: The Model

- A model for doing *probabilistic* binary classification
- Predicts *label probabilities* rather than a hard value of the label

$$\begin{aligned} p(y_n = 1|\boldsymbol{x}_n, \boldsymbol{w}) &= \mu_n \\ p(y_n = 0|\boldsymbol{x}_n, \boldsymbol{w}) &= 1 - \mu_n \end{aligned}$$

- The model's prediction is a probability defined using the sigmoid function

$$f(\boldsymbol{x}_n) = \mu_n = \sigma(\mathbf{w}^\top \mathbf{x}_n) = \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x}_n)} = \frac{\exp(\mathbf{w}^\top \mathbf{x}_n)}{1 + \exp(\mathbf{w}^\top \mathbf{x}_n)}$$

Logistic Regression: The Model

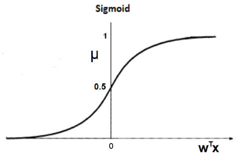
- A model for doing *probabilistic* binary classification
- Predicts *label probabilities* rather than a hard value of the label

$$
\begin{aligned}
p(y_n = 1|\boldsymbol{x}_n, \boldsymbol{w}) &= \mu_n \\
p(y_n = 0|\boldsymbol{x}_n, \boldsymbol{w}) &= 1 - \mu_n
\end{aligned}
$$

- The model's prediction is a probability defined using the sigmoid function

$$
f(\boldsymbol{x}_n) = \mu_n = \sigma(\mathbf{w}^\top \mathbf{x}_n) = \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x}_n)} = \frac{\exp(\mathbf{w}^\top \mathbf{x}_n)}{1 + \exp(\mathbf{w}^\top \mathbf{x}_n)}
$$

- The sigmoid first computes a real-valued "score" $\boldsymbol{w}^\top \boldsymbol{x} = \sum_{d=1}^{D} w_d x_d$ and "squashes" it between (0,1) to turn this score into a probability score



Logistic Regression: The Model

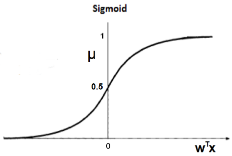
- A model for doing *probabilistic* binary classification
- Predicts *label probabilities* rather than a hard value of the label

$$
\begin{aligned}
p(y_n = 1|\boldsymbol{x}_n, \boldsymbol{w}) &= \mu_n \\
p(y_n = 0|\boldsymbol{x}_n, \boldsymbol{w}) &= 1 - \mu_n
\end{aligned}
$$

- The model's prediction is a probability defined using the sigmoid function

$$
f(\boldsymbol{x}_n) = \mu_n = \sigma(\mathbf{w}^\top \mathbf{x}_n) = \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x}_n)} = \frac{\exp(\mathbf{w}^\top \mathbf{x}_n)}{1 + \exp(\mathbf{w}^\top \mathbf{x}_n)}
$$

- The sigmoid first computes a real-valued "score" $\boldsymbol{w}^\top \boldsymbol{x} = \sum_{d=1}^{D} w_d x_d$ and "squashes" it between (0,1) to turn this score into a probability score



- Model parameter is the unknown $\boldsymbol{w}$. Need to learn it from training data.

Logistic Regression: An Interpretion

- Recall that the logistic regression model defines

$$
\begin{aligned}
p(y=1|\boldsymbol{x},\boldsymbol{w}) &= \mu = \sigma(\boldsymbol{w}^\top \boldsymbol{x}) = \frac{1}{1+\exp(-\boldsymbol{w}^\top \boldsymbol{x})} = \frac{\exp(\boldsymbol{w}^\top \boldsymbol{x})}{1+\exp(\boldsymbol{w}^\top \boldsymbol{x})} \\
p(y=0|\boldsymbol{x},\boldsymbol{w}) &= 1-\mu = 1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}) = \frac{1}{1+\exp(\boldsymbol{w}^\top \boldsymbol{x})}
\end{aligned}
$$

Logistic Regression: An Interpretion

- Recall that the logistic regression model defines

$$
\begin{aligned}
p(y=1|\boldsymbol{x},\boldsymbol{w}) &= \mu = \sigma(\boldsymbol{w}^\top \boldsymbol{x}) = \frac{1}{1+\exp(-\boldsymbol{w}^\top \boldsymbol{x})} = \frac{\exp(\boldsymbol{w}^\top \boldsymbol{x})}{1+\exp(\boldsymbol{w}^\top \boldsymbol{x})} \\
p(y=0|\boldsymbol{x},\boldsymbol{w}) &= 1-\mu = 1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}) = \frac{1}{1+\exp(\boldsymbol{w}^\top \boldsymbol{x})}
\end{aligned}
$$

- The log-odds of this model

$$
\log \frac{p(y=1|\boldsymbol{x},\boldsymbol{w})}{p(y=0|\boldsymbol{x},\boldsymbol{w})} = \log \exp(\boldsymbol{w}^\top \boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x}
$$

Logistic Regression: An Interpretion

- Recall that the logistic regression model defines

$$
\begin{aligned}
p(y=1|\boldsymbol{x},\boldsymbol{w}) &= \mu = \sigma(\boldsymbol{w}^\top \boldsymbol{x}) = \frac{1}{1+\exp(-\boldsymbol{w}^\top \boldsymbol{x})} = \frac{\exp(\boldsymbol{w}^\top \boldsymbol{x})}{1+\exp(\boldsymbol{w}^\top \boldsymbol{x})} \\
p(y=0|\boldsymbol{x},\boldsymbol{w}) &= 1-\mu = 1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}) = \frac{1}{1+\exp(\boldsymbol{w}^\top \boldsymbol{x})}
\end{aligned}
$$

- The log-odds of this model

$$
\log \frac{p(y=1|\boldsymbol{x},\boldsymbol{w})}{p(y=0|\boldsymbol{x},\boldsymbol{w})} = \log \exp(\boldsymbol{w}^\top \boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x}
$$

- Thus if $\boldsymbol{w}^\top \boldsymbol{x} > 0$ then the positive class is more probable

Logistic Regression: An Interpretion

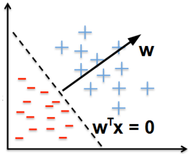
- Recall that the logistic regression model defines

$$
\begin{aligned}
p(y=1|\boldsymbol{x},\boldsymbol{w}) &= \mu = \sigma(\boldsymbol{w}^\top\boldsymbol{x}) = \frac{1}{1+\exp(-\boldsymbol{w}^\top\boldsymbol{x})} = \frac{\exp(\boldsymbol{w}^\top\boldsymbol{x})}{1+\exp(\boldsymbol{w}^\top\boldsymbol{x})} \\
p(y=0|\boldsymbol{x},\boldsymbol{w}) &= 1-\mu = 1-\sigma(\boldsymbol{w}^\top\boldsymbol{x}) = \frac{1}{1+\exp(\boldsymbol{w}^\top\boldsymbol{x})}
\end{aligned}
$$

- The log-odds of this model

$$
\log\frac{p(y=1|\boldsymbol{x},\boldsymbol{w})}{p(y=0|\boldsymbol{x},\boldsymbol{w})} = \log\exp(\boldsymbol{w}^\top\boldsymbol{x}) = \boldsymbol{w}^\top\boldsymbol{x}
$$

- Thus if $\boldsymbol{w}^\top\boldsymbol{x} > 0$ then the positive class is more probable
- A linear classification model. Separates the two classes via a hyperplane (similar to other linear classification models such as Perceptron and SVM)



Loss Function Optimization View for Logistic Regression

Logistic Regression: The Loss Function

- What loss function to use? One option is to use the squared loss

$$\ell(y_n, f(\boldsymbol{x}_n)) = (y_n - f(\boldsymbol{x}_n))^2 = (y_n - \mu_n)^2 = (y_n - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n))^2$$

Logistic Regression: The Loss Function

- What loss function to use? One option is to use the squared loss

$$\ell(y_n, f(\boldsymbol{x}_n)) = (y_n - f(\boldsymbol{x}_n))^2 = (y_n - \mu_n)^2 = (y_n - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n))^2$$

- This is non-convex and not easy to optimize

Logistic Regression: The Loss Function

- What loss function to use? One option is to use the squared loss

$$\ell(y_n, f(\boldsymbol{x}_n)) = (y_n - f(\boldsymbol{x}_n))^2 = (y_n - \mu_n)^2 = (y_n - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n))^2$$

- This is non-convex and not easy to optimize
- Consider the following loss function

$$\ell(y_n, f(\boldsymbol{x}_n)) = \begin{cases} -\log(\mu_n) & y_n = 1 \\ -\log(1 - \mu_n) & y_n = 0 \end{cases}$$

Logistic Regression: The Loss Function

- What loss function to use? One option is to use the squared loss

$$\ell(y_n, f(\boldsymbol{x}_n)) = (y_n - f(\boldsymbol{x}_n))^2 = (y_n - \mu_n)^2 = (y_n - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n))^2$$

- This is non-convex and not easy to optimize
- Consider the following loss function

$$\ell(y_n, f(\boldsymbol{x}_n)) = \begin{cases} -\log(\mu_n) & y_n = 1 \\ -\log(1 - \mu_n) & y_n = 0 \end{cases}$$

- This loss function makes intuitive sense

Logistic Regression: The Loss Function

- What loss function to use? One option is to use the squared loss

$$\ell(y_n, f(\boldsymbol{x}_n)) = (y_n - f(\boldsymbol{x}_n))^2 = (y_n - \mu_n)^2 = (y_n - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n))^2$$

- This is non-convex and not easy to optimize
- Consider the following loss function

$$\ell(y_n, f(\boldsymbol{x}_n)) = \begin{cases} -\log(\mu_n) & y_n = 1 \\ -\log(1 - \mu_n) & y_n = 0 \end{cases}$$

- This loss function makes intuitive sense
  - If $y_n = 1$ but μ_n is close to 0 (model makes error) then loss will be high

Logistic Regression: The Loss Function

- What loss function to use? One option is to use the squared loss

$$\ell(y_n, f(\boldsymbol{x}_n)) = (y_n - f(\boldsymbol{x}_n))^2 = (y_n - \mu_n)^2 = (y_n - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n))^2$$

- This is non-convex and not easy to optimize
- Consider the following loss function

$$\ell(y_n, f(\boldsymbol{x}_n)) = \begin{cases} -\log(\mu_n) & y_n = 1 \\ -\log(1 - \mu_n) & y_n = 0 \end{cases}$$

- This loss function makes intuitive sense
  - If $y_n = 1$ but μ_n is close to 0 (model makes error) then loss will be high
  - If $y_n = 0$ but μ_n is close to 1 (model makes error) then loss will be high

Logistic Regression: The Loss Function

- What loss function to use? One option is to use the squared loss

$$\ell(y_n, f(\boldsymbol{x}_n)) = (y_n - f(\boldsymbol{x}_n))^2 = (y_n - \mu_n)^2 = (y_n - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n))^2$$

- This is non-convex and not easy to optimize
- Consider the following loss function

$$\ell(y_n, f(\boldsymbol{x}_n)) = \begin{cases} -\log(\mu_n) & y_n = 1 \\ -\log(1 - \mu_n) & y_n = 0 \end{cases}$$

- This loss function makes intuitive sense
  - If $y_n = 1$ but μ_n is close to 0 (model makes error) then loss will be high
  - If $y_n = 0$ but μ_n is close to 1 (model makes error) then loss will be high
- The above loss function can be combined and written more compactly as

$$\boxed{\ell(y_n, f(\boldsymbol{x}_n)) = -y_n \log(\mu_n) - (1 - y_n)\log(1 - \mu_n)}$$

Logistic Regression: The Loss Function

- What loss function to use? One option is to use the squared loss

$$\ell(y_n, f(\boldsymbol{x}_n)) = (y_n - f(\boldsymbol{x}_n))^2 = (y_n - \mu_n)^2 = (y_n - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n))^2$$

- This is non-convex and not easy to optimize
- Consider the following loss function

$$\ell(y_n, f(\boldsymbol{x}_n)) = \begin{cases} -\log(\mu_n) & y_n = 1 \\ -\log(1-\mu_n) & y_n = 0 \end{cases}$$

- This loss function makes intuitive sense
  - If $y_n = 1$ but μ_n is close to 0 (model makes error) then loss will be high
  - If $y_n = 0$ but μ_n is close to 1 (model makes error) then loss will be high
- The above loss function can be combined and written more compactly as

$$\boxed{\ell(y_n, f(\boldsymbol{x}_n)) = -y_n \log(\mu_n) - (1-y_n)\log(1-\mu_n)}$$

- This is a function of the unknown parameter $\boldsymbol{w}$ since $\mu_n = \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)$

Logistic Regression: The Loss Function

- The loss function over the entire training data

$$L(\boldsymbol{w}) = \sum_{n=1}^{N} \ell(y_n, f(\boldsymbol{x}_n)) = \sum_{n=1}^{N} [-y_n \log(\mu_n) - (1 - y_n) \log(1 - \mu_n)]$$

Logistic Regression: The Loss Function

- The loss function over the entire training data

$$L(\boldsymbol{w}) = \sum_{n=1}^{N} \ell(y_n, f(\boldsymbol{x}_n)) = \sum_{n=1}^{N} [-y_n \log(\mu_n) - (1 - y_n) \log(1 - \mu_n)]$$

- This is also known as the cross-entropy loss
  - Sum of the cross-entropies b/w true label y_n and predicted label prob. μ_n

Logistic Regression: The Loss Function

- The loss function over the entire training data

$$L(\boldsymbol{w}) = \sum_{n=1}^{N} \ell(y_n, f(\boldsymbol{x}_n)) = \sum_{n=1}^{N} [-y_n \log(\mu_n) - (1 - y_n) \log(1 - \mu_n)]$$

- This is also known as the cross-entropy loss
  - Sum of the cross-entropies b/w true label y_n and predicted label prob. μ_n
- Plugging in $\mu_n = \frac{\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}{1+\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}$ and chugging, we get (verify yourself)

$$\boxed{L(\boldsymbol{w}) = -\sum_{n=1}^{N} (y_n \boldsymbol{w}^\top \boldsymbol{x}_n - \log(1 + \exp(\boldsymbol{w}^\top \boldsymbol{x}_n)))}$$

Logistic Regression: The Loss Function

- The loss function over the entire training data

$$L(\boldsymbol{w}) = \sum_{n=1}^{N} \ell(y_n, f(\boldsymbol{x}_n)) = \sum_{n=1}^{N} [-y_n \log(\mu_n) - (1-y_n)\log(1-\mu_n)]$$

- This is also known as the cross-entropy loss
  - Sum of the cross-entropies b/w true label y_n and predicted label prob. μ_n
- Plugging in $\mu_n = \frac{\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}{1+\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}$ and chugging, we get (verify yourself)

$$\boxed{L(\boldsymbol{w}) = -\sum_{n=1}^{N} (y_n \boldsymbol{w}^\top \boldsymbol{x}_n - \log(1+\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)))}$$

- We can add a regularizer (e.g., squared ℓ_2 norm of $\boldsymbol{w}$) to prevent overfitting

$$\boxed{L(\boldsymbol{w}) = -\sum_{n=1}^{N} (y_n \boldsymbol{w}^\top \boldsymbol{x}_n - \log(1+\exp(\boldsymbol{w}^\top \boldsymbol{x}_n))) + \lambda||\boldsymbol{w}||^2}$$

Probabilstic Modeling View (MLE/MAP) for Logistic Regression

Logistic Regression: MLE Formulation

- Recall, each label y_n is binary with prob. μ_n. Assume Bernoulli likelihood:

$$p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) = \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) = \prod_{n=1}^{N} \mu_n^{y_n}(1-\mu_n)^{1-y_n}$$

where $\mu_n = \frac{\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}{1+\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}$

Logistic Regression: MLE Formulation

- Recall, each label y_n is binary with prob. μ_n. Assume Bernoulli likelihood:

$$p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) = \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) = \prod_{n=1}^{N} \mu_n^{y_n}(1-\mu_n)^{1-y_n}$$

  where $\mu_n = \frac{\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}{1+\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}$

- Doing MLE would require maximizing the log likelihood w.r.t. $\boldsymbol{w}$

$$\log p(\mathbf{Y}|\mathbf{X}, \boldsymbol{w}) = \sum_{n=1}^{N} (y_n \log \mu_n + (1-y_n)\log(1-\mu_n))$$

Logistic Regression: MLE Formulation

- Recall, each label y_n is binary with prob. μ_n. Assume Bernoulli likelihood:

$$p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) = \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) = \prod_{n=1}^{N} \mu_n^{y_n}(1-\mu_n)^{1-y_n}$$

where $\mu_n = \frac{\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}{1+\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}$

- Doing MLE would require maximizing the log likelihood w.r.t. $\boldsymbol{w}$

$$\log p(\mathbf{Y}|\mathbf{X}, \boldsymbol{w}) = \sum_{n=1}^{N}(y_n \log \mu_n + (1-y_n)\log(1-\mu_n))$$

- This is equivalent to minimizing the NLL. Plugging in $\mu_n = \frac{\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}{1+\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}$ we get

$$\boxed{\text{NLL}(\boldsymbol{w}) = -\sum_{n=1}^{N}(y_n \boldsymbol{w}^\top \boldsymbol{x}_n - \log(1+\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)))}$$

Logistic Regression: MLE Formulation

- Recall, each label y_n is binary with prob. μ_n. Assume Bernoulli likelihood:

$$p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) = \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) = \prod_{n=1}^{N} \mu_n^{y_n}(1-\mu_n)^{1-y_n}$$

where $\mu_n = \frac{\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}{1+\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}$

- Doing MLE would require maximizing the log likelihood w.r.t. $\boldsymbol{w}$

$$\log p(\mathbf{Y}|\mathbf{X}, \boldsymbol{w}) = \sum_{n=1}^{N}(y_n \log \mu_n + (1-y_n)\log(1-\mu_n))$$

- This is equivalent to minimizing the NLL. Plugging in $\mu_n = \frac{\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}{1+\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}$ we get

$$\boxed{\text{NLL}(\boldsymbol{w}) = -\sum_{n=1}^{N}(y_n \boldsymbol{w}^\top \boldsymbol{x}_n - \log(1+\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)))}$$

- Not surprisingly, the NLL expression is the same as the loss function

Logisic Regression: MAP Formulation

- MLE estimate of $\boldsymbol{w}$ can lead to overfitting. Solution: use a prior on $\boldsymbol{w}$
- Just like the linear regression case, let's put a Gausian prior on $\boldsymbol{w}$

$$p(\boldsymbol{w}) = \mathcal{N}(0, \lambda^{-1}\mathbf{I}_D) \propto \exp(-\lambda \boldsymbol{w}^\top \boldsymbol{w})$$

Logisic Regression: MAP Formulation

- MLE estimate of $\boldsymbol{w}$ can lead to overfitting. Solution: use a prior on $\boldsymbol{w}$
- Just like the linear regression case, let's put a Gausian prior on $\boldsymbol{w}$

$$p(\boldsymbol{w}) = \mathcal{N}(0, \lambda^{-1}\mathbf{I}_D) \propto \exp(-\lambda \boldsymbol{w}^\top \boldsymbol{w})$$

- MAP objective: MLE objective + $\log p(\boldsymbol{w})$

Logisic Regression: MAP Formulation

- MLE estimate of $\boldsymbol{w}$ can lead to overfitting. Solution: use a prior on $\boldsymbol{w}$
- Just like the linear regression case, let's put a Gausian prior on $\boldsymbol{w}$

$$p(\boldsymbol{w}) = \mathcal{N}(0, \lambda^{-1}\mathbf{I}_D) \propto \exp(-\lambda \boldsymbol{w}^\top \boldsymbol{w})$$

- MAP objective: MLE objective $+ \log p(\boldsymbol{w})$
- Can maximize the MAP objective (log-posterior) w.r.t. $\boldsymbol{w}$ or minimize the negative of log posterior which will be

$$\text{NLL}(\boldsymbol{w}) - \log p(\boldsymbol{w})$$

Logisic Regression: MAP Formulation

- MLE estimate of $\boldsymbol{w}$ can lead to overfitting. Solution: use a prior on $\boldsymbol{w}$
- Just like the linear regression case, let's put a Gausian prior on $\boldsymbol{w}$

$$p(\boldsymbol{w}) = \mathcal{N}(0, \lambda^{-1}\mathbf{I}_D) \propto \exp(-\lambda \boldsymbol{w}^\top \boldsymbol{w})$$

- MAP objective: MLE objective + $\log p(\boldsymbol{w})$
- Can maximize the MAP objective (log-posterior) w.r.t. $\boldsymbol{w}$ or minimize the negative of log posterior which will be

$$\text{NLL}(\boldsymbol{w}) - \log p(\boldsymbol{w})$$

- Ignoring the constants, we get the following objective for MAP estimation

$$-\sum_{n=1}^{N}(y_n \boldsymbol{w}^\top \boldsymbol{x}_n - \log(1 + \exp(\boldsymbol{w}^\top \boldsymbol{x}_n))) + \lambda \boldsymbol{w}^\top \boldsymbol{w}$$

Logisic Regression: MAP Formulation

- MLE estimate of $\boldsymbol{w}$ can lead to overfitting. Solution: use a prior on $\boldsymbol{w}$
- Just like the linear regression case, let's put a Gausian prior on $\boldsymbol{w}$

$$p(\boldsymbol{w}) = \mathcal{N}(0, \lambda^{-1}\mathbf{I}_D) \propto \exp(-\lambda \boldsymbol{w}^\top \boldsymbol{w})$$

- MAP objective: MLE objective + $\log p(\boldsymbol{w})$
- Can maximize the MAP objective (log-posterior) w.r.t. $\boldsymbol{w}$ or minimize the negative of log posterior which will be

$$\text{NLL}(\boldsymbol{w}) - \log p(\boldsymbol{w})$$

- Ignoring the constants, we get the following objective for MAP estimation

$$-\sum_{n=1}^{N}(y_n \boldsymbol{w}^\top \boldsymbol{x}_n - \log(1 + \exp(\boldsymbol{w}^\top \boldsymbol{x}_n))) + \lambda \boldsymbol{w}^\top \boldsymbol{w}$$

- Thus MAP estimation is equivalent to regularized logistic regression

Estimating the Weight Vector $\boldsymbol{w}$

- Loss function/NLL for logistic regression (ignoring the regularizer term)

$$L(\boldsymbol{w}) = -\sum_{n=1}^{N}(y_n \boldsymbol{w}^\top \boldsymbol{x}_n - \log(1 + \exp(\boldsymbol{w}^\top \boldsymbol{x}_n)))$$

- The loss function is convex in $\boldsymbol{w}$ (thus has a unique minimum)

Estimating the Weight Vector $\boldsymbol{w}$

- Loss function/NLL for logistic regression (ignoring the regularizer term)

$$L(\boldsymbol{w}) = -\sum_{n=1}^{N}(y_n \boldsymbol{w}^\top \boldsymbol{x}_n - \log(1+\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)))$$

- The loss function is convex in $\boldsymbol{w}$ (thus has a unique minimum)
- The gradient/derivative of $L(\boldsymbol{w})$ w.r.t. $\boldsymbol{w}$ (let's ignore the regularizer)

$$\mathbf{g} = \frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} \quad = \quad \frac{\partial}{\partial \boldsymbol{w}}[-\sum_{n=1}^{N}(y_n \boldsymbol{w}^\top \boldsymbol{x}_n - \log(1+\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)))]$$

Estimating the Weight Vector $\boldsymbol{w}$

- Loss function/NLL for logistic regression (ignoring the regularizer term)

$$L(\boldsymbol{w}) = -\sum_{n=1}^{N}(y_n \boldsymbol{w}^\top \boldsymbol{x}_n - \log(1 + \exp(\boldsymbol{w}^\top \boldsymbol{x}_n)))$$

- The loss function is convex in $\boldsymbol{w}$ (thus has a unique minimum)
- The gradient/derivative of $L(\boldsymbol{w})$ w.r.t. $\boldsymbol{w}$ (let's ignore the regularizer)

$$\begin{aligned}\mathbf{g} = \frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} &= \frac{\partial}{\partial \boldsymbol{w}}[-\sum_{n=1}^{N}(y_n \boldsymbol{w}^\top \boldsymbol{x}_n - \log(1 + \exp(\boldsymbol{w}^\top \boldsymbol{x}_n)))] \\ &= -\sum_{n=1}^{N}\left(y_n \boldsymbol{x}_n - \frac{\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}{(1 + \exp(\boldsymbol{w}^\top \boldsymbol{x}_n))}\boldsymbol{x}_n\right)\end{aligned}$$

Estimating the Weight Vector $\boldsymbol{w}$

- Loss function/NLL for logistic regression (ignoring the regularizer term)

$$L(\boldsymbol{w}) = -\sum_{n=1}^{N}(y_n \boldsymbol{w}^\top \boldsymbol{x}_n - \log(1 + \exp(\boldsymbol{w}^\top \boldsymbol{x}_n)))$$

- The loss function is convex in $\boldsymbol{w}$ (thus has a unique minimum)
- The gradient/derivative of $L(\boldsymbol{w})$ w.r.t. $\boldsymbol{w}$ (let's ignore the regularizer)

$$\begin{aligned}
\mathbf{g} = \frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} &= \frac{\partial}{\partial \boldsymbol{w}}[-\sum_{n=1}^{N}(y_n \boldsymbol{w}^\top \boldsymbol{x}_n - \log(1 + \exp(\boldsymbol{w}^\top \boldsymbol{x}_n)))] \\
&= -\sum_{n=1}^{N}\left(y_n \boldsymbol{x}_n - \frac{\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}{(1 + \exp(\boldsymbol{w}^\top \boldsymbol{x}_n))}\boldsymbol{x}_n\right) \\
&= -\sum_{n=1}^{N}(y_n - \mu_n)\boldsymbol{x}_n = \mathbf{X}^\top(\boldsymbol{\mu} - \boldsymbol{y})
\end{aligned}$$

Estimating the Weight Vector $\boldsymbol{w}$

- Loss function/NLL for logistic regression (ignoring the regularizer term)

$$L(\boldsymbol{w}) = -\sum_{n=1}^{N}(y_n \boldsymbol{w}^\top \boldsymbol{x}_n - \log(1 + \exp(\boldsymbol{w}^\top \boldsymbol{x}_n)))$$

- The loss function is convex in $\boldsymbol{w}$ (thus has a unique minimum)
- The gradient/derivative of $L(\boldsymbol{w})$ w.r.t. $\boldsymbol{w}$ (let's ignore the regularizer)

$$\begin{aligned}
\mathbf{g} = \frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} &= \frac{\partial}{\partial \boldsymbol{w}}[-\sum_{n=1}^{N}(y_n \boldsymbol{w}^\top \boldsymbol{x}_n - \log(1 + \exp(\boldsymbol{w}^\top \boldsymbol{x}_n)))] \\
&= -\sum_{n=1}^{N}\left(y_n \boldsymbol{x}_n - \frac{\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}{(1 + \exp(\boldsymbol{w}^\top \boldsymbol{x}_n))}\boldsymbol{x}_n\right) \\
&= -\sum_{n=1}^{N}(y_n - \mu_n)\boldsymbol{x}_n = \mathbf{X}^\top(\boldsymbol{\mu} - \boldsymbol{y})
\end{aligned}$$

- Can't get a closed form solution for $\boldsymbol{w}$ by setting the derivative to zero
  - Need to use iterative methods (e.g., gradient descent) to solve for $\boldsymbol{w}$

Gradient Descent for Logistic Regression

- We can use gradient descent (GD) to solve for $\boldsymbol{w}$ as follows:

Gradient Descent for Logistic Regression

- We can use gradient descent (GD) to solve for $\boldsymbol{w}$ as follows:
    - Initialize $\boldsymbol{w}^{(1)} \in \mathbb{R}^D$ randomly.

Gradient Descent for Logistic Regression

- We can use gradient descent (GD) to solve for $\boldsymbol{w}$ as follows:
  - Initialize $\boldsymbol{w}^{(1)} \in \mathbb{R}^D$ randomly.
  - Iterate the following until convergence

$$\underbrace{\boldsymbol{w}^{(t+1)}}_{\text{new value}} = \underbrace{\boldsymbol{w}^{(t)}}_{\text{previous value}} - \eta \underbrace{\sum_{n=1}^{N}(\mu_n^{(t)} - y_n)\boldsymbol{x}_n}_{\text{gradient at previous value}}$$

where η is the learning rate and $\mu^{(t)} = \sigma(\boldsymbol{w}^{(t)^\top}\boldsymbol{x}_n)$ is the predicted label probability for $\boldsymbol{x}_n$ using $\boldsymbol{w} = \boldsymbol{w}^{(t)}$ from the previous iteration

Gradient Descent for Logistic Regression

- We can use gradient descent (GD) to solve for $\boldsymbol{w}$ as follows:
  - Initialize $\boldsymbol{w}^{(1)} \in \mathbb{R}^D$ randomly.
  - Iterate the following until convergence

$$\underbrace{\boldsymbol{w}^{(t+1)}}_{\text{new value}} = \underbrace{\boldsymbol{w}^{(t)}}_{\text{previous value}} - \eta \underbrace{\sum_{n=1}^{N}(\mu_n^{(t)} - y_n)\boldsymbol{x}_n}_{\text{gradient at previous value}}$$

  where η is the learning rate and $\mu^{(t)} = \sigma(\boldsymbol{w}^{(t)\top}\boldsymbol{x}_n)$ is the predicted label probability for $\boldsymbol{x}_n$ using $\boldsymbol{w} = \boldsymbol{w}^{(t)}$ from the previous iteration

- Note that the updates give larger weights to those examples on which the current model makes larger mistakes, as measured by $(\mu_n^{(t)} - y_n)$

Gradient Descent for Logistic Regression

- We can use gradient descent (GD) to solve for $\boldsymbol{w}$ as follows:
  - Initialize $\boldsymbol{w}^{(1)} \in \mathbb{R}^D$ randomly.
  - Iterate the following until convergence

$$\underbrace{\boldsymbol{w}^{(t+1)}}_{\text{new value}} = \underbrace{\boldsymbol{w}^{(t)}}_{\text{previous value}} - \eta \underbrace{\sum_{n=1}^{N}(\mu_n^{(t)} - y_n)\boldsymbol{x}_n}_{\text{gradient at previous value}}$$

  where η is the learning rate and $\mu^{(t)} = \sigma(\boldsymbol{w}^{(t)^\top}\boldsymbol{x}_n)$ is the predicted label probability for $\boldsymbol{x}_n$ using $\boldsymbol{w} = \boldsymbol{w}^{(t)}$ from the previous iteration

- Note that the updates give larger weights to those examples on which the current model makes larger mistakes, as measured by $(\mu_n^{(t)} - y_n)$
- **Note:** Computing the gradient in every iteration requires all the data. Thus GD can be expensive if N is very large. A cheaper alternative is to do GD using only a small randomly chosen minibatch of data. It is known as **Stochastic Gradient Descent** (SGD). Runs faster and converges faster.

More on Gradient Descent..

- GD can converge slowly and is also sensitive to the step size

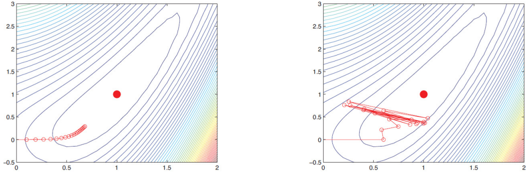


Figure: Left: small step sizes. Right: large step sizes

[1] Also see: "A comparison of numerical optimizers for logistic regression" by Tom Minka

More on Gradient Descent..

- GD can converge slowly and is also sensitive to the step size

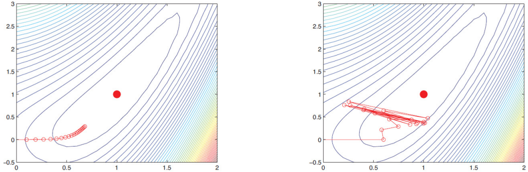


Figure: Left: small step sizes. Right: large step sizes

- Several ways to remedy this[1].

[1] Also see: "A comparison of numerical optimizers for logistic regression" by Tom Minka

More on Gradient Descent..

- GD can converge slowly and is also sensitive to the step size

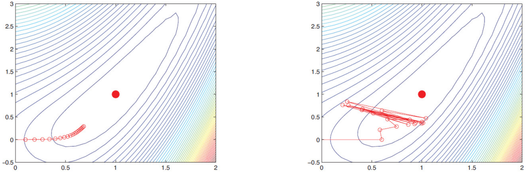


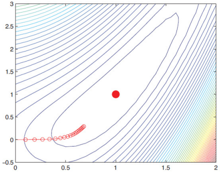
Figure: Left: small step sizes. Right: large step sizes

- Several ways to remedy this[1]. E.g.,

[1] Also see: "A comparison of numerical optimizers for logistic regression" by Tom Minka

More on Gradient Descent..

- GD can converge slowly and is also sensitive to the step size



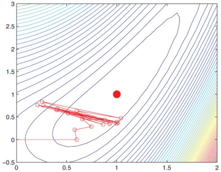


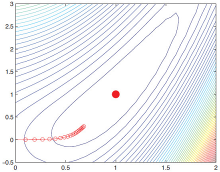
Figure: Left: small step sizes. Right: large step sizes

- Several ways to remedy this[1]. E.g.,
  - Choose the optimal step size η_t (different in each iteration) by line-search

[1] Also see: "A comparison of numerical optimizers for logistic regression" by Tom Minka

More on Gradient Descent..

- GD can converge slowly and is also sensitive to the step size



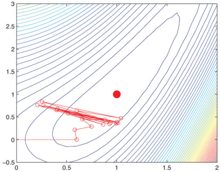


Figure: Left: small step sizes. Right: large step sizes

- Several ways to remedy this[1]. E.g.,
  - Choose the optimal step size η_t (different in each iteration) by line-search
  - Add a momentum term to the updates

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta_t \mathbf{g}^{(t)} + \alpha_t(\boldsymbol{w}^{(t)} - \boldsymbol{w}^{(t-1)})$$

[1] Also see: "A comparison of numerical optimizers for logistic regression" by Tom Minka

Multiclass Logistic (or "Softmax") Regression

- Logistic regression can be extended to handle $K > 2$ classes
- In this case, $y_n \in \{0, 1, 2, \ldots, K-1\}$ and label probabilities are defined as

$$p(y_n = k | \boldsymbol{x}_n, \mathbf{W}) = \frac{\exp(\boldsymbol{w}_k^\top \boldsymbol{x}_n)}{\sum_{\ell=1}^{K} \exp(\boldsymbol{w}_\ell^\top \boldsymbol{x}_n)} = \mu_{nk}$$

- μ_{nk}: probability that example n belongs to class k. Also, $\sum_{\ell=1}^{K} \mu_{n\ell} = 1$
- $\mathbf{W} = [\boldsymbol{w}_1 \ \boldsymbol{w}_2 \ \ldots \ \boldsymbol{w}_K]$ is $D \times K$ weight matrix (column k for class k)

Multiclass Logistic (or "Softmax") Regression

- Logistic regression can be extended to handle $K > 2$ classes
- In this case, $y_n \in \{0, 1, 2, \ldots, K-1\}$ and label probabilities are defined as

$$p(y_n = k | \boldsymbol{x}_n, \mathbf{W}) = \frac{\exp(\boldsymbol{w}_k^\top \boldsymbol{x}_n)}{\sum_{\ell=1}^{K} \exp(\boldsymbol{w}_\ell^\top \boldsymbol{x}_n)} = \mu_{nk}$$

- μ_{nk}: probability that example n belongs to class k. Also, $\sum_{\ell=1}^{K} \mu_{n\ell} = 1$
- $\mathbf{W} = [\boldsymbol{w}_1 \ \boldsymbol{w}_2 \ \ldots \ \boldsymbol{w}_K]$ is $D \times K$ weight matrix (column k for class k)
- "Softmax" because class 'k' with largest $\boldsymbol{w}_k^\top \boldsymbol{x}_n$ dominates the probability

Multiclass Logistic (or "Softmax") Regression

- Logistic regression can be extended to handle $K > 2$ classes
- In this case, $y_n \in \{0, 1, 2, \ldots, K-1\}$ and label probabilities are defined as

$$p(y_n = k|\boldsymbol{x}_n, \mathbf{W}) = \frac{\exp(\boldsymbol{w}_k^\top \boldsymbol{x}_n)}{\sum_{\ell=1}^{K} \exp(\boldsymbol{w}_\ell^\top \boldsymbol{x}_n)} = \mu_{nk}$$

- μ_{nk}: probability that example n belongs to class k. Also, $\sum_{\ell=1}^{K} \mu_{n\ell} = 1$
- $\mathbf{W} = [\boldsymbol{w}_1 \ \boldsymbol{w}_2 \ \ldots \ \boldsymbol{w}_K]$ is $D \times K$ weight matrix (column k for class k)
- "Softmax" because class 'k' with largest $\boldsymbol{w}_k^\top \boldsymbol{x}_n$ dominates the probability
- We can think of the y_n's as drawn from a multinomial distribution

$$p(\boldsymbol{y}|\mathbf{X}, \mathbf{W}) = \prod_{n=1}^{N} \prod_{\ell=1}^{K} \mu_{n\ell}^{y_{n\ell}} \qquad \text{(Likelihood function)}$$

where $y_{n\ell} = 1$ if true class of example n is ℓ and $y_{n\ell'} = 0$ for all other $\ell' \neq \ell$

Multiclass Logistic (or "Softmax") Regression

- Logistic regression can be extended to handle $K > 2$ classes
- In this case, $y_n \in \{0, 1, 2, \ldots, K-1\}$ and label probabilities are defined as

$$p(y_n = k|\boldsymbol{x}_n, \mathbf{W}) = \frac{\exp(\boldsymbol{w}_k^\top \boldsymbol{x}_n)}{\sum_{\ell=1}^K \exp(\boldsymbol{w}_\ell^\top \boldsymbol{x}_n)} = \mu_{nk}$$

- μ_{nk}: probability that example n belongs to class k. Also, $\sum_{\ell=1}^K \mu_{n\ell} = 1$
- $\mathbf{W} = [\boldsymbol{w}_1 \ \boldsymbol{w}_2 \ \ldots \ \boldsymbol{w}_K]$ is $D \times K$ weight matrix (column k for class k)
- "Softmax" because class 'k' with largest $\boldsymbol{w}_k^\top \boldsymbol{x}_n$ dominates the probability
- We can think of the y_n's as drawn from a multinomial distribution

$$p(\boldsymbol{y}|\mathbf{X}, \mathbf{W}) = \prod_{n=1}^N \prod_{\ell=1}^K \mu_{n\ell}^{y_{n\ell}} \qquad \text{(Likelihood function)}$$

  where $y_{n\ell} = 1$ if true class of example n is ℓ and $y_{n\ell'} = 0$ for all other $\ell' \neq \ell$

- Can do MLE/MAP for $\mathbf{W}$ similar to the binary logistic regression case

Logistic Regression: Summary

- A probabilistic model for binary classification
- Simple objective, easy to optimize using gradient based methods
- Very widely used, very efficient solvers exist
- Can be extended for multiclass (softmax) classification
- Used as modules in more complex models (e.g, deep neural nets)