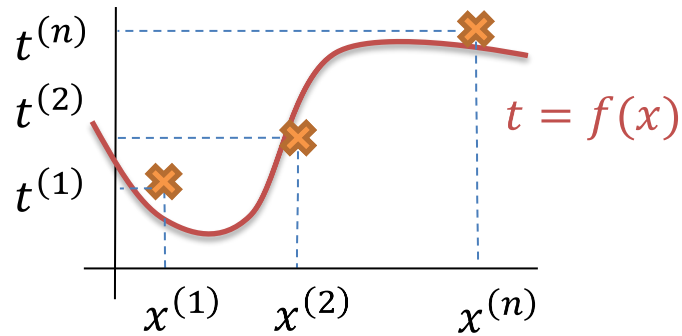
Regression and Classification

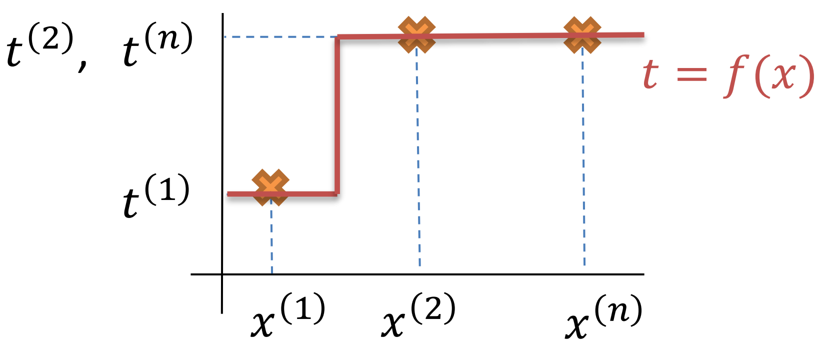
Minlie Huang
aihuang@tsinghua.edu.cn
Dept. of Computer Science and Technology
Tsinghua University
http://coai.cs.tsinghua.edu.cn/hml/

Regression and classification

Given: a set of data points $x^{(n)} \in R^m$ and the corresponding labels $t^{(n)} \in \Omega$: $\{(x^{(1)}, t^{(1)}), (x^{(2)}, t^{(2)}), \dots, (x^{(N)}, t^{(N)})\}$
Task: for a new data point x, predict its label t

- The goal is to find a mapping
$$f: R^m \to \Omega$$
- If Ω is a continuous set, this is called regression
- If Ω is a discrete set, this is called classification





Outline

- Linear regression
- Support vector regression
- Logistic regression
- Softmax regression

Linear regression

- $f(x)$ is linear

$$f(x) = w^T x + b$$

where $w \in R^n, b \in R$.

- The cost function can be chosen as the least square error

$$E = \sum_{n=1}^{N} \left(f\left(x^{(n)}\right) - t^{(n)}\right)^2 = \sum_{n=1}^{N} \left(w^T x^{(n)} + b - t^{(n)}\right)^2$$

- Find optimal w^* and b^* by minimizing the cost function

$$\left.\begin{aligned} \nabla_w E &= \sum_{n=1}^{N} \left(w^T x^{(n)} + b - t^{(n)}\right) x^{(n)} = 0 \\ \nabla_b E &= \sum_{n=1}^{N} \left(w^T x^{(n)} + b - t^{(n)}\right) = 0 \end{aligned}\right\} w^*, b^*$$

These equations have close-form solutions

Linear Regression

- Data Matrix: X=[$x^{(1)}$ ---Row vector

  $x^{(2)}$

  …

  $x^{(n)}$]

Y=[$t^{(1)}$; $t^{(2)}$; … $t^{(n)}$] is the output vector

Weight Matrix (parameters): W=$(X^TX)^{-1}X^TY$

Weight regularization

- To prevent overfitting, a regularization term is often incorporated into the cost function

$$J = \sum_{n=1}^{N} \left(f\left(x^{(n)}\right) - t^{(n)} \right)^2 + \frac{\lambda}{2} ||w||_2^2$$

where $\lambda > 0$ is a constant

Still have close-form solutions

- Again the optimal w^* and b^* are obtained by minimizing the cost function
- Regularization term
  - Encourages small values of weights
  - Improves generalization: often used in supervised learning systems, e.g., multi-layer perceptron (MLP).

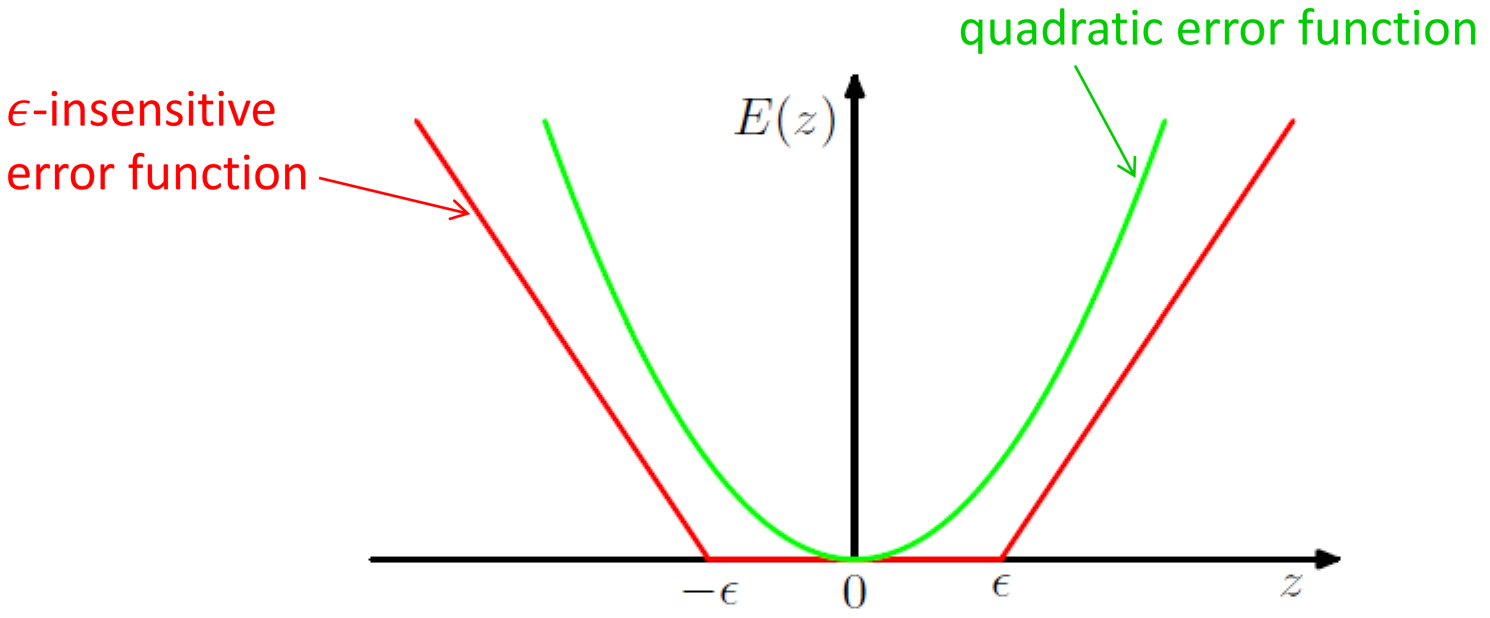
Outline

- Linear regression
- Support vector regression
- Logistic regression
- Softmax regression

ϵ-insensitive error function

- Define another error function

$$E_\epsilon(f(x)-t) = \begin{cases} 0, & \text{if } |f(x)-t| < \epsilon; \\ |f(x)-t| - \epsilon, & \text{otherwise} \end{cases}$$



Cost function

- The cost function

$$J = C\sum_{n=1}^{N} E_\epsilon\left(f\left(x^{(n)}\right) - t^{(n)}\right) + \frac{1}{2}||w||_2^2$$

where $C > 0$ is a constant ($C = 1/\lambda$)

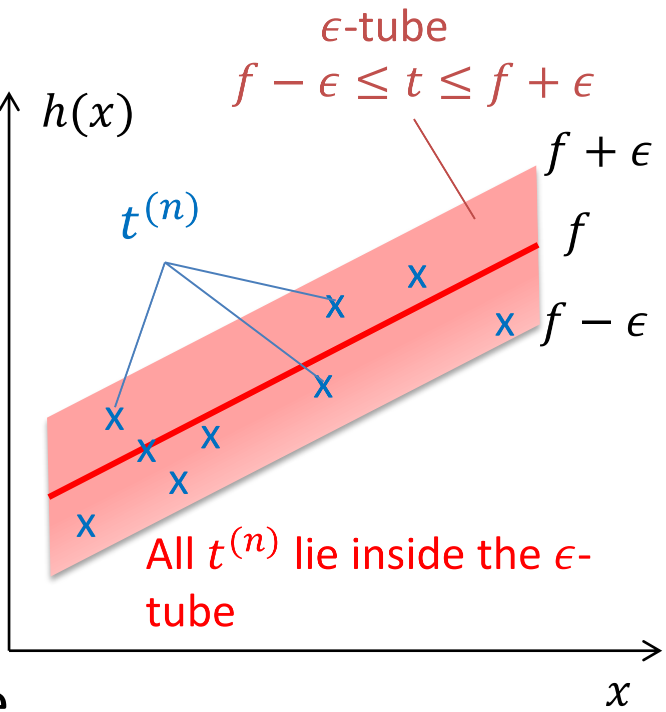
- If $\left|f\left(x^{(n)}\right) - t^{(n)}\right| \leq \epsilon$ for all n,

then minimizing E is equivalent to

$$\begin{array}{ll} \min_{w,b} & \frac{1}{2}||w||^2 \\ \text{s.t.} & \begin{cases} w^T x^{(n)} + b - t^{(n)} \leq \epsilon \\ -w^T x^{(n)} - b + t^{(n)} \leq \epsilon \end{cases} \end{array}$$

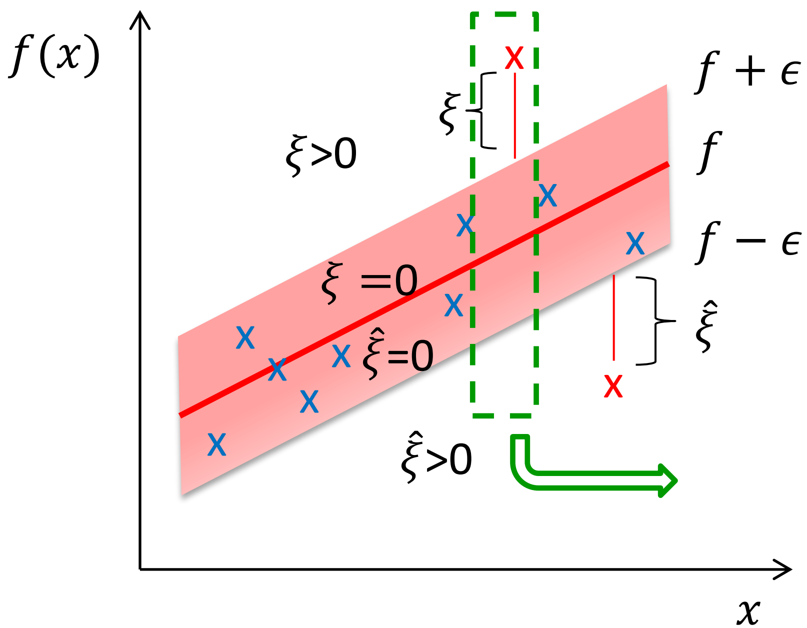
(Note $f(x) = w^T x + b$)

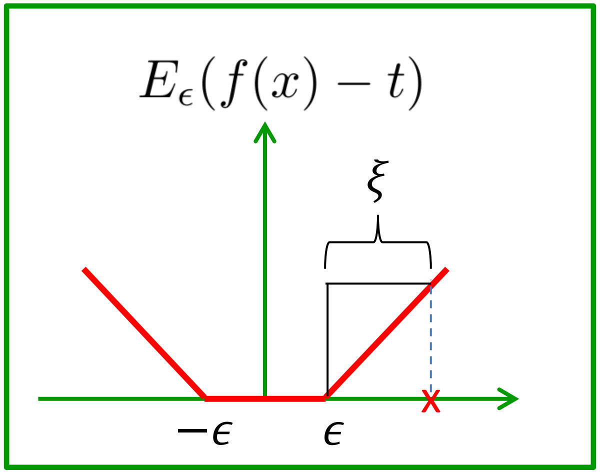
and this optimization problem is feasible



Slack variables

- In practice for some n, $\left|f\left(x^{(n)}\right) - t^{(n)}\right| > \epsilon$
- Introduce slack variables ξ and $\hat{\xi}$ to allow points to lie outside the tube: $f(x^{(n)}) - \epsilon - \hat{\xi}_n \leq t^{(n)} \leq f(x^{(n)}) + \epsilon + \xi_n$, where $\xi_n \geq 0$ and $\hat{\xi}_n \geq 0$





min $\sum_n E_\epsilon$ is equivalent to min $\sum_n(\xi_n + \hat{\xi}_n)$

New optimization problem

- The (primal) problem

cost function

$$\begin{array}{ll} \min_{w,b,\xi_n,\hat{\xi}_n} & J = C\sum_{n=1}^{N}(\xi_n + \hat{\xi}_n) + \frac{1}{2}\|w\|^2 \\ \text{s.t.} & \xi_n \geq t^{(n)} - f(x^{(n)}) - \epsilon \\ & \hat{\xi}_n \geq -t^{(n)} + f(x^{(n)}) - \epsilon \\ & \xi_n \geq 0, \hat{\xi}_n \geq 0 \end{array}$$

Similar to the soft margin SVM

- We can derive the dual problem with optimization theory, and then the kernel SVR

Outline

- Linear regression
- Support vector regression
- Logistic regression
- Softmax regression

Actually classification methods

Representation of class labels

- For classification, given $\{(x^{(1)}, t^{(1)}), \dots, (x^{(N)}, t^{(N)})\}$, the goal is to find a mapping from $x^{(n)}$ to $t^{(n)}$

$$f: R^m \to \Omega$$

where Ω is a discrete set

- $t^{(n)}$ can be a scalar or vector (in SVR, we assume it to be a scalar)

Suppose there are 5 classes in total

Scalar representation

$t^{(1)} = 3$

$t^{(3)} = 5$

Vector representation

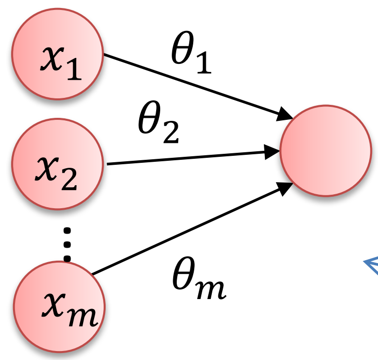
$t^{(1)} = (0, 0, 1, 0, 0)^T$

$t^{(3)} = (0, 0, 0, 0, 1)^T$

- 1-of-K representation
- Property: $t_k^{(n)} \in \{0,1\}$; $\sum_k t_k^{(n)} = 1$

Representation of class labels

- For 2-class problems, one 0-1 unit is enough for representing a label



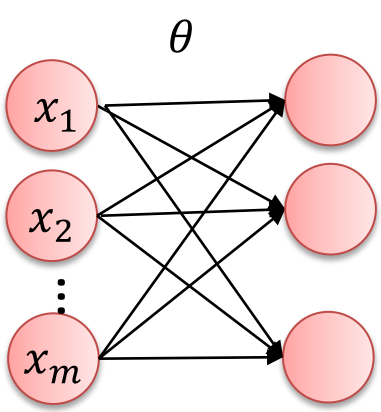
$t^{(1)}$	$t^{(2)}$	$t^{(3)}$	…
1	0	1	…

Therefore we use the scalar representation for simplicity

This representation is often used in logistic regression

Representation of class labels

- For K-class problems ($K > 2$),
  - One unit is enough for representing a label if it can take discrete values, e.g., $0, 1, 2, \dots, K$ $\leftarrow$ scalar representation
  - K 0-1 units can be also used to represent a label $\leftarrow$ vector representation



$t^{(1)}$	$t^{(2)}$	$t^{(3)}$	…
0	0	0	
1	0	1	…
0	1	0	

This has been used in least square regression

$$E = \sum_{n=1}^{N} E^{(n)}$$

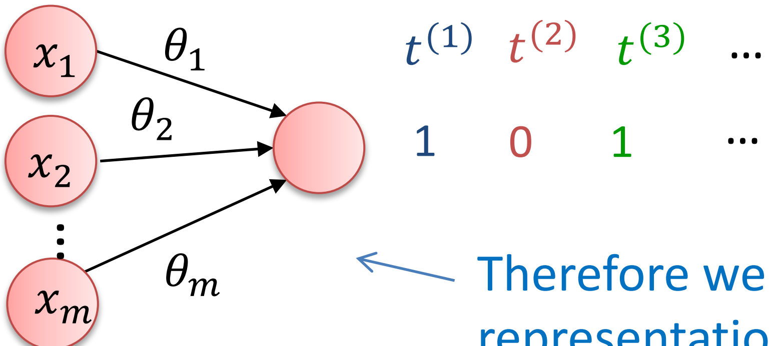
vector

$$E^{(n)} = \frac{1}{2} \| f(x^{(n)}) - t^{(n)} \|^2$$

This representation is often used in softmax regression

Logistic regression

- For two-class problems, one unit is enough to represent a label if it is constrained to take 0 or 1



Therefore we use the scalar representation for simplicity

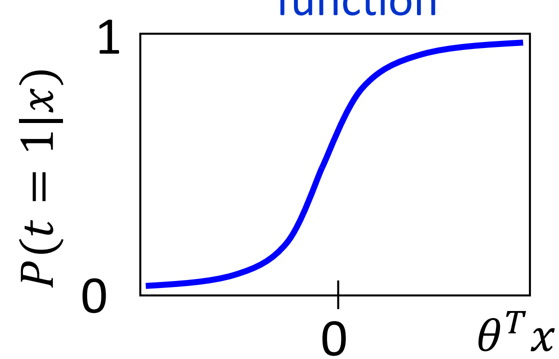
- We try to learn a function of the form

$$P(t=1|x) = \frac{1}{1+\exp(-\theta^T x)} \triangleq h(x)$$

$$P(t=0|x) = 1 - P(t=1|x) = 1 - h(x)$$

where x is sample and t is label

logistic sigmoid function



Logistic regression

$$P(t=1|x) = \frac{1}{1+\exp(-\theta^T x)} \triangleq h(x)$$

$$P(t=0|x) = 1 - P(t=1|x) = 1 - h(x)$$

where x is sample and t is label

- Our goal is to search for a value of θ so that the probability $P(t = 1|x) = h(x)$ is
  - large when x belongs to the 1 class and
  - small when x belongs to the 0 class (so that $P(t = 0|x)$ is large)
- $h(x)$ is not equivalent to $f: R^m \rightarrow \Omega$, but determines f; therefore we only need to learn $h(x)$

Cross-entropy error function

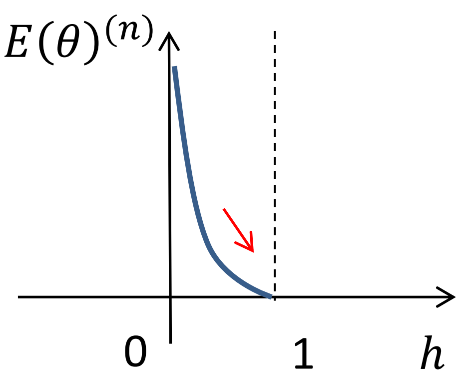
- For a set of training examples with binary labels $\{(x^{(n)}, t^{(n)}) : n = 1, \dots, N\}$ define the *cross-entropy* error function

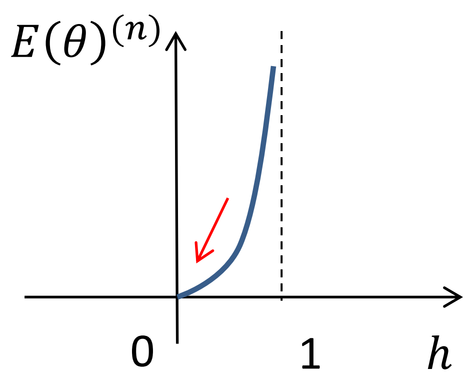
$$E(\theta) = -\sum_n \left(t^{(n)} \ln(h(x^{(n)})) + (1 - t^{(n)}) \ln(1 - h(x^{(n)}))\right)$$

$$E(\theta)^{(n)} = -t^{(n)} \ln(h(x^{(n)})) - (1 - t^{(n)}) \ln(1 - h(x^{(n)}))$$

> If $t^{(n)} = 1$, then
> $E(\theta)^{(n)} = -\ln(h)$

> If $t^{(n)}$ =0, then
> $E(\theta)^{(n)} = -\ln(1-h)$





Maximum likelihood formulation

- Why do we have this error function?
- For a dataset $\left\{\left(x^{(1)}, t^{(1)}\right), \ldots ,\ \left(x^{(N)}, t^{(N)}\right)\right\}$ where $t^{(n)} \in \{0,1\}$, the data likelihood function is

$$p(t^{(1)}, \ldots, t^{(N)} | \theta) = \Pi_{n=1}^{N} h(x^{(n)})^{t^{(n)}} (1 - h(x^{(n)}))^{1-t^{(n)}}$$

- Maximizing the likelihood is equivalent to minimizing

$$\begin{aligned} E(\theta) &= -\ln p(t^{(1)}, \ldots, t^{(N)}) \\ &= -\sum_{n=1}^{N} \left(t^{(n)} \ln h(x^{(n)}) + (1 - t^{(n)}) \ln(1 - h(x^{(n)})) \right) \end{aligned}$$

Training and testing

$$E(\theta) = -\sum_{n=1}^{N} \left(t^{(n)} \ln h(x^{(n)}) + (1 - t^{(n)}) \ln(1 - h(x^{(n)})) \right)$$

- Calculate the gradient (exercise)

$$\nabla E(\theta) = \sum_{n} x^{(n)} \left(h(x^{(n)}) - t^{(n)} \right)$$

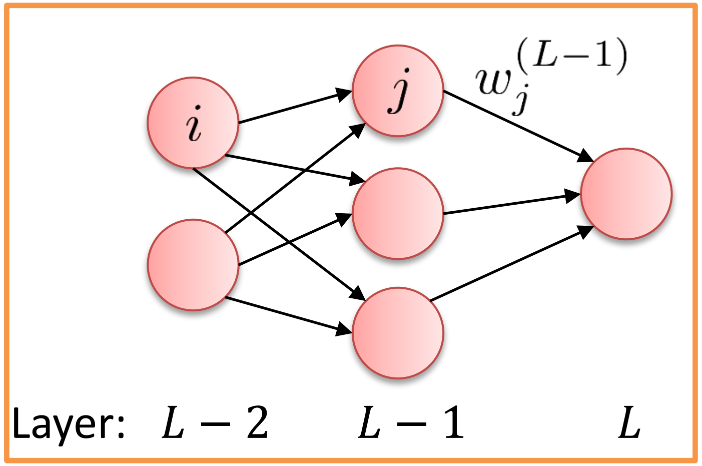
- As before, some regularization term can be incorporated into the cost function

$$J(\theta) = E(\theta) + \left|\left|\theta\right|\right|^2/2$$

- Training: learn θ to minimize the cost function
- Prediction: for a new input x, if $P(t = 1|x) > P(t = 0|x)$ then we label the example as a 1, and 0 otherwise

Apply to the multi-layer perceptron

- If logistic regression is used in the last layer of an MLP, then θ is replaced with $w^{(L-1)}$ and $b^{(L-1)}$ and the probabilistic function becomes



Output of the units on the (L-1)-th layer

$$y^{(L)} \triangleq h(\boxed{y^{(L-1)}}) = \frac{1}{1+\exp\left(-\left(w^{(L-1)}\right)^\top y^{(L-1)} - b^{(L-1)}\right)}$$

$$\nabla_{w^{(L-1)}} E = \sum_n y^{(L-1)(n)} \left(y^{(L)(n)} - t^{(n)}\right)$$

$$\nabla_{b^{(L-1)}} E = \sum_n \left(y^{(L)(n)} - t^{(n)}\right)$$

Outline

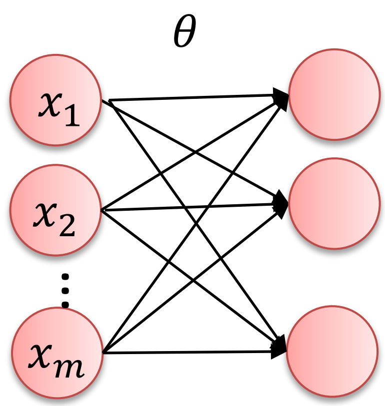
- Linear regression
- Support vector regression
- Logistic regression
- Softmax regression

Two Comments

- What we can create is what we truly understand!
- At what level of deep learners will you be?

Motivation

- For K-class problems ($K > 2$), K 0-1 units is used to represent a label



$t^{(1)}$	$t^{(2)}$	$t^{(3)}$	…
0	0	0	
1	0	1	…
0	1	0	

Note that $\sum_k t_k^{(n)} = 1$

- We try to learn a hypothesis $h(x)$ of the form

$$h(x) \triangleq \begin{bmatrix} P(t_1 = 1|x;\theta) \\ P(t_2 = 1|x;\theta) \\ \vdots \\ P(t_K = 1|x;\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x)} \begin{bmatrix} \exp(\theta^{(1)\top} x) \\ \exp(\theta^{(2)\top} x) \\ \vdots \\ \exp(\theta^{(K)\top} x) \end{bmatrix}$$

Motivation

Then $$h_k(x) = P(t_k = 1|x) = \frac{\exp(\theta^{(k)\top} x)}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x)}$$

- Given a test input x, estimate $P(t_k = 1|x)$ for each value of $k = 1, \ldots, K$.
- Goal: search for a value of θ so that the probability $P(t_k = 1|x)$ is
  - large when x belongs to the k class and
  - small when x belongs to other classes

where $$\theta = \left[\begin{array}{cccc} | & | & | & | \\ \theta^{(1)} & \theta^{(2)} & \ldots & \theta^{(K)} \\ | & | & | & | \end{array} \right].$$

- $h(x)$ is not equivalent to $f: R^m \to \Omega$, but determines f; therefore we only need to learn $h(x)$

Softmax function

$$h_k(x) = P(t_k = 1|x) = \frac{\exp(\theta^{(k)\top} x)}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x)}$$

Why?

- The following function is called *softmax* function

$$\psi(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)} = \frac{\exp(z_i)}{\exp(z_i) + \sum_{j \neq i} \exp(z_j)} \in (0, 1)$$

- If $z_i > z_j$ for all $j \neq i$
  - Then $\psi(z_i) > \psi(z_j)$ for all $j \neq i$ but it is smaller than 1
- If $z_i \gg z_j$ for all $j \neq i$,
  - then $\psi(z_i) \to 1$ and $\psi(z_j) \to 0$ for $j \neq i$.

Cross-entropy error function

- The data likelihood function is

$$p(t^{(1)}, \ldots, t^{(N)} | \theta) = \Pi_{n=1}^{N} \Pi_{k=1}^{K} P(t_k^{(n)} = 1 | x^{(n)})^{t_k^{(n)}}$$

- The cross-entropy error function is

$$\begin{aligned} E(\theta) &= -\ln p(t^{(1)}, \ldots, t^{(N)}) \\ &= -\sum_{n=1}^{N} \sum_{k=1}^{K} t_k^{(n)} \ln \frac{\exp(\theta^{(k)\top} x^{(n)})}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x^{(n)})} \end{aligned}$$

$$E(\theta) = \sum_{n=1}^{N} E^{(n)}(\theta), \qquad E^{(n)}(\theta) = -\sum_{i=1}^{K} t_i^{(n)} \ln h_i^{(n)}$$

where $h_i^{(n)} = P(t_i^{(n)} = 1 | x^{(n)}) = \frac{\exp(u_i^{(n)})}{\sum_{j=1}^{K} \exp(u_j^{(n)})}, \quad u_k^{(n)} = \theta^{(k)\top} x^{(n)}$

Calculate the gradient

$$E(\theta) = \sum_{n=1}^{N} E^{(n)}(\theta), \qquad E^{(n)}(\theta) = -\sum_{i=1}^{K} t_i^{(n)} \ln h_i^{(n)}$$

where $\quad h_i^{(n)} = P(t_i^{(n)} = 1 | x^{(n)}) = \dfrac{\exp(u_i^{(n)})}{\sum_{j=1}^{K} \exp(u_j^{(n)})}, \quad u_k^{(n)} = \theta^{(k)\top} x^{(n)}$

$$\frac{\partial E^{(n)}}{\partial \theta^{(k)}} = \frac{\partial E^{(n)}}{\partial u^{(k)}} \frac{\partial u^{(k)}}{\partial \theta^{(k)}} = \sum_{i=1}^{K} \frac{\partial E^{(n)}}{\partial h_i^{(n)}} \frac{\partial h_i^{(n)}}{\partial u_k^{(n)}} \frac{\partial u_k^{(n)}}{\partial \theta^{(k)}}$$

Local sensitivity or local gradient

$$\frac{\partial E^{(n)}}{\partial h_i^{(n)}} = -t_i^{(n)} \frac{1}{h_i^{(n)}}$$

?

$$\frac{\partial u_k^{(n)}}{\partial \theta^{(k)}} = x^{(n)}$$

Calculate the gradient

$$E(\theta) = \sum_{n=1}^{N} E^{(n)}(\theta), \qquad E^{(n)}(\theta) = -\sum_{i=1}^{K} t_i^{(n)} \ln h_i^{(n)}$$

where $h_i^{(n)} = P(t_i^{(n)} = 1 | x^{(n)}) = \dfrac{\exp(u_i^{(n)})}{\sum_{j=1}^{K} \exp(u_j^{(n)})}, \quad u_k^{(n)} = \theta^{(k)\top} x^{(n)}$

If $k \neq i$, u_k appears only in the denominator

$$\frac{\partial h_i^{(n)}}{\partial u_k^{(n)}} = -\frac{\exp(u_i^{(n)}) \exp(u_k^{(n)})}{\left(\sum_j \exp(u_j^{(n)})\right)^2} = -h_k^{(n)} h_i^{(n)}$$

If $k = i$, u_k appears in both numerator and denominator

$$\frac{\partial h_i^{(n)}}{\partial u_k^{(n)}} = \frac{\exp(u_k^{(n)})}{\sum_j \exp(u_j^{(n)})} - \frac{\left(\exp(u_k^{(n)})\right)^2}{\left(\sum_j \exp(u_j^{(n)})\right)^2} = h_k^{(n)}(1 - h_k^{(n)})$$

Therefore $\dfrac{\partial h_i^{(n)}}{\partial u_k^{(n)}} = h_i^{(n)}(\Delta_{i,k} - h_k^{(n)})$ where $\Delta_{i,k} = \begin{cases} 1, & \text{if } i = k \\ 0, & \text{else.} \end{cases}$

Calculate the gradient

$$E(\theta) = \sum_{n=1}^{N} E^{(n)}(\theta), \qquad E^{(n)}(\theta) = -\sum_{i=1}^{K} t_i^{(n)} \ln h_i^{(n)}$$

where $\quad h_i^{(n)} = P(t_i^{(n)} = 1 | x^{(n)}) = \dfrac{\exp(u_i^{(n)})}{\sum_{j=1}^{K} \exp(u_j^{(n)})}, \quad u_k^{(n)} = \theta^{(k)\top} x^{(n)}$

$$\begin{aligned}
\frac{\partial E^{(n)}}{\partial \theta^{(k)}} &= \sum_{i=1}^{K} \frac{\partial E^{(n)}}{\partial h_i^{(n)}} \frac{\partial h_i^{(n)}}{\partial u_k^{(n)}} \frac{\partial u_k^{(n)}}{\partial \theta^{(k)}} \\
&= \sum_{i=1}^{K} \left(-t_i^{(n)} \frac{1}{h_i^{(n)}} \right) \left(h_i^{(n)} (\Delta_{i,k} - h_k^{(n)}) \right) \left(x^{(n)} \right) \\
&= -\left(\sum_{i=1}^{K} t_i^{(n)} \Delta_{i,k} - \underbrace{\sum_{i=1}^{K} t_i^{(n)}}_{=1} h_k^{(n)} \right) x^{(n)} \\
&= -\left(t_k^{(n)} - h_k^{(n)} \right) x^{(n)}
\end{aligned}$$

Calculate the gradient

$$E(\theta) = \sum_{n=1}^{N} E^{(n)}(\theta), \qquad E^{(n)}(\theta) = -\sum_{i=1}^{K} t_i^{(n)} \ln h_i^{(n)}$$

where $h_i^{(n)} = P(t_i^{(n)} = 1 | x^{(n)}) = \dfrac{\exp(u_i^{(n)})}{\sum_{j=1}^{K} \exp(u_j^{(n)})}, \quad u_k^{(n)} = \theta^{(k)\top} x^{(n)}$

$$\frac{\partial E^{(n)}}{\partial \theta^{(k)}} = \delta_k^{(n)} x^{(n)}, \quad \text{where } \delta_k^{(n)} \triangleq \frac{\partial E^{(n)}}{\partial u_k} = -\left(t_k^{(n)} - h_k^{(n)}\right)$$

is the local sensitivity. Note the sign is inconsistent with the slides about MLP

The overall gradient

$$\nabla_{\theta^{(k)}} E(\theta) = \sum_{n=1}^{N} \frac{\partial E^{(n)}}{\partial \theta^{(k)}} = -\sum_{n=1}^{N} \left(t_k^{(n)} - h_k^{(n)}\right) x^{(n)}$$

$$= -\sum_{n=1}^{N} \left(t_k^{(n)} - P(t_k^{(n)} = 1 | x^{(n)})\right) x^{(n)}$$

Training and testing

- Calculate the gradient of the cross-entropy error function

$$\nabla_{\theta^{(k)}} E(\theta) = -\sum_{n=1}^{N} \left(t_k^{(n)} - P(t_k^{(n)} = 1 | x^{(n)}; \theta) \right) x^{(n)}$$

- As before, some regularization term can be incorporated into the cost function

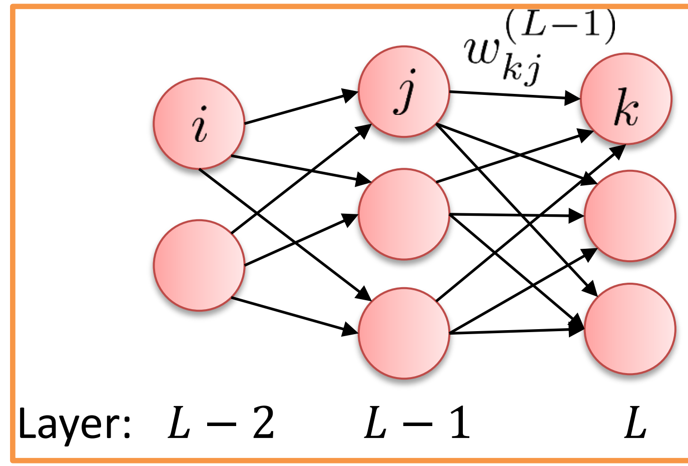
$$J(\theta) = E(\theta) + \left|\left|\theta\right|\right|^2 / 2$$

- Training: minimize the cost function with gradient $\nabla J(\theta)$
- Prediction: find the maximum $P(t_k = 1|x)$ among k for a new input x

$$P(t_k = 1|x) = \frac{\exp(\theta^{(k)\top} x)}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x)}$$

Apply to the multi-layer perceptron

- If logistic regression is used in the last layer of an MLP, then θ is replaced with $w^{(L-1)}$ and $b^{(L-1)}$ and the probabilistic function becomes



Output of the units on the (L-1)-th layer

$$y_k^{(L)} \triangleq P(t_k = 1 | \boxed{y^{(L-1)}}) = \frac{\exp(w_k^{(L-1)\top} y^{(L-1)} + b_k^{(L-1)})}{\sum_{j=1}^{K} \exp(w_j^{(L-1)\top} y^{(L-1)} + b_j^{(L-1)})}$$

$$\nabla_{w_k^{(L-1)}} E = -\sum_{n=1}^{N} \left(t_k^{(n)} - y_k^{(L)(n)} \right) y^{(L-1)(n)}$$

$$\nabla_{b_k^{(L-1)}} E = -\sum_{n=1}^{N} \left(t_k^{(n)} - y_k^{(L)(n)}) \right)$$

Softmax is over-parameterized

- The hypothesis

$$h_k(x) = P(t_k = 1|x) = \frac{\exp(\theta^{(k)\top} x)}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x)} = \frac{\exp((\theta^{(k)} - \phi)^\top x)}{\sum_{j=1}^{K} \exp((\theta^{(j)} - \phi)^\top x)}$$

  Then the new parameters $\hat{\theta}^{(k)} \equiv \theta^{(k)} - \phi$ will result in the same prediction

- Minimizing the cross-entropy function has infinite number of solutions since

$$E(\theta) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_k^{(n)} \ln \frac{\exp(\theta^{(k)\top} x^{(n)})}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x^{(n)})} = E(\theta - \Phi)$$

  where $\Phi = (\phi, ..., \phi)$

Relationship between softmax regression and logistic regression

Let $K = 2$ in softmax

Sigmoid function

- The hypotheses

$$h_1(x) = P(t_1 = 1|x) = \frac{\exp(\theta^{(1)\top}x)}{\exp(\theta^{(1)\top}x) + \exp(\theta^{(2)\top}x)} = \sigma(\theta^{(1)} - \theta^{(2)})$$

$$h_2(x) = P(t_2 = 1|x) = \frac{\exp(\theta^{(2)\top}x)}{\exp(\theta^{(1)\top}x) + \exp(\theta^{(2)\top}x)} = 1 - \sigma(\theta^{(1)} - \theta^{(2)})$$

The same as in the two-unit version of the logistic regression if we define a new variable $\hat{\theta} = \theta^{(1)} - \theta^{(2)}$.

- The error function for each sample

$$E^{(n)}(\theta) = -t_1^{(n)} \ln h_1^{(n)} - t_2^{(n)} \ln h_2^{(n)} = -t_1^{(n)} \ln h_1^{(n)} - (1 - t_1^{(n)}) \ln(1 - h_1^{(n)})$$

The same as in the logistic regression

Logistic regression is a **special case** of softmax regression

Summary

- Linear regression
  - Least square error function
  - weight regularization
- Support vector regression
  - ϵ-insensitive error function
- Logistic regression
  - Logistic sigmoid function
  - Cross-entropy error function
- Softmax regression
  - Softmax function
  - Cross-entropy error function