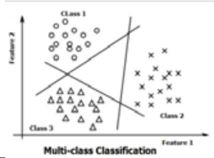
Classification

- The goal of classification is to take an input vector **x** and to assign y to one of K discrete classes C_k where $k = 1, \ldots, K$, i.e.,

  $f : \mathrm{x} \in R^D \rightarrow y \in \{1, \ldots, K\}$
- The input space is divided into K regions (one for each class), whose boundaries are called decision boundaries or decision surfaces.

Linear classification



Classification Approaches

- Deterministic approach: Construct a discriminant function f(x) that directly maps each input vector to a specific class
  - **Linear deterministic classifier**
    - SVM
    - Perceptron
- Probabilistic Approach: Model the probability distribution **x** and y and then use this distribution to make optimal decisions
  - Discriminative approach – p(y|x)
  - Generative approach –p(x,y)

Linear Deterministic Classifiers

- Construct a linear discriminate function

$$f(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}^t \mathbf{x} + w_0$$

- Linear classification

$$\mathrm{y}(\boldsymbol{x}, \boldsymbol{w}) = \phi(f(\boldsymbol{x}, \boldsymbol{w})) \quad y \ni \{1, 2, \ldots K\}$$

where $\phi()$ is the activation function that maps **x** into y that belongs to one of K different classes.

Linear Deterministic Classifiers

- Binary discriminant classifiers map input into two classes
- Multi-class discriminant classifiers map input into K classes, K>2.

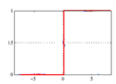
Binary Discriminant Classifier

•Construct a linear discriminant function

f(**x**)=$\mathbf{w}^t\mathbf{x}+w_0$

• Choose the *activation function as a step function, i.e.,*

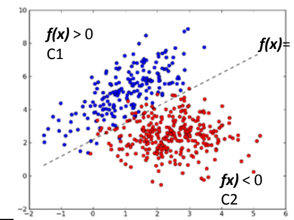
$$\mathbf{y}(\mathbf{w},\mathbf{x}) = \varphi(f(\mathbf{x})) = \begin{cases} C_1 & if \;\; f(\mathbf{x}) > 0 \\ C_2 & else \end{cases}$$



•Input vector **x** is assigned to class C_1 if the discriminant function f() is larger than 0 and to class C_2 otherwise.

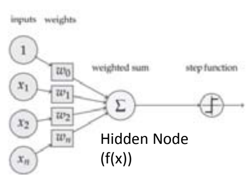
Binary Discriminant (cont'd)

Discriminant function f(**x**) $=\boldsymbol{w}^t\boldsymbol{x}+w_0$ =0 corresponds to a (N – 1)-dimensional hyperplane within the N-dimensional input space. **w** determines the orientation and location of the decision surface. For 2D **x**, $\boldsymbol{w}^t\boldsymbol{x}+w_0$ = 0 is a line



The Perceptron Algorithm

x



y ϵ {+1.-1}

$$y(\mathbf{x}) = \phi(\mathbf{w}^t\mathbf{x}+w_0)$$

where nonlinear activation function f() is given by a step function:

$$y(\mathbf{x}) = \begin{cases} +1 & if \;\; \mathbf{w}^t\mathbf{x}+w_0 > 0 \\ -1 & else \end{cases}$$

Binary Support Vector Machine

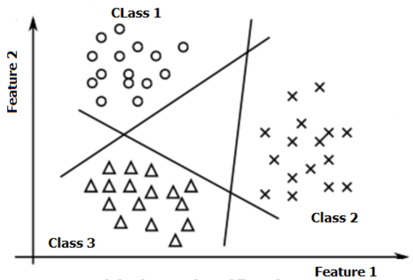
Two-class classification with the linear model is

$$f(\mathbf{x}) = w^t x + w_0 = \mathbf{w}^t\mathbf{x} + w_0$$

Given the target y={-1,+1}, an input **x** is classified into +1 if f(**x**)>0 and to -1 else

Multi-class discriminant classifiers

- The output y represents K classes (K > 2). The goal is to map input **x** into multiple classes



Multi-class Classification

Multi-class discriminant classifiers

- Construct K linear discriminant functions, one for each class :

$$f_k(\mathbf{x}) = \mathbf{w}_k{}^t\mathbf{x} + w_{k,0}$$

- Apply the max activation function (pick the max), assign x to class C_k, i.e.,

$$y = \arg\max_k(f_k(\mathbf{x}))$$

Linear Probabilistic Classifiers

Model the probability distribution **x** and y and then use this distribution to make optimal decisions

- **Discriminative approach** – p(y|x)
- Generative approach –p(x,y)

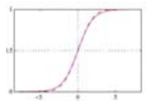
Linear Discriminative Classifiers

•Construct the probability of p(y|**x**) and perform classification based on p(y|**x**), i.e.,

$$y^* = \arg\max_{y\in\{1,2,\ldots,K\}} p(y\,|\,\mathbf{x})$$

Binary Discriminative Classifier

- Construct the discriminant function
  f($\mathbf{x}$)=$\mathbf{w}^t\mathbf{x}+w_0$



- Compute p(y|x) by choosing the sigmoid activity function, i.e.,
  $p(y|\boldsymbol{x},\boldsymbol{w}) = \sigma(f(\boldsymbol{x}))$=$\frac{1}{1+e^{-y*f(x)}}$
- The sigmoid function σ () maps the discriminant function f($\mathbf{x}$) into a number between 0 and 1 (probability).
- $\mathbf{x}$ is classified into y=1 if p(y=1|x)>0.5 and y= -1 otherwise.
- This is the so-called logistic regression

Multiclass Discriminative Models

•Output y ∈{1,2,..,K} K>2 and construct P(y|$\mathbf{x}$) by a *multiclass sigmoid function (σ_M)* using K discriminant functions $f_k(\mathbf{x},\mathbf{w}_k, w_{k,0})$

$$p(y=k|\boldsymbol{x})=\sigma_M(f_k(\mathbf{x},\boldsymbol{w}_k,w_{k,0}))=\frac{\exp(\boldsymbol{w}_k^t x+w_{k,0})}{\sum_{j=1}^K \exp(\boldsymbol{w}_j^t x+w_{j,0})}$$

$$k^* = \arg\max_k p(y=k\,|\,\mathbf{x})$$

- This is called softmax classifier or multinomial logistic regression

Classifier Summary

- Deterministic Classifiers
  - Binary linear classifier discriminant function
    - f(x)=$w^t x+w_0$-binary discriminant function
    - y=ϕ(f(x))-ϕ() is activation function
    - ϕ() is step function-binary discriminant classifier
      Perceptron
      SVM
  - Multi-class classifier
    - $f_k(x)=w_k^t x+w_{k,0}$-contruct K discriminant functions
    - y=ϕ(f_k(x))=argmax $_k$ f_k(x)-multi-class discriminant classifier

Classifier Summary

- Discriminative Classifiers p(y|x)
  - Binary classifier
    p(y|x)= σ(yf(x))- ϕ() =sigmoid function-logistic regression
  - Multi-class classifier
    - p(y=k|x)=$\sigma_M(f_k(x))=\frac{e^{f_k(x)}}{\sum_{k=1}^K e^{f_k(x)}}$
    - σ_M is multi-class sigmoid function

Classifier Learning

- Deterministic classifier learning
  - Binary discriminant classifier learning
  - Multi-class discriminant classifier learning
- Discriminative classifier learning
  - Binary classifier learning
  - Multi-class classifier learning

Binary Discriminant Classifier Learning

- Given the training data **D**={**x**[m], y[m]}, m=1,2,..,M, learn the model parameters **Θ** by minimizing the overall loss function L(**D**:**Θ**) of the form

$$L(\mathbf{D}:\mathbf{\Theta}) = \frac{1}{M}\sum_{m=1}^{M} l(\mathbf{x}[m], y[m], \mathbf{\Theta}) + \lambda R(\mathbf{\Theta})$$

where the first term is the individual loss function and the second term is the regularization

Binary Classifier Loss Functions

For binary classifier y∈ {-1,+1}

- 0-1 loss function – the loss is 0 if the predicted labels and groundtruth labels are the same and 1 otherwise

$$l_{0/1} = \begin{cases} 0 \ \ if \ y(\mathbf{w}^t\mathbf{x}+w_0) > 0 \\ 1 \ \ else \end{cases}$$

- Hinge loss function

$$l_h = \max(0, 1 - y(\mathbf{w}^T\mathbf{x}+w_0))$$

when y ($\mathbf{w}^t\mathbf{x}+w_0$) >1 the hinge loss is zero.

Otherwise the loss increases linearly with $\mathbf{w}^t\mathbf{x}+w_0$.

Hinge loss is an upper bound of 0/1 loss

Classifier Learning with 0/1 Loss Function

- For the 0/1 loss function, the total loss function can be written as

$$L_{0/1}(\mathbf{D}:\mathbf{w}, w_0) = \frac{1}{M}\sum_{m=1}^{M} l_{0/1}(y[m](\mathbf{w}^t\mathbf{x}[m]+w_0) + \lambda R(\mathbf{w}, w_0)$$

- The parameters can be obtained by minimizing the loss function, i.e.,

$$\mathbf{w}^*, w_0^* = \arg\min_{\mathbf{w}, w} L_{0/1}(\mathbf{D}:\mathbf{w}, w_0)$$

Classifier Learning with 0/1 Loss Function

- Gradient-based method cannot solve for the parameters by minimizing the total 0/1 loss function $L_{0/1}(\mathbf{D}:\mathbf{w},w)$ since the gradient of $l_{0/1}(y[m](\mathbf{w}^t\mathbf{x}[m]+w_0))$ is zero everywhere
- Minimizing the overall 0/1 loss function is intractable
- Solution is to minimize an alterative loss function such as the hinge loss function

Binary Classifier Learning with Hinge Loss

- For the hinge loss function, the total loss function can be written as

$$L_H(D:\mathbf{w},w_0)=\frac{1}{M}\sum_{m=1}^{M}\max(0,1-y[m](\mathbf{w}^t\mathbf{x}[m]+w_0)+\lambda R(\mathbf{w},w_0)$$

- The parameters can be obtained by minimizing the loss function, i.e.,

$$\mathbf{w}^*,w_0^*=\arg\min_{\mathbf{w},w} L_H(\mathbf{D}:\mathbf{w},w_0)$$

Binary Classifier Learning with Hinge Loss

Let

$$\mathbf{\Theta}=(\mathbf{w},w_0)^t \quad \mathbf{X}=(\mathbf{x},1)^t \quad z[m]=y[m](\mathbf{w}^t\mathbf{x}[m]+w_0)$$

$$\begin{aligned}L_H(D:\mathbf{\Theta})&=\frac{1}{M}\sum_{m=1}^{M}\max(0,1-y[m](\mathbf{w}^t\mathbf{x}[m]+w_0)+\lambda R(\mathbf{w},w_0)\\&=\frac{1}{M}\sum_{m=1}^{M}\max(0,1-y[m]\mathbf{\Theta}^t\mathbf{X}[m])+\lambda R(\mathbf{\Theta})\\&=\frac{1}{M}\sum_{m=1}^{M}\max(0,1-z[m])+\lambda R(\mathbf{\Theta})\end{aligned}$$

$$\frac{\partial L_H(D:\mathbf{\Theta})}{\partial\mathbf{\Theta}}=\frac{1}{M}\sum_{m=1}^{M}\frac{\partial\max(0,1-z[m])}{\partial z}\frac{\partial z}{\partial\mathbf{\Theta}}+\lambda\frac{\partial R(\mathbf{\Theta})}{\partial\mathbf{\Theta}}$$

$$\frac{\partial\max(0,1-z[m])}{\partial z}=\begin{cases}0 & if\ \ z[m]>1\\-1 & else\end{cases}$$

$$\frac{\partial z}{\partial\mathbf{\Theta}}=\frac{\partial[y[m](\mathbf{w}^t\mathbf{x}[m]+w_0)]}{\partial\mathbf{\Theta}}=\frac{\partial[y[m]\mathbf{\Theta}^t\mathbf{X}[m]]}{\partial\mathbf{\Theta}}=y[m]\mathbf{X}[m]$$

$$\frac{\partial L_H(D:\mathbf{\Theta})}{\partial\mathbf{\Theta}}=\frac{-1}{M}\sum_{m=1}^{M}y[m]\mathbf{X}[m]I[m]+\lambda\frac{\partial R(\mathbf{\Theta})}{\partial\mathbf{\Theta}},\ \text{where I[m]}=\begin{cases}0\text{ if z[m]}>1\\1\text{ else}\end{cases}$$

Similarly, we can compute $\frac{\partial R(\boldsymbol{\Theta})}{\partial\boldsymbol{\Theta}}$. Given $\frac{\partial L_H(D:\boldsymbol{\Theta})}{\partial\boldsymbol{\Theta}}$, as there is no analytic solution, gradient descent can apply to update $\boldsymbol{\Theta}$ iteratively.

Multiclass Discrimnative Classifier Learning

- Suppose y $\in$\{1,2,…,K\}, we can construct a discriminate function for each class, i.e., $f_k(\mathbf{x})=\mathbf{w}^t_k\mathbf{x}+w_{k,0}$
- A generalized hinge loss for K class can be constructed as follows

$$l_h(\mathbf{x},y=k,\mathbf{W})=\max(0,1+\max_{k'\neq k}(\mathbf{w}^t_{k'}\mathbf{x}+w_{k',0}-\mathbf{w}^t_k\mathbf{x}-w_{k,0}))$$

where $\mathbf{W}=(\mathbf{w}_1,w_{1,0},\mathbf{w}_2,w_{2,0},\ldots,\mathbf{w}_K,w_{K,0})^t$

- **W** can be learnt by minimizing the generalized hinge loss function

Discriminative Classifier Learning

- Binary logistic regression learning
- Multi-class logistic regression learning

Binary Logistic Regression Learning

- Given the training data **D**={**x**[m], y[m]}, m=1,2,..,M and y[m] ∈{-1,+1}, learn the model parameters **Θ** by minimizing the negative log conditional likelihood, i.e.,
- $$-LCL(\mathbf{D}:\boldsymbol{\Theta}) = -\sum_{m=1}^{M}\log p(y[m] \mid \mathbf{x}[m]) = -\sum_{m=1}^{M}\log \sigma(yf(\mathbf{x}[m]))$$
$$= -\sum_{m=1}^{M}\log\{\frac{1}{1+e^{-y[m](\mathbf{w}^t\mathbf{x}[m]+w_0)}}\}$$
$$= \sum_{m=1}^{M}\log\{1+e^{-y[m]\boldsymbol{\Theta}^t\mathbf{X}[m]}\}$$

where $\boldsymbol{\Theta} = \begin{bmatrix}\mathbf{w}\\ w_0\end{bmatrix}$ $\mathbf{X}[m] = \begin{bmatrix}\mathbf{x}[m]\\ 1\end{bmatrix}$

σ is the sigmoid function

Binary Logistic Regression Learning (cont'd)

- **Θ** is found by minimizing the negative log likelihood function ,i.e.,

$$\boldsymbol{\Theta}^* = \arg\min_{\boldsymbol{\Theta}} -LCL(D:\boldsymbol{\Theta})$$

- There is no closed-form solution. Gradient descent can be used to estimate **Θ** iteratively, i.e.,

$$\boldsymbol{\Theta}^t = \boldsymbol{\Theta}^{t-1} - \eta\frac{\partial -LCL(\mathbf{D}:\boldsymbol{\Theta}^t)}{\partial\boldsymbol{\Theta}}\Big|_{\boldsymbol{\Theta}^{t-1}}$$

where

$$\frac{\partial -LCL(\mathbf{D}:\boldsymbol{\Theta}^t)}{\partial\boldsymbol{\Theta}} = \frac{\partial\sum_{m=1}^{M}\log\{1+e^{-y[m]\boldsymbol{\Theta}^t\mathbf{X}[m]}\}}{\partial\boldsymbol{\Theta}}$$
$$= -\sum_{m=1}^{M}\frac{e^{-y[m]\boldsymbol{\Theta}^t\mathbf{X}[m]}}{1+e^{-y[m]\boldsymbol{\Theta}^t\mathbf{X}[m]}}y[m]\mathbf{X}[m]$$

Binary Logistic Regression Learning (cont'd)

Alternatively, the gradient can be computed as

$$\frac{\partial -LCL(\mathbf{D}:\boldsymbol{\Theta}^t)}{\partial\boldsymbol{\Theta}} = -\frac{\partial\sum_{m=1}^{m}\log\sigma(y[m]f(\mathbf{x}[m]))}{\partial\boldsymbol{\Theta}}$$
$$= -\sum_{m=1}^{M}\frac{1}{\sigma(y[m]f(\mathbf{x}[m]))}\frac{\partial\sigma(y[m]f(\mathbf{x}[m]))}{\partial(y[m]f(\mathbf{x}[m])}\frac{\partial(y[m]f(\mathbf{x}[m]))}{\partial\boldsymbol{\Theta}}$$
$$= -\sum_{m=1}^{M}\frac{1}{\sigma(y[m]f(\mathbf{x}[m]))}\sigma(y[m]f(\mathbf{x}[m]))(1-\sigma(y[m]f(\mathbf{x}[m])))y[m]\mathbf{X}[m]$$
$$= -\sum_{m=1}^{M}(1-\sigma(y[m]f(\mathbf{x}[m]))y[m]\mathbf{X}[m]$$

Multiclass Logistic Regression Learning

- Suppose y $\in\{1,2,\ldots,K\}$, we can construct a discriminant function for each class, i.e., $f_k(\mathbf{x})=\mathbf{w}^t_k\mathbf{x}+w_{k,0}=\Theta_K^t\mathbf{X}$, where $\Theta_k=[\mathbf{w}^t,w_{k,0}]^t$ $\mathbf{X}=[\mathbf{x}\ \ 1]^t$
- For output y, we use 1-of-K encoding, where we use a Kx1 binary vector **y**, whose contains a single 1 for element k (the correct class) and 0 elsewhere.
- For 5 classes and the input belongs to class 2, **y** is

$$\mathbf{y}=\begin{bmatrix}0\\1\\0\\0\\0\end{bmatrix}$$

Multiclass Logistic Regression Learning

Given training data D={**X**[m],y[m]}, m=1,2,..M, learn the parameters $\boldsymbol{\Theta}$ by minimizing the total negative log conditional likelihood

$$L(\mathbf{D}:\boldsymbol{\Theta})=-\sum_{m=1}^{N}\log p(y[m]\,|\,\mathbf{X}[m],\boldsymbol{\Theta})=-\sum_{m=1}^{N}\log\prod_{k'=1}^{K}[p(y[m]=k'\,|\,\mathbf{X}[m],\boldsymbol{\Theta})]^{\mathbf{y}[m][k']}$$

$$=-\sum_{m=1}^{N}\sum_{k'=1}^{K}\mathbf{y}[m][k']\log p(y[m]=k'|\mathbf{X}[m],\boldsymbol{\Theta}))$$

$$=-\sum_{m=1}^{N}\sum_{k'=1}^{K}\mathbf{y}[m][k']\log\frac{\exp(\boldsymbol{\Theta}^t_{k'}\mathbf{X}[m])}{\sum_{j=1}^{K}\exp(\boldsymbol{\Theta}_j\mathbf{X}[m])}$$

Multiclass Logistic Regression Learning

$$L(\mathbf{D}:\boldsymbol{\Theta})=-\sum_{m=1}^{M}\log p(y[m]\,|\,\mathbf{X}[m],\boldsymbol{\Theta})=-\sum_{m=1}^{M}\log\prod_{k'=1}^{K}[p(y[m]=k'\,|\,\mathbf{X}[m],\boldsymbol{\Theta})]^{y[m][k']}$$

$$=-\sum_{m=1}^{M}\log\prod_{k'=1}^{K}[\sigma_M(f(\mathbf{X}[m],\boldsymbol{\Theta}_k))]^{y[m][k']}=-\sum_{m=1}^{M}\log\prod_{k'=1}^{K}\left(\frac{\exp(\boldsymbol{\Theta}^t_{k'}\mathbf{X}[m])}{\sum_{j=1}^{K}\exp(\boldsymbol{\Theta}^t_j\mathbf{X}[m])}\right)^{\mathbf{y}[m][k']}$$

$$=-\sum_{m=1}^{M}\sum_{k'=1}^{K}\left[\mathbf{y}[m][k']\log\frac{\exp(\boldsymbol{\Theta}^t_{k'}\mathbf{X}[m])}{\sum_{j=1}^{K}\exp(\boldsymbol{\Theta}^t_j\mathbf{X}[m])}\right]$$

$$=-\sum_{m=1}^{M}\sum_{k'=1}^{K}\left[\mathbf{y}[m][k']\left(\boldsymbol{\Theta}^t_{k'}\mathbf{X}[m]-\log\sum_{j=1}^{K}\exp(\boldsymbol{\Theta}^t_j\mathbf{X}[m])\right)\right]$$

$$\boldsymbol{\Theta}^*=\arg\min_{\boldsymbol{\Theta}}L(\mathbf{D}:\boldsymbol{\Theta})$$

σ_m is the multi-class sigmoid function

Multiclass Logistic Regression Learning

$$\frac{\partial L(\mathbf{D}:\boldsymbol{\Theta})}{\partial\boldsymbol{\Theta}}=\begin{bmatrix}\dfrac{\partial L(\mathbf{D}:\boldsymbol{\Theta})}{\partial\boldsymbol{\Theta}_1}\\\dfrac{\partial L(\mathbf{D}:\boldsymbol{\Theta})}{\partial\boldsymbol{\Theta}_2}\\..\\\dfrac{\partial L(\mathbf{D}:\boldsymbol{\Theta})}{\partial\boldsymbol{\Theta}_K}\end{bmatrix}$$

Multiclass Logistic Regression Learning

$$\frac{\partial L(\mathbf{D}:\mathbf{\Theta})}{\partial \mathbf{\Theta}_k} = \frac{\partial \sum_{m=1}^{M}\sum_{k'=1}^{K} \mathbf{y}[m][k'] \log \frac{\exp(\mathbf{\Theta}_{k'}^t \mathbf{X}[m])}{\sum_{j=1}^{K}\exp(\mathbf{\Theta}_j^t \mathbf{X}[m])}}{\partial \mathbf{\Theta}_k}$$

$$= -\frac{\partial \sum_{m=1}^{M}\sum_{k'=1}^{K} \{\mathbf{y}[m][k']\mathbf{\Theta}_{k'}^t \mathbf{X}[m] - \mathbf{y}[m][k'] \log \sum_{j=1}^{K}\exp(\mathbf{\Theta}_j^t \mathbf{X}[m])\}}{\partial \mathbf{\Theta}_k} = -\sum_{m=1}^{M}\{\mathbf{y}[m][k]\mathbf{X}[m] - \frac{\partial \overbrace{\sum_{k'=1}^{K}\mathbf{y}[m][k']}^{1} \log \sum_{j=1}^{K}\exp(\mathbf{\Theta}_j^t \mathbf{X}[m])}{\partial \mathbf{\Theta}_k}\}$$

$$= -\sum_{m=1}^{N}\{\mathbf{y}[m][k]\mathbf{X}[m] - \frac{\exp(\mathbf{\Theta}_k^t \mathbf{X}[m])\mathbf{X}[m]}{\sum_{j=1}^{K}\exp(\mathbf{\Theta}_j^t \mathbf{X}[m])}\} = -\sum_{m=1}^{N}\mathbf{X}[m]\{\mathbf{y}[m][k] - \sigma_M(\mathbf{\Theta}_k^t \mathbf{X}[m])\}$$

Multiclass Logistic Regression Learning

Θ can be solved iteratively gradient descent

$$\nabla_{\mathbf{\Theta}_k} L(\mathbf{D}:\mathbf{\Theta}) = \frac{\partial L(\mathbf{D}:\mathbf{\Theta})}{\partial \mathbf{\Theta}_k} = -\frac{\partial \sum_{m=1}^{M}\sum_{k'=1}^{K}\left[\mathbf{y}[m][k']\left(\mathbf{\Theta}_{k'}^t \mathbf{X}[m] - \log \sum_{j=1}^{K}\exp(\mathbf{\Theta}_j^t \mathbf{X}[m])\right)\right]}{\partial \mathbf{\Theta}_k}$$

$$= -\frac{\sum_{m=1}^{M}\sum_{k'=1}^{K}\left[\mathbf{y}[m][k']\partial\left(\mathbf{\Theta}_{k'}^t \mathbf{X}[m] - \log \sum_{j=1}^{K}\exp(\mathbf{\Theta}_j^t \mathbf{X}[m])\right)\right]}{\partial \mathbf{\Theta}_k} = -\sum_{m=1}^{M}\{\mathbf{y}[m][k]\mathbf{X}[m] - \frac{\partial \sum_{k'=1}^{K}\left[\mathbf{y}[m][k'] \log \sum_{j=1}^{K}\exp(\mathbf{\Theta}_j^t \mathbf{X}[m])\right]}{\partial \mathbf{\Theta}_k}\}$$

$$= -\sum_{m=1}^{M}\left\{\mathbf{y}[m][k]\mathbf{X}[m] - \sum_{k'=1}^{K}\left[\mathbf{y}[m][k']\frac{\exp(\mathbf{\Theta}_k^t \mathbf{X}[m])}{\sum_{j=1}^{K}\exp(\mathbf{\Theta}_j^t \mathbf{X}[m])}\mathbf{X}[m]\right]\right\} = -\sum_{m=1}^{M}\left\{\mathbf{y}[m][k]\mathbf{X}[m] - \frac{\exp(\mathbf{\Theta}_k^t \mathbf{X}[m])}{\sum_{j=1}^{K}\exp(\mathbf{\Theta}_j^t \mathbf{X}[m])}\mathbf{X}[m]\sum_{k'=1}^{K}[\mathbf{y}[m][k']]\right\}$$

$$= -\sum_{m=1}^{M}\mathbf{X}[m]\left\{\mathbf{y}[m][k] - \frac{\exp(\mathbf{\Theta}_k^t \mathbf{X}[m])}{\sum_{k'=1}^{K}\exp(\mathbf{\Theta}_{k'}^t \mathbf{X}[m])}\right\}$$

$$\mathbf{\Theta}_k^{t+1} = \mathbf{\Theta}_k^t - \eta \nabla_{\mathbf{\Theta}_k} L(\mathbf{D}:\mathbf{\Theta})$$

Repeat this for k=1, 2, …, K

Multiclass Logistic Regression Learning

Alternatively, the gradient can be compute as

$$\frac{\partial L(\mathbf{D}:\mathbf{\Theta})}{\partial \mathbf{\Theta}_k} = -\frac{\partial \sum_{m=1}^{M}\log \prod_{k'=1}^{K}[\sigma_M(f_{'}(\mathbf{x}[m],\mathbf{\Theta}_k))]^{y[m][k']}}{\partial \mathbf{\Theta}_k} = -\frac{\partial \sum_{m=1}^{M}\sum_{k'=1}^{M} y[m][k'] \log \sigma_M(f(\mathbf{X}[m],\mathbf{\Theta}_k)}{\partial \mathbf{\Theta}_k}$$

$$= -\sum_{m=1}^{M}\sum_{k'=1}^{K} y[m][k']\frac{\partial \log \sigma_M(f(\mathbf{X}[m],\mathbf{\Theta}_k))}{\partial \mathbf{\Theta}_k} = -\sum_{m=1}^{M}\left[y[m][k]\frac{\partial \log \sigma_M(f(\mathbf{X}[m],\mathbf{\Theta}_k)}{\partial \mathbf{\Theta}_k} + \sum_{k'=1,k'\neq k}^{k} y[m][k']\frac{\partial \log \sigma_M(f(\mathbf{X}[m],\mathbf{\Theta}_k)}{\partial \mathbf{\Theta}_k}\right]$$

$$= -\sum_{m=1}^{M}\left[y[m][k](1-\sigma_M(f\ (\mathbf{X}[m]),\mathbf{\Theta}_k)\mathbf{X}[m] - \sum_{k'=1,k'\neq k}^{k} y[m][k']\frac{\sigma_M(f(\mathbf{X}[m],\mathbf{\Theta}_{k'})\sigma_M(f_k(\mathbf{X}[m],\mathbf{\Theta}_k)}{\sigma_M(f(\mathbf{X}[m],\mathbf{\Theta}_{k'})}\mathbf{X}[m]\right]$$

$$= -\sum_{m=1}^{M}\left[y[m][k](1-\sigma_M(f\ (\mathbf{X}[m],\mathbf{\Theta}_k))\mathbf{X}[m] - \sum_{k'=1,k'\neq k}^{k} y[m][k']\sigma_M(f(\mathbf{X}[m],\mathbf{\Theta}_k)\mathbf{X}[m]\right]$$

$$= -\sum_{m=1}^{M}\left[y[m][k](1-\sigma_M(f(\mathbf{X}[m],\mathbf{\Theta}_k))\mathbf{X}[m] - \sum_{k'=1,k'\neq k}^{k} y[m][k']\sigma_M(f(\mathbf{X}[m],\mathbf{\Theta}_k)\mathbf{X}[m]\right]$$

$$= -\sum_{m=1}^{M}\mathbf{X}[m]\left[y[m][k] - \sigma_M(f\ (\mathbf{X}[m],\mathbf{\Theta}_k))\right]$$

Note

$$\frac{\partial \sigma_M(z)}{\partial z} = \sigma_M(z)(1-\sigma_M(z))$$

Multiclass Logistic Regression Learning

Add regularization

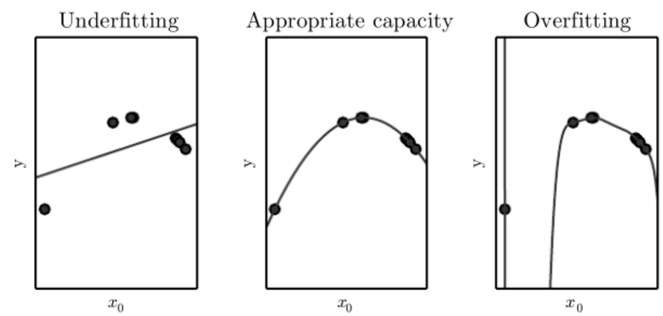
$$\mathbf{\Theta}_k^{t+1} = \mathbf{\Theta}_k^{t+1} - \eta[\nabla_{\mathbf{\Theta}_k} L(\mathbf{D}:\mathbf{\Theta}) + \lambda \nabla_{\mathbf{\Theta}_k} R(\mathbf{\Theta}_k)]$$

where R(Θ) can be L1-norm or squared L2-norm

Capacity, Overfitting, and Underfitting

- Capacity-a model's ability to fit a variety of functions. It is determined by the number of parameters the model has.
- Overfitting-a model performs well on training data but poorly on unseen testing data
- Underfitting-a model performs poorly both on training and testing data

Capacity, Overfitting, and Underfitting



Capacity, Overfitting, and Underfitting