Ch02. Logistic Regression

Hwanjo Yu

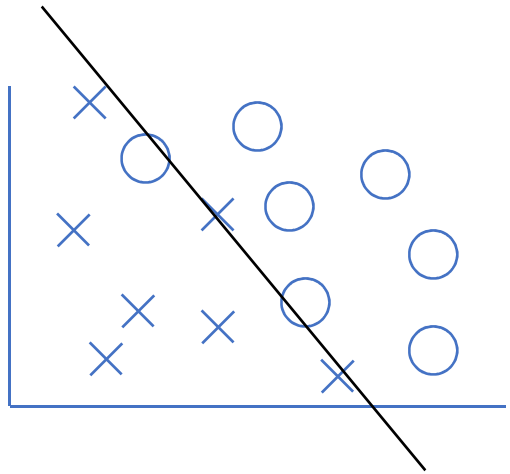
POSTECH

http://hwanjoyu.org

Binary Classification

Examples

- Medical diagnosis: Disease or no disease
- Cancer diagnosis: Benign or malignant
- Image classification: Cat or dog
- Image classification: Face or non-face
- Sentiment analysis: Positive or negative
- Name entity recognition: Name or not
- Speech or audio: Voice or not

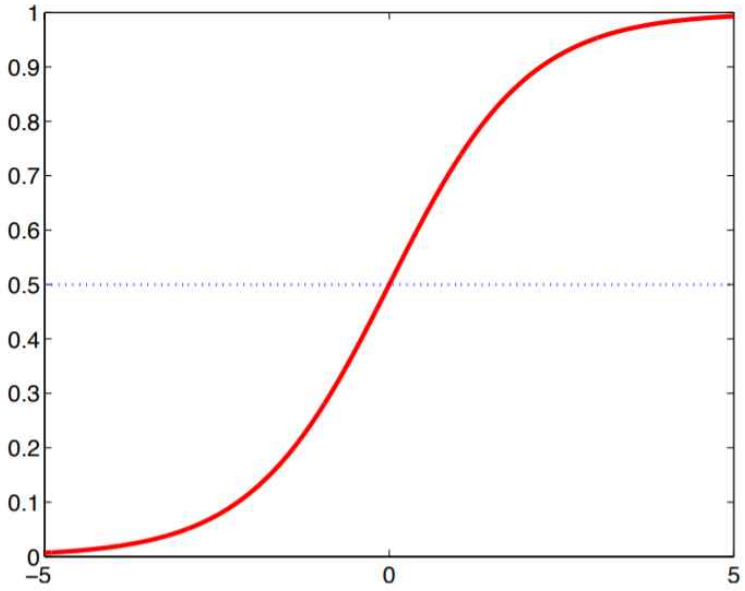


Logistic Function

Logistic function (or sigmoid function) $\sigma(\xi)$:

$$\sigma(\xi) = \frac{1}{1 + e^{-\xi}}.$$

- $\sigma(\xi) \to 0$ as $\xi \to -\infty$.
- $\sigma(\xi) \to 1$ as $\xi \to \infty$.
- $\sigma(-\xi) = 1 - \sigma(\xi)$.
- $\frac{d}{d\xi}[\sigma(\xi)] = \sigma(\xi)\sigma(-\xi)$.



Maximum Likelihood Formulation

Logistic Regression

Predict a binary output $y_n \in \{0,1\}$ from an input $\mathbf{x}_n$.

The logistic regression models the input-output by a **conditional Bernoulli distribution**

$$\mathbb{E}[y_n|\mathbf{x}_n] = \mathbb{P}(y_n = 1|\mathbf{x}_n) = \sigma\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n\right),$$

where

$$\sigma(\xi) = \frac{1}{1+e^{-\xi}} = \frac{e^{\xi}}{1+e^{\xi}}.$$

Discriminative model: Directly model $p(y|\mathbf{x})$.

Logistic Regression: MLE

Given $\{(\mathbf{x}_n, y_n)|n = 1, \dots, N\}$, the likelihood is given by

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

$$= \prod_{n=1}^{N} \sigma\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n\right)^{y_n}\left(1 - \sigma\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n\right)\right)^{1-y_n}$$

Then, log-likelihood function is given by

$$\mathcal{L} = \sum_{n=1}^{N} \log p(y_n|\mathbf{x}_n) = \sum_{n=1}^{N} \{y_n \log \hat{y}_n + (1 - y_n)\log(1 - \hat{y}_n)\},$$

where $\hat{y}_n = \sigma\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n\right)$.

- This is a nonlinear function of $\mathbf{w}$ whose maximum cannot be computed in a closed form.
- Iterative re-weighted least squares (IRLS) is a popular algorithm, derived from Newton's method.

Newton’s Method

Mathematical Preliminaries

- Gradient
- Hessian matrix
- Gradient descent/ascent
- Newton's method

Gradient

- Consider a real-valued function $f(\mathbf{x})$, that takes a real-valued vector $\mathbf{x} \in \mathbb{R}^D$ as an input,

$$f(\mathbf{x}): \mathbb{R}^D \longrightarrow \mathbb{R}.$$

- The gradient of $f(\mathbf{x})$ is defined by

$$\nabla f(\mathbf{x}) = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \left[\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_D}\right]^{\mathrm{T}}.$$

Hessian Matrix

The Hessian matrix $\mathbf{H}$ is defined as the symmetric matrix with the (i,j)-element $\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}$,

$$\mathbf{H} = \nabla^2 f(\mathbf{x}) = \left[\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}\right] = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_D} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_D} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_D \partial x_1} & \frac{\partial^2 f}{\partial x_D \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_D^2} \end{bmatrix} = \frac{\partial}{\partial \mathbf{x}}\left[\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\right]^{\mathrm{T}}$$

$$= \frac{\partial}{\partial \mathbf{x}}[\nabla f(\mathbf{x})]^{\mathrm{T}}$$

Gradient Descent/Ascent

- The gradient descent/ascent learning is a simple (first-order) iterative method for minimization/maximization
- Gradient descent: iterative minimization

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \eta \left(\frac{\partial \mathcal{J}}{\partial \mathbf{w}} \right).$$

- Gradient ascent: iterative maximization

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \eta \left(\frac{\partial \mathcal{L}}{\partial \mathbf{w}} \right).$$

- Learning rate: $\eta > 0$

Taylor series expansion

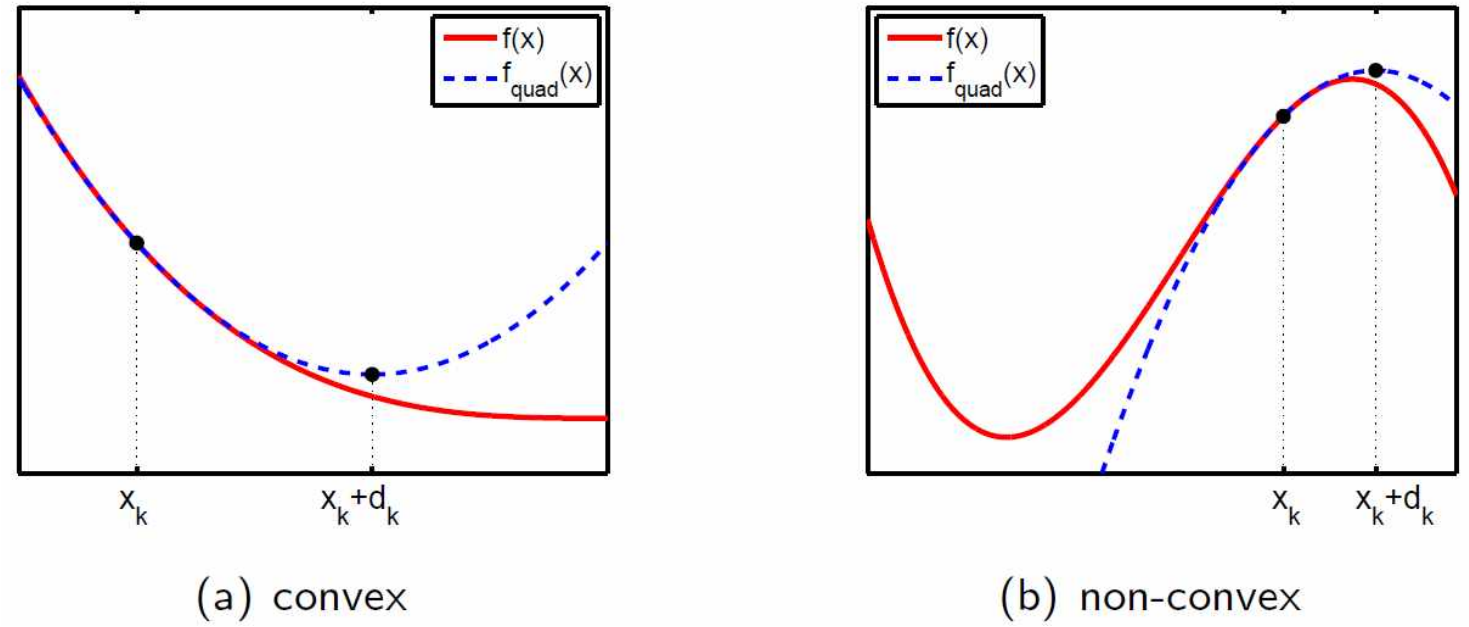
- Approximate an arbitrary function with a polynomial form.
- $f(x) = C_0 + C_1(x-a) + C_2(x-a)^2 + C_3(x-a)^3 + C_4(x-a)^4 + \cdots$
  - $f(a) = C_0$
  - $f'(x) = C_1 + 2C_2(x-a) + 3C_3(x-a)^2 + 4C_4(x-a)^3 + \cdots$
  - $f'(a) = C_1$
  - $f''(x) = 2*1C_2 + 3*2C_3(x-a)^1 + 4*3C_4(x-a)^2 + \cdots$
  - $f''(a) = 2C_2$
  - $f'''(x) = 3*2*1C_3 + 4*3*2C_4(x-a)^1 + \cdots$
  - $f'''(a) = 3*2*1C_2$
- $f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots$

$$= \sum_{n=1}^{\infty} \frac{1}{n!} f^{(n)}(a)(x-a)^n$$

- Must $f(x)$ be "multiply differentiable" at $x = a$ to approximate $f(x)$ using Taylor

Newton's method

The basic idea of Newton's method is to optimize the quadratic approximation of the objective function $\mathcal{J}(\mathbf{w})$ around the current point $\mathbf{w}^{(k)}$.



[from Murphy's book]

The second-order Taylor series expansion of $\mathcal{J}(\mathbf{w})$ at the current point $\mathbf{w}^{(k)}$ gives

$$\mathcal{J}_{\text{quad}}(\mathbf{w}) = \mathcal{J}(\mathbf{w}^{(k)}) + [\nabla\mathcal{J}(\mathbf{w}^{(k)})]^{\mathrm{T}}(\mathbf{w} - \mathbf{w}^{(k)}) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^{(k)})^{\mathrm{T}}\nabla^2\mathcal{J}(\mathbf{w}^{(k)})(\mathbf{w} - \mathbf{w}^{(k)}).$$

where $\nabla^2\mathcal{J}(\mathbf{w}^{(k)})$ is the Hessian of $\mathcal{J}(\mathbf{w})$ evaluated at $\mathbf{w} = \mathbf{w}^{(k)}$.

Newton's method

Differentiate $\mathcal{J}_{\text{quad}}(\mathbf{w})$ w.r.t. $\mathbf{w}$ and set it equal 0, which leads to

$$\nabla\mathcal{J}\left(\mathbf{w}^{(k)}\right) + \nabla^2\mathcal{J}\left(\mathbf{w}^{(k)}\right)\mathbf{w} - \nabla^2\mathcal{J}\left(\mathbf{w}^{(k)}\right)\mathbf{w}^{(k)} = 0.$$

Thus, we have

$$\mathbf{w} = \mathbf{w}^{(k)} - \left[\nabla^2\mathcal{J}\left(\mathbf{w}^{(k)}\right)\right]^{-1}\nabla\mathcal{J}\left(\mathbf{w}^{(k)}\right).$$

Hence the Newton's method is of the form

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \left[\nabla^2\mathcal{J}\left(\mathbf{w}^{(k)}\right)\right]^{-1}\nabla\mathcal{J}\left(\mathbf{w}^{(k)}\right).$$

Logistic Regression: Algorithms

Logistic Regression: Gradient Ascent

The gradient ascent learning has the form

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} + \eta \left(\frac{\partial \mathcal{L}}{\partial \mathbf{w}} \right),$$

where the log-likelihood

$$\mathcal{L} = \sum_{n=1}^{N} \log p(y_n | \mathbf{x}_n) = \sum_{n=1}^{N} \{ y_n \log \hat{y}_n + (1 - y_n) \log(1 - \hat{y}_n) \},$$

where $\hat{y}_n = \sigma(\mathbf{w}^{\mathrm{T}} \mathbf{x}_n)$.

Then, the gradient

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \sum_{n=1}^{N} \left[y_n \frac{\hat{y}_n'}{\hat{y}_n} \mathbf{x}_n + (1 - y_n) \frac{-\hat{y}_n'}{1 - \hat{y}_n} \mathbf{x}_n \right] = \sum_{n=1}^{N} \left[y_n \frac{\hat{y}_n (1 - \hat{y}_n)}{\hat{y}_n} - (1 - y_n) \frac{\hat{y}_n (1 - \hat{y}_n)}{1 - \hat{y}_n} \right] \mathbf{x}_n$$

$$= \sum_{n=1}^{N} [y_n (1 - \hat{y}_n) - (1 - y_n) \hat{y}_n] \mathbf{x}_n = \sum_{n=1}^{N} [y_n - \hat{y}_n] \mathbf{x}_n .$$

Hence, the gradient ascent update rule for $\mathbf{w}$ is

$$\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} + \eta \sum_{n=1}^{N} [y_n - \sigma(\mathbf{w}^{\mathrm{T}} \mathbf{x}_n)] \mathbf{x}_n .$$

Logistic Regression: Hessian

Calculate the Hessian:

$$\nabla^2 \mathcal{L} = \frac{\partial}{\partial \mathbf{w}} [\nabla \mathcal{L}]^{\mathrm{T}} = \frac{\partial}{\partial \mathbf{w}} \left[\sum_{n=1}^{N} (y_n - \hat{y}_n) \mathbf{x}_n^{\mathrm{T}} \right] = \sum_{n=1}^{N} -\hat{y}_n (1 - \hat{y}_n) \mathbf{x}_n \mathbf{x}_n^{\mathrm{T}} .$$

We set the objective function $\mathcal{J}(\mathbf{w})$ as the negative log-likelihood (cross-entropy error):

$$\mathcal{J}(\mathbf{w}) = -\mathcal{L}(\mathbf{w}) = -\sum_{n=1}^{N} \{ y_n \log \hat{y}_n + (1 - y_n) \log(1 - \hat{y}_n) \} .$$

Thus, the gradient and the Hessian are:

$$\nabla \mathcal{J}(\mathbf{w}) = -\sum_{n=1}^{N} (y_n - \hat{y}_n) \mathbf{x}_n ,$$

$$\nabla^2 \mathcal{J}(\mathbf{w}) = \sum_{n=1}^{N} \hat{y}_n (1 - \hat{y}_n) \mathbf{x}_n \mathbf{x}_n^{\mathrm{T}} .$$

Note that the Hessian matrix $\nabla^2 \mathcal{J}(\mathbf{w}) = \sum_{n=1}^{N} \hat{y}_n (1 - \hat{y}_n) \mathbf{x}_n \mathbf{x}_n^{\mathrm{T}}$ is positive definite since $\sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^{\mathrm{T}}$ is positive definite and $\hat{y}_n (1 - \hat{y}_n) \geq 0, \forall n$.

Logistic Regression: IRLS

Newton's update has the form

$$\Delta\mathbf{w} = -\underbrace{\left[\sum_{n=1}^{N} \hat{y}_n(1-\hat{y}_n)\mathbf{x}_n\mathbf{x}_n^{\mathrm{T}}\right]^{-1}}_{[\nabla^2 J(\mathbf{w})]^{-1}} \underbrace{\left[-\sum_{n=1}^{N}(y_n-\hat{y}_n)\mathbf{x}_n^{\mathrm{T}}\right]}_{\nabla J(\mathbf{w})},$$

Newton's update reduces to iterative re-weighted least squares (IRLS):

$$\Delta\mathbf{w} = \left(\mathbf{X}\mathbf{S}\mathbf{X}^{\mathrm{T}}\right)^{-1}\mathbf{X}\mathbf{S}\mathbf{b},$$

where

$$\mathbf{S} = \begin{bmatrix} \hat{y}_1(1-\hat{y}_1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \hat{y}_N(1-\hat{y}_N) \end{bmatrix},$$

$$\mathbf{b} = \begin{bmatrix} \dfrac{y_1-\hat{y}_1}{\hat{y}_1(1-\hat{y}_1)} \\ \vdots \\ \dfrac{y_N-\hat{y}_N}{\hat{y}_N(1-\hat{y}_N)} \end{bmatrix}.$$

Logistic Regression: IRLS

Alternatively we write

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \left(\mathbf{X}\mathbf{S}_k\mathbf{X}^{\mathrm{T}}\right)^{-1}\mathbf{X}\mathbf{S}_k\mathbf{b}_k,$$

Where $\mathbf{S}_k$ and $\mathbf{b}_k$ are computed using $\mathbf{w}_k$.

That is,

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \left(\mathbf{X}\mathbf{S}_k\mathbf{X}^{\mathrm{T}}\right)^{-1}\mathbf{X}\mathbf{S}_k\mathbf{b}_k = \left(\mathbf{X}\mathbf{S}_k\mathbf{X}^{\mathrm{T}}\right)^{-1}\left[\left(\mathbf{X}\mathbf{S}_k\mathbf{X}^{\mathrm{T}}\right)\mathbf{w}_k + \mathbf{X}\mathbf{S}_k\mathbf{b}_k\right]$$
$$= \left(\mathbf{X}\mathbf{S}_k\mathbf{X}^{\mathrm{T}}\right)^{-1}\mathbf{X}\mathbf{S}_k[\mathbf{X}^{\mathrm{T}}\mathbf{w}_k + \mathbf{b}_k].$$

Algorithm: IRLS

Input: $\{(\mathbf{x}_n, y_n)|n = 1, \ldots, N\}$

1. Initialize $\mathbf{w} = 0$ and $w_0 = \log(\bar{y}/(1-\bar{y}))$
2. repeat
3. for$n = 1, \ldots, N$ do
4. Compute $\eta_n = \mathbf{w}^{\mathrm{T}}\mathbf{x}_n + w_0$
5. Compute $\hat{y}_n = \sigma(\eta_n)$
6. Compute $s_n = \hat{y}_n(1-\hat{y}_n)$
7. Compute $z_n = \eta_n + (y_n - \hat{y}_n)/s_n$
8. end for
9. Construct $\mathbf{S} = \mathrm{diag}(s_{1:N})$
10. Update $\mathbf{w} = \left(\mathbf{X}\mathbf{S}\mathbf{X}^{\mathrm{T}}\right)^{-1}\mathbf{X}\mathbf{S}\mathbf{z}$
11. until convergence
12. return $\mathbf{w}$

Softmax Regression for Multiclass Classification

- Model input-output by a softmax transformation of logits $\theta_k = \mathbf{w}_k^{\mathrm{T}}\mathbf{x}_n$:

$$p(y_n = k|\mathbf{x}_n) = \mathrm{softmax}\left(\theta_k = \mathbf{w}_k^{\mathrm{T}}\mathbf{x}_n\right) = \frac{\exp\{\mathbf{w}_k^{\mathrm{T}}\mathbf{x}_n\}}{\sum_{j=1}^{K}\exp\{\mathbf{w}_j^{\mathrm{T}}\mathbf{x}_n\}}$$

- Given $\mathbf{Y} \in \mathbb{R}^{K\times N}$ (each column $\mathbf{y}_n \in \mathbb{R}^K$ follows the 1-of-K coding) and $\mathbf{X} \in \mathbb{R}^{D\times N}$, the likelihood is given by

$$p(\mathbf{Y}|\mathbf{X}, \mathbf{w}_1, \ldots, \mathbf{w}_K) = \prod_{n=1}^{N}\prod_{k=1}^{K} p(y_n = k|\mathbf{x}_n)^{Y_{k,n}} \ .$$

- The log-likelihood is given by

$$\mathcal{L} = \sum_{n=1}^{N}\sum_{k=1}^{K} Y_{k,n}\log[p(y_n = k|\mathbf{x}_n)] \ .$$

- One can apply Newton's update to derive IRLS, as in logistic regression.

Cross Entropy Error

- Cross entropy between two probability distributions p and q is $H(p;q) = -\sum_x p(x)\log q(x)$ (refer to Wikipedia)

- In the case of logistic regression, we have $p \in \{y, 1-y\}, q \in \{\hat{y}, 1-\hat{y}\}$. Then, the **cross entropy error** is calculated as

$$\mathcal{E} = \sum_{n=1}^{N} [-y_n \log \hat{y}_n - (1-y_n)\log(1-\hat{y}_n)].$$

- In the case of softmax regression, we have $p \in \{y_1, \dots y_K\}, q \in \{\hat{y}_1, \dots, \hat{y}_K\}$. Then, the **cross entropy error** is calculated as

$$\mathcal{E} = \sum_{n=1}^{N}\sum_{k=1}^{K} \left[-y_{k,n} \log \hat{y}_{k,n}\right].$$

- Note, the cross entropy error $\mathcal{E}$ is equal to the negative log-likelihood $-\mathcal{L}(\mathbf{w})$.

Cross Entropy Error

- $y = \begin{bmatrix} \text{tiger} \\ \text{lion} \\ \text{cat} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \hat{y} = \begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix}, \mathcal{E} = -\log 0.7 = 0.36$
- $y = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \hat{y} = \begin{bmatrix} 0.5 \\ 0.3 \\ 0.2 \end{bmatrix}, \mathcal{E} = -\log 0.5 = 0.69$

Multi-Label Learning

- $y = \begin{bmatrix} \text{dog} \\ \text{cat} \\ \text{sky} \\ \text{sand} \\ \text{lake} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$

Labels: dog, sand, sky

- The error function is given by

$$\mathcal{J} = \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} \underbrace{\ell(y_{k,n}, \hat{y}_{k,n})}_{\text{logistic loss}}$$

$$= \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} \left[-y_{k,n} \log \hat{y}_{k,n} - (1 - y_{k,n}) \log(1 - \hat{y}_{k,n}) \right] ,$$

- where $\hat{y}_{k,n}$ is the output of the kth logistic regression model,

$$\hat{y}_{k,n} = \sigma\left(\mathbf{w}_k^{\mathrm{T}} \mathbf{x}_n\right)$$

Multi-Label Learning

- For single-label learning, $y = \begin{bmatrix} \text{tiger} \\ \text{lion} \\ \text{cat} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \hat{y} = \begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix}, \mathcal{E} = -\log 0.7 = 0.36$
- For multi-label learning, $y = \begin{bmatrix} \text{tiger} \\ \text{lion} \\ \text{cat} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \hat{y} = \begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix}, \mathcal{E} = -\log 0.7 - \log 0.8 - \log 0.9 = 0.69$
- When having '?' labels such as $[\mathbf{Y} \in \mathbb{R}^{K \times N}] = \begin{bmatrix} 1 & 0 & ? & \dots \\ 0 & 0 & 1 & \dots \\ 1 & ? & 0 & \dots \\ 1 & 1 & ? & \dots \\ 0 & 1 & ? & \dots \end{bmatrix}$,
- k runs only for indices associated with 0 or 1 in $\mathcal{J} = \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} \underbrace{\ell(y_{k,n}, \hat{y}_{k,n})}_{\text{logistic loss}}$.