CS 559 Machine Learning

Logistic Regression

Yue Ning
Department of Computer Science
Stevens Institute of Technology

Plan for today

Probabilistic Generative Models

Logistic Regression

Review of last lecture

- Bayesian Decision Theory
- Linear classifiers:
  1. Least square classification,
  2. Fisher's linear discriminant (Geometrical properties of Linear Discriminant Analysis)
  3. Perceptron

- Three important elements: data$(\mathbf{x}, y)$, loss function, model parameters $\mathbf{w}$
- One important line, gradient update:

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \eta \nabla \mathbf{w}$$

Gradient Descent

```
procedure BATCH GRADIENT DESCENT
    for i in range(epochs) do
        g^(i)(w) = evaluate_gradient(TrainLoss, data, w)
        w = w − learning_rate ∗ g^(i)(w)
    end for
end procedure
```

- Can be very slow.
- Intractable for datasets that don't fit in memory.
- Doesn't allow us to update our model online, i.e. with new examples on-the-fly.
- Guaranteed to converge to the global minimum for convex error surfaces and to a local minimum for non-convex surfaces.

Stochastic Gradient Descent

procedure STOCHASTIC GRADIENT DESCENT
    for i in range(epochs) **do**
        np.random.shuffle(data)
        for example $\in$ data **do**
            $g^{(i)}(\mathbf{w}) =$ evaluate_gradient(loss, example, $\mathbf{w}$)
            $\mathbf{w} = \mathbf{w} -$ learning_rate $* \, g^{(i)}(\mathbf{w})$
        end for
    end for
end procedure

- Allow for online update with new examples.
- With a high variance that cause the objective function to fluctuate heavily.

Mini-batch Gradient Descent

```
procedure MINI-BATCH GRADIENT DESCENT
    for i in range(epochs) do
        np.random.shuffle(data)
        for batch ∈ get_batches(data, batch_size=50) do
            g^(i)(w) = evaluate_gradient(loss, batch, w)
            w = w − learning_rate ∗ g^(i)(w)
        end for
    end for
end procedure
```

- Reduces the variance of the parameter updates, which can lead to more stable convergence;
- Can make use of highly optimized matrix optimizations common to state-of-the-art deep learning libraries that make computing the gradient w.r.t. a mini-batch very efficient.

Evaluation Metrics for Binary Classification

- The contingency table or confusion matrix:

		True value	
		Positive	Negative
Predicted	Positive	True positive (TP)	False positive (FP)
	Negative	False negative (FN)	True negative (TN)

- Recall $= \frac{tp}{tp+fn}$
- Precision $= \frac{tp}{tp+fp}$
- Accuracy $= \frac{tp+tn}{tp+tn+fp+fn}$ (not a good metric for imbalanced dataset)

Part I: Probabilistic Generative Models

Probabilistic Generative Models

- We now turn to a probabilistic approach to classification.
- How models with linear decision boundaries arise from simple assumptions about the distribution of the data.

Generative Approach

- Infer the prior class probabilities $p(C_k)$.
- Estimate the class-conditional densities $p(\mathbf{x}|C_k)$ for each class C_k.
- Use Bayes' theorem to find the class posterior probabilities:

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)p(C_k)}{p(\mathbf{x})} \tag{1}$$

where

$$p(x) = \sum_k p(\mathbf{x}|C_k)p(C_k) \tag{2}$$

- Use decision theory to determine class membership for each new input $\mathbf{x}$.

Two-class case:

$$p(C_1|\mathbf{x}) = \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_1)p(C_1) + p(\mathbf{x}|C_2)p(C_2)}$$

$$= \frac{1}{1+e^{-a}} = \sigma(a) \tag{3}$$

where

$$a = \ln \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_2)p(C_2)} \tag{4}$$

$$\sigma(a) = \frac{1}{1+e^{-a}} \quad \text{(logistic sigmoid function)} \tag{5}$$

$K > 2$ classes:

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)p(C_k)}{\sum_{j=1}^{K} p(\mathbf{x}|C_j)p(C_j)} = \frac{e^{a_k}}{\sum_{j=1}^{K} e^{a_j}} \tag{6}$$

where

$$a_k = \ln\ p(\mathbf{x}|C_k)p(C_k) \tag{7}$$

$$\text{softmax}(a_k) = \frac{e^{a_k}}{\sum_{j=1}^{K} e^{a_j}} \quad \text{(softmax function)} \tag{8}$$

Probabilistic Generative Models

- Lets assume the class-conditional densities are Gaussian with the same covariance matrix:

$$p(\mathbf{x}|C_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} e^{\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu}_k)\right)} \tag{9}$$

- Two class case first. We can show the following result:

$$P(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + \omega_0) \tag{10}$$

$$\mathbf{w} = \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \tag{11}$$

$$\omega_0 = -\frac{1}{2}\boldsymbol{\mu}_1^T \Sigma^{-1} \boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T \Sigma^{-1} \boldsymbol{\mu}_2 + \ln \frac{p(C_1)}{p(C_2)} \tag{12}$$

- How?

Probabilistic Generative Models

$$P(C_1|\mathbf{x}) = \sigma(a) = \frac{1}{1+e^{-a}} \tag{13}$$

$$\begin{aligned} a &= \ln \underbrace{p(\mathbf{x}|C_1)}_{\text{Replace with Eq.9}} - \ln \underbrace{p(\mathbf{x}|C_2)}_{\text{Replace with Eq.9}} + \ln \frac{p(C_1)}{p(C_2)} \\ &= -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_1) + \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_2)^T \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_2) \\ &+ \ln \frac{p(C_1)}{p(C_2)} \end{aligned} \tag{14}$$

Probabilistic Generative Models

Because we want to have $a = \mathbf{w}^T \mathbf{x} + \omega_0$, we shift the notations around (see appendix Eq. 45), and get:

$$\mathbf{w} = \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \tag{15}$$

$$\omega_0 = -\frac{1}{2}\boldsymbol{\mu}_1^T \Sigma^{-1} \boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T \Sigma^{-1} \boldsymbol{\mu}_2 + \ln \frac{p(C_1)}{p(C_2)} \tag{16}$$

- We have shown:

$$
\begin{aligned}
P(C_1|\mathbf{x}) &= \sigma(\mathbf{w}^T\mathbf{x} + \omega_0) \\
\mathbf{w} &= \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\
\omega_0 &= -\frac{1}{2}\boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2 + \ln\frac{p(C_1)}{p(C_2)}
\end{aligned}
$$

- Decision boundary:

$$p(C_1|\mathbf{x}) = p(C_2|\mathbf{x}) = 0.5 \tag{17}$$

$$\Rightarrow \frac{1}{1 + e^{-(\mathbf{w}^T\mathbf{x}+\omega_0)}} = 0.5 \Rightarrow \mathbf{w}^T\mathbf{x} + \omega_0 = 0 \tag{18}$$

Probabilistic Generative Models

- $K > 2$ classes:

$$P(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)p(C_k)}{\sum_j p(\mathbf{x}|C_j)p(C_j)} = \frac{e^{a_k}}{\sum_j e^{a_j}} \tag{19}$$

$$a_k = \ln\ p(\mathbf{x}|C_k)p(C_k) \tag{20}$$

- We can show the following result:

$$p(C_k|\mathbf{x}) = \frac{e^{\mathbf{w}_k^T\mathbf{x}+\omega_{k0}}}{\sum_j e^{\mathbf{w}_j^T\mathbf{x}+\omega_{j0}}} \tag{21}$$

$$\mathbf{w}_k = \Sigma^{-1}\boldsymbol{\mu}_k \tag{22}$$

$$\omega_{k0} = -\frac{1}{2}\boldsymbol{\mu}_k^T\Sigma^{-1}\boldsymbol{\mu}_k + \ln\ p(C_k) \tag{23}$$

Maximum Likelihood Solution

- We have a parametric functional form for the class-conditional densities:

$$p(\mathbf{x}|C_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} e^{\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu}_k)\right)} \tag{24}$$

- We can estimate the parameters and the prior class probabilities using maximum likelihood.
  - Two class case with shared covariance matrix.
  - Training data:

$$\{\mathbf{x}_n, y_n\}, \quad n = 1, ..., N$$

$$y_n = 1 \text{ denotes class } C_1; \ y_n = 0 \text{ denotes class } C_2;$$

$$\text{Priors: } p(C_1) = \gamma, p(C_2) = 1 - \gamma$$

Maximum Likelihood Solution

- For a data point $\mathbf{x}_n$ from class C_1, we have $y_n = 1$ and therefore:

$$p(\mathbf{x}_n, C_1) = p(\mathbf{x}|C_1)p(C_1) = \gamma\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \Sigma) \tag{25}$$

- For a data point $\mathbf{x}_n$ from class C_2, we have $y_n = 0$ and therefore:

$$p(\mathbf{x}_n, C_2) = p(\mathbf{x}|C_2)p(C_2) = (1-\gamma)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \Sigma) \tag{26}$$

- Assuming observations are drawn independently, the likelihood function is as below:

$$\begin{aligned} p(\mathcal{D}|\gamma, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma) &= \prod_{n=1}^{N} \left[p(\mathbf{x}_n, C_1)\right]^{y_n} \left[p(\mathbf{x}_n, C_2)\right]^{1-y_n} \\ &= \prod_{n=1}^{N} \left[\gamma\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \Sigma)\right]^{y_n} \left[(1-\gamma)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \Sigma)\right]^{1-y_n} \end{aligned}$$

Maximum Likelihood Solution

- We want to find the values of the parameters that **maximize the likelihood function**, i.e., fit a model that best describes the observed data.

$$p(\mathcal{D}|\gamma, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma) = \prod_{n=1}^{N} \left[\gamma \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \Sigma)\right]^{y_n} \left[(1-\gamma)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \Sigma)\right]^{1-y_n} \tag{27}$$

- As usual, we consider the log of the likelihood:

$$\begin{aligned} \ln\ p(\mathcal{D}|\gamma,\boldsymbol{\mu}_1,\boldsymbol{\mu}_2,\Sigma) = \sum_{n=1}^{N} \Big[& y_n \ln\gamma + y_n \ln\ \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \Sigma) \\ & + (1-y_n)\ln\ (1-\gamma) + (1-y_n)\ln\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \Sigma)\Big] \end{aligned} \tag{28}$$

Maximum Likelihood Solution - parameter γ

Log Likelihood:

$$\ln\, p(\mathcal{D}|\gamma,\boldsymbol{\mu}_1,\boldsymbol{\mu}_2,\Sigma) = \sum_{n=1}^{N} \big[y_n \ln\gamma + y_n \ln\, \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1,\Sigma) + (1-y_n)\ln\,(1-\gamma) + (1-y_n)\ln\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2,\Sigma) \big]$$

- We first maximize the log-likelihood with respect to γ (set derivate to 0)

$$\gamma = \frac{1}{N}\sum_{n=1}^{N} y_n = \frac{N_1}{N} = \frac{N_1}{N_1 + N_2} \tag{29}$$

- The maximum likelihood estimate of γ is the fraction of points in class C_1. (For multi-class: ML estimate for $p(C_k)$ is given by the fraction of points in the training set in C_k).

Maximum Likelihood Solution - parameter $\boldsymbol{\mu}$

Log Likelihood:

$$\ln\, p(\mathcal{D}|\gamma,\boldsymbol{\mu}_1,\boldsymbol{\mu}_2,\Sigma) = \sum_{n=1}^{N} \Big[y_n \ln \gamma + y_n \ln\, \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \Sigma) + (1-y_n)\ln\,(1-\gamma) + (1-y_n)\ln \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \Sigma) \Big]$$

- We them maximize the log-likelihood with respect to $\boldsymbol{\mu}_1$ (set derivate to 0)

$$\boldsymbol{\mu}_1 = \frac{1}{N_1} \sum_{n=1}^{N} y_n \mathbf{x}_n \qquad (30)$$

- The maximum likelihood estimate of $\boldsymbol{\mu}_1$ is the sample mean of all input $\mathbf{x}_n$ in class C_1. Same for $\boldsymbol{\mu}_2$.

Maximum Likelihood Solution - parameter Σ

- Maximize the log-likelihood with respect to Σ (set derivate to 0), we obtain the estimate Σ_{ML}

$$\Sigma_{ML} = \frac{N_1}{N} S_1 + \frac{N_2}{N} S_2 \tag{31}$$

where

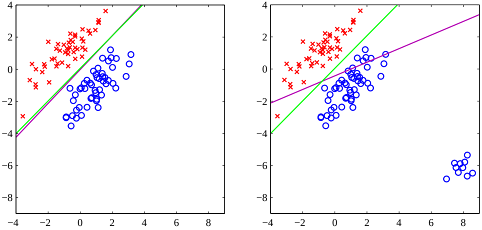
$$S_1 = \frac{1}{N_1} \sum_{n \in C_1} (\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^T \tag{32}$$

$$S_2 = \frac{1}{N_2} \sum_{n \in C_2} (\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^T \tag{33}$$

- The maximum likelihood estimate of the covariance is given by the weighted average of the sample covariance matrices associated with each of the classes.
- The results extend to K classes.

Summary so far

- We assumed $p(\mathbf{x}|(y=1)) \sim \mathcal{N}(\boldsymbol{\mu}_+, \Sigma_+)$ and $p(\mathbf{x}|(y=-1)) \sim \mathcal{N}(\boldsymbol{\mu}_-, \Sigma_-)$, and two class-probabilities $p(y=1)$ and $p(y=-1)$.
- This is called an generative model, as we have written down a full joint model over the data.
- We saw that violations of the model assumption can lead to "bad" decision boundaries.



Figures from Bishop PRML, 44a and b

Probabilistic Generative Models - parameters

- How many parameters did we estimate to fit Gaussian class-conditional densities (the generative approach)?

$$
\begin{aligned}
p(C_1) &\Rightarrow 1 \\
\text{2 mean vectors} &\Rightarrow 2d \\
\Sigma &\Rightarrow d + \frac{d^2 - d}{2} = \frac{d^2 + d}{2} \\
\text{total} &= 1 + 2d + \frac{d^2 + d}{2} = O(d^2)
\end{aligned}
$$

Part II: Logistic Regression

Logistic Regression

$$P(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + \omega_0) = f(\mathbf{x}) \tag{34}$$

- We use maximum likelihood to determine the parameters of the logistic regression model.

$$\{\mathbf{x}_n, y_n\}, \quad n = 1, ..., N$$

$$y_n = 1 \text{ denotes class } C_1; \; y_n = 0 \text{ denotes class } C_2;$$

- We want to find the values of **w** that maximize the posterior probabilities associated to the observed data
- Likelihood function:

$$\begin{aligned} \mathcal{L}(\mathbf{w}) &= \prod_{n=1}^{N} p(C_1|\mathbf{x}_n)^{y_n}(1 - p(C_1|\mathbf{x}_n))^{1-y_n} \\ &= \prod_{n=1}^{N} f(\mathbf{x}_n)^{y_n}(1 - f(\mathbf{x}_n))^{1-y_n} \end{aligned} \tag{35}$$

Logistic Regression

$$P(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + \omega_0) = f(\mathbf{x})$$

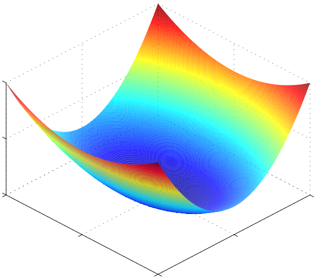
- We consider the negative logarithm of the likelihood (Cross Entropy):

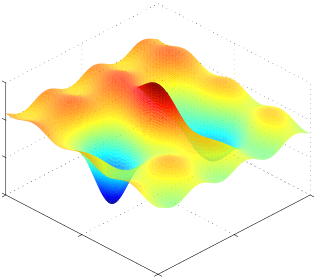
$$\begin{aligned}\mathcal{E}(\mathbf{w}) = -\ln\ \mathcal{L}(\mathbf{w}) &= -\ln\ \prod_{n=1}^{N} p(C_1|\mathbf{x}_n)^{y_n}(1 - p(C_1|\mathbf{x}_n))^{1-y_n} \\ &= -\sum_{n=1}^{N}(y_n \ln f(\mathbf{x}_n) + (1 - y_n)\ln\ (1 - f(\mathbf{x}_n)))\end{aligned} \tag{36}$$

- Thus:

$$\max \mathcal{L}(\mathbf{w}) = \min \mathcal{E}(\mathbf{w}) \tag{37}$$

The cost-function for logistic regression is convex.





- Fact: The negative log-likelihood is *convex* – this makes life much more easier.
- There are no local minima to get stuck in, and there is good optimization techniques for convex problems.

Logistic Regression - Gradient

$$P(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + \omega_0) = f(\mathbf{x})$$

- We compute the derivative of the error function with respect to $\mathbf{w}$

$$\frac{\partial \mathcal{E}(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}}\Big[-\ln \prod_{n=1}^{N} p(C_1|\mathbf{x}_n)^{y_n}(1 - p(C_1|\mathbf{x}_n))^{1-y_n}\Big] \quad (38)$$

- The derivative of the logistic sigmoid function:

$$\begin{aligned}
\frac{\partial}{\partial a}\sigma(a) &= \frac{\partial}{\partial a}\frac{1}{1+e^{-a}} = \frac{e^{-a}}{(1+e^{-a})^2} \\
&= \frac{1}{1+e^{-a}}\frac{e^{-a}}{(1+e^{-a})} = \frac{1}{1+e^{-a}}(1 - \frac{1}{1+e^{-a}}) \\
&= \sigma(a)(1-\sigma(a))
\end{aligned} \quad (39)$$

The derivative of the loss function with respect to $\mathbf{w}$ is ?

Probabilistic Discriminative Models - parameters

- Two-class case:

$$P(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + \omega_0) = f(\mathbf{x})$$

$$P(C_2|\mathbf{x}) = 1 - P(C_1|\mathbf{x})$$

- This model is know as Logistic Regression.
- Assuming $\mathbf{x} \in \mathbb{R}^d$, how many parameters do we need to estimate?

$$d + 1$$

Summary so far

- From the previous introduction, we know that

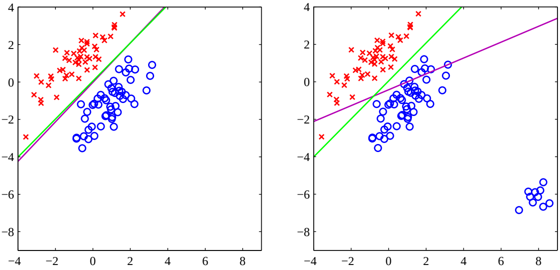
$$P(y = 1|\mathbf{x}) = \sigma(a(\mathbf{x}))$$

  where $\sigma(a) = 1/(1 + \exp(-a))$ and $a(x) = \mathbf{w}^\top \mathbf{x} + \omega_o$.
- Notation is simpler if we use 0 and 1 as class labels, so we define $y_n = 1$ as the label for the positive class, and $y_n = 0$ as label for the negative class.
- In other words, $y|x \sim \text{Bernoulli}(\sigma(f(x)))$.
- The parameters of $a(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + \omega_o$ can be learned by minimizing the negative log-likelihood $L(\mathbf{w}) = -\sum_n \{y_n \log p(y_n|\mathbf{x}_n, \mathbf{w}) + (1-y_n) \log(1-p(y_n|\mathbf{x}_n, \mathbf{w}))\}$
- This is a discriminative approach to classification, as we only model the labels, and not the inputs.

Summary: Discriminative vs. Generative

Decision rule and function shape of $p(y|\mathbf{x})$ will be the same for the generative and the discriminative model, but the parameters were obtained differently.

Maximum likelihood estimation of Logistic Regression



Bishop PRML Figure 44 a and b

- Logistic regression is a *much* better algorithm than the algorithms we discussed last week.
- Need to optimize log-likelihood numerically.
- People typically minimize the negative log-likelihood $\mathcal{L}$ rather than maximize the log-likelihood...
- To numerically minimize the negative log-likelihood, we need its gradient (and maybe its hessian)

Multiclass Logistic Regression

- Multiclass case:

$$p(C_k|\mathbf{x}) = \frac{e^{\mathbf{w}_k^T\mathbf{x}+\omega_{k0}}}{\sum_j e^{\mathbf{w}_j^T\mathbf{x}+\omega_{j0}}} = f_k(\mathbf{x}) \tag{40}$$

- We use maximum likelihood to determine the parameters:

$$\{\mathbf{x}_n, y_n\}, \quad n = 1, ..., N$$

$$y_n = (0, ..., 1, ..., 0) \text{ denotes class } C_k$$

- We want to find the values of $\mathbf{w}_1, ..., \mathbf{w}_k$ that maximize the posterior probabilities associated to the observed data likelihood function:

$$\mathcal{L}(\mathbf{w}_1, ..., \mathbf{w}_k) = \prod_{n=1}^{N}\prod_{k=1}^{K} p(C_k|\mathbf{x}_n)^{y_{nk}} = \prod_{n=1}^{N}\prod_{k=1}^{K} f_k(\mathbf{x}_n)^{y_{nk}} \tag{41}$$

Multiclass Logistic Regression

$$\mathcal{L} = \prod_{n=1}^{N} \prod_{k=1}^{K} p(C_k|\mathbf{x}_n)^{y_{nk}} = \prod_{n=1}^{N} \prod_{k=1}^{K} f_k(\mathbf{x}_n)^{y_{nk}}$$

- Consider the negative logarithm of the likelihood:

$$\mathcal{E}(\mathbf{w}_1, ..., \mathbf{w}_k) = -\ln \mathcal{L}(\mathbf{w}_1, ..., \mathbf{w}_k) = -\sum_{n=1}^{N} \sum_{k=1}^{K} y_{nk} \ln f_k(\mathbf{x}_n) \quad (42)$$

$$\min_{\mathbf{w}_j} \mathcal{E}((\mathbf{w}_j)) \quad (43)$$

- The gradient of the error function w.r.t one of the parameter vectors:

$$\frac{\partial}{\partial \mathbf{w}_j} \mathcal{E}(\mathbf{w}_1, ..., \mathbf{w}_k) = \frac{\partial}{\partial \mathbf{w}_j} \left[-\sum_{n=1}^{N} \sum_{k=1}^{K} y_{nk} \ln\ f_k(\mathbf{x}_n) \right] \quad (44)$$

Multiclass Logistic Regression

$$\frac{\partial}{\partial \mathbf{w}_j}\mathcal{E}(\mathbf{w}_1, ..., \mathbf{w}_k) = \frac{\partial}{\partial \mathbf{w}_j}\left[-\sum_{n=1}^{N}\sum_{k=1}^{K} y_{nk} \ln\ f_k(\mathbf{x}_n)\right]$$

- The derivatives of the softmax function:

$$\frac{\partial}{\partial a_k} f_k = \frac{\partial}{\partial a_k}\frac{e^{a_k}}{\sum_j e^{a_j}} = \frac{e^{a_k}\sum_j e^{a_j} - e^{a_k}e^{a_k}}{(\sum_j e^{a_j})^2} = f_k - f_k^2 = f_k(1-f_k)$$

- Thus:

$$\text{for } j \neq k, \frac{\partial}{\partial a_j} f_k = \frac{\partial}{\partial a_j} f_k = \frac{\partial}{\partial a_j}\frac{e^{a_k}}{\sum_j e^{a_j}} = \frac{-e^{a_k}e^{a_j}}{(\sum_j e^{a_j})^2} = -f_k f_j$$

- Compact expression (I_{kj} are the elements of the identity matrix)

$$\frac{\partial}{\partial a_j} f_k = f_k(I_{kj} - f_j)$$

Multiclass Logistic Regression

$$\nabla_{\mathbf{w}_j}\mathcal{E}(\mathbf{w}_1, ..., \mathbf{w}_k) = \sum_{n=1}^{N}(f_{nj} - y_{nj})\mathbf{x}_n$$

Multiclass Logistic Regression

$$\nabla_{\mathbf{w}_j}\mathcal{E}(\mathbf{w}_1, ..., \mathbf{w}_k) = \sum_{n=1}^{N}(f_{nj} - y_{nj})\mathbf{x}_n$$

- It can be shown that $\mathcal{E}$ is a convex function of $\mathbf{w}$. Thus, it has a unique minimum.
- For a batch solution, we can use the Newton-Raphson optimization technique.
- Online solution (SGD):

$$\mathbf{w}_j^{t+1} = \mathbf{w}_j^t - \eta\nabla_{\mathbf{w}_j}\mathcal{E}_n(\mathbf{w}) = \mathbf{w}_j^t - \eta(f_{nj} - y_{nj})\mathbf{x}_n$$

Acknowledgements

Slides adapted from Dr. Carlotta Domeniconi's *Pattern Recognition* at George Mason University.

$$
\begin{aligned}
a &= -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_1) + \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_2)^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_2) + \ln\frac{p(C_1)}{p(C_2)} \\
&= -\frac{1}{2}\mathbf{x}^T\Sigma^{-1}\mathbf{x} + \frac{1}{2}\boldsymbol{\mu}_1^T\Sigma^{-1}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\Sigma^{-1}\boldsymbol{\mu}_1 - \frac{1}{2}\boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_1 \\
&+ \frac{1}{2}\mathbf{x}^T\Sigma^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_2^T\Sigma^{-1}\mathbf{x} - \frac{1}{2}\mathbf{x}^T\Sigma^{-1}\boldsymbol{\mu}_2 + \frac{1}{2}\boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2 + \ln\frac{p(C_1)}{p(C_2)} \\
&= \boldsymbol{\mu}_1^T\Sigma^{-1}\mathbf{x} - \boldsymbol{\mu}_2^T\Sigma^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2 + \ln\frac{p(C_1)}{p(C_2)} \\
&= (\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^T\Sigma^{-1}\mathbf{x} + \omega_0 = \mathbf{w}^T\mathbf{x} + \omega_0 \qquad (45)
\end{aligned}
$$