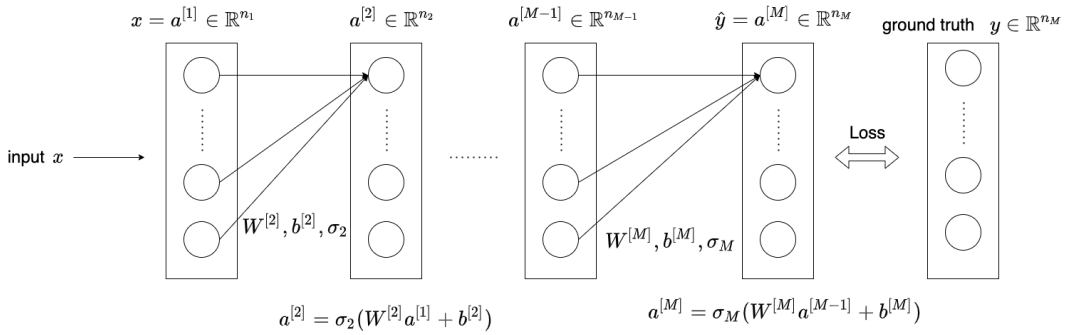
Contents

1 Mathematics behind Deep Neural Network

1.1 Neural Network

A *Neural network* is a structure that defines a specific connection among neurons. A neural network consists of *layers* (the following rectangles). A layer consists of *neurons* (circles). Neurons are just numbers. The connections (arrow lines) between neurons are some kind of computations, they are called *model weights*.



To construct a neural network, construct layers first. Each layer can be regarded as a real vector. In the image above, the first layer is $a^{[1]}$, it's a real vector has n_1 dimension; the second layer is $a^{[2]}$, a real vector has n_2 dimension, and so on, where each $n_i \in \mathbb{N}$.

Second, we make connections among layers. Because layers are regarded as vectors, it's quite intuitive to define connections between them by matrix computation.

For example, in the above image, the connection between the first two layers, $a^{[1]}$ and $a^{[2]}$, can be defined by a matrix $W^{[2]}$, a bias vector $b^{[2]}$ and an activation function σ_2. That is,

$$a^{[2]} = \sigma_2(W^{[2]}a^{[1]} + b^{[2]})$$

where $W^{[2]} \in \mathbb{R}^{n_2 \times n_1}$, $b^{[2]} \in \mathbb{R}^{n_2}$ and $\sigma_2 : \mathbb{R}^{n_2} \to \mathbb{R}^{n_2}$.

After finishing the construction of all connections, we can put our data x into this neural network, x will go through all these connections and output a vector $\hat{y} = a^{[M]}$. This process is called **feed forward**, **predict** or **inference**. Though this structure is quite complex, we can just interpret a neural network as a vector function f that

$$f : \mathbb{R}^{n_1} \to \mathbb{R}^{n_M}$$

The goal of a neural network is predicting a vector $\hat{y}$ that almost the same as a predefined ground truth y. When $\hat{y}$ is way more different from y, we need a mechanism to let this network correct all the connections to let $\hat{y}$ look more alike to y. So here comes the loss function, the double arrow in the picture. It's a function that compare how similar $\hat{y}$ and y are. In general, we'll choose a function that take two vectors as input and a non negative number as output, and then **define these two vectors are similar when the output number closes to** 0.(For example, the well known L2 norm $\|\hat{y} - y\|_2$.) That is to say, the mechanism need to let the network try its best to lower down the value of the loss function(by adjusting the connection), then its output $\hat{y}$ will look like the ground truth y.

In reality, there won't be only one (x, y) pair. When we have lots of training data $\{(x_i, y_i)\}_1^n$, we'll give a neural network lots of example to follow up:

Input x_1, output $\hat{y}_1$, adjust the connection to fit $\hat{y}_1$ to y_1; Input x_2, output $\hat{y}_2$, adjust the connection to fit $\hat{y}_2$ to y_2; Input x_3, output $\hat{y}_3$, adjust the connection to fit $\hat{y}_3$ to y_3, ...

This mechanism is so called **training a neural network** and we use a method called **back propagation** to achieve training.

We need to set up some notations to explain the details of how back propagation works.

1.2 Notations

Assume there are M layers. The k^{th} layer will has n_k neurons and is denoted by a vector $a^{[k]} \in \mathbb{R}^{n_k}$. The matrix, bias vector and the activation function between the $k-1^{\text{th}}$ layer and the k^{th} layer are denoted by $W^{[k]} \in \mathbb{R}^{n_k \times n_{k-1}}$, $b^{[k]} \in \mathbb{R}^{n_k}$ and $\sigma_k : \mathbb{R}^{n_k} \to \mathbb{R}^{n_k}$, respectively, where $k = 2, \ldots, M$. To simplify the computation, we let the activation functions are all the same as the function $\sigma(x) = \frac{1}{1+e^{-x}}$ and note that

$$\sigma(x) = \begin{pmatrix} \frac{1}{1+e^{-x_1}} \\ \frac{1}{1+e^{-x_2}} \\ \vdots \\ \frac{1}{1+e^{-x_n}} \end{pmatrix}, \quad \text{for } x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R}^n$$

Let $\{(x_i, y_i)\}_1^n$ be our training data, then the feed forward process can be formalized as

$$\begin{aligned}
a_i^{[1]} &= x_i \in \mathbb{R}^{n_1}, \\
a_i^{[2]} &= \sigma(W^{[2]} a_i^{[1]} + b^{[2]}) \in \mathbb{R}^{n_2}, \\
a_i^{[3]} &= \sigma(W^{[3]} a_i^{[2]} + b^{[3]}) \in \mathbb{R}^{n_3}, \\
&\vdots \\
a_i^{[k]} &= \sigma(W^{[k]} a_i^{[k-1]} + b^{[k]}) \in \mathbb{R}^{n_k}, \\
&\vdots \\
a_i^{[M]} &= \sigma(W^{[M]} a_i^{[M-1]} + b^{[M]}) = \hat{y}_i \in \mathbb{R}^{n_M}
\end{aligned}$$

We need to let neural network know what "closeness of two vectors" actually mean. Let's defined a loss function that take $\hat{y}_i$ and y_i as input and a non negative number as output:

$$L(\hat{y}_i, y_i) = \frac{1}{2} \|\hat{y}_i - y_i\|_2^2$$

When $L(\hat{y}_i, y_i) \to 0$, we consider $\hat{y}_i$ and y_i are similar at neural network's aspect.

You may wonder why we choose L2 norm rather than other function. The answer is:

There's no best solution of how to choose a loss function, you need lots of experiment and paper reading to hone your intuition.

1.3 *Training

$$\sum_{i=1}^{n} L(a_i^{[M]}, y) = \sum_{i=1}^{n} \frac{1}{2} \left\| a_i^{[M]} - y_i \right\|_2^2$$

The recursive relations of each $a^{[i]}$ shows that the loss function actually depends on all the matrices $W^{[i]}$, all the bias vectors $b^{[i]}$ and the ground truth y

$$L(W^{[2]}, W^{[3]}, \ldots, W^{[M]}, b^{[2]}, b^{[3]}, \ldots, b^{[M]}, y) = \frac{1}{2} \left\| y - a^{[M]} \right\|_2^2$$

We can ignore y here since it's a constant in the computation, so

$$L(W^{[2]}, W^{[3]}, \ldots, W^{[M]}, b^{[2]}, b^{[3]}, \ldots, b^{[M]}) = \frac{1}{2} \left\| y - a^{[M]} \right\|_2^2$$

Now our goal is finding each $W^{[k]}$ and $b^{[k]}$ to let $a^{[M]}$ be closer to y. This is what training means in mathematics.

Let $\theta = (W^{[2]}, \cdots, W^{[M]}, b^{[2]}, \cdots, b^{[M]})$, then $\theta \in \mathbb{R}^D$, where

$$D = \sum_{k=2}^{M} n_k \times n_{k-1} + \sum_{k=2}^{M} n_k$$

Then the training problem becomes like this

$$\arg\min_{\theta} \left\| y - a^{[M]} \right\|_2^2 \equiv \arg\min_{\theta} L(\theta)$$

where $L : \mathbb{R}^D \to \mathbb{R}$.

1.4 Main Computations

To compute $\nabla L(\theta)$. Let $z^{[k]} = W^{[k]} a^{[k-1]} + b^{[k]}$ then $a^{[k]} = \sigma(z^{[k]})$. Let j be the index represents neurons of each layer. The notation becomes

$$\begin{aligned}
&z_j^{[k]} = (W^{[k]} a^{[k-1]} + b^{[k]})_j = (W^{[k]} a^{[k-1]})_j + b_j^{[k]}, \\
&a_j^{[k]} = \sigma(z^{[k]})_j, \\
&k = 2, \ldots, M, \quad j = 1, \ldots, n_k
\end{aligned}$$

The core concept of back propagation is using deep level derivatives to compute shallow level derivative. This can be achieved with the help of chain rule. First we derive the derivative of $z_j^{[k]}$(the j^{th} neuron of the k^{th} layer).

$$\begin{aligned}
\frac{\partial L(\theta)}{\partial z_j^{[M]}} &= \frac{\partial L(\theta)}{\partial a_j^{[M]}} \frac{\partial a_j^{[M]}}{\partial z_j^{[M]}} = \frac{\partial L(\theta)}{\partial a_j^{[M]}} \sigma'(z_j^{[M]}) = (a_j^{[M]} - y_j)\sigma'(z_j^{[M]}) = (a_j^{[M]} - y_j) a_j^{[M]} (1 - a_j^{[M]}), \\
\frac{\partial L(\theta)}{\partial z_j^{[M-1]}} &= \frac{\partial}{\partial z_j^{[M-1]}} \frac{1}{2} \sum_{k=1}^{n_M} \left(y_k - \sigma\big(W^{[M]} \sigma(z^{[M-1]}) + b^{[M]}\big)_k \right)^2 \\
&= \nabla_{z^{[M]}} L(\theta) \cdot \frac{\partial z^{[M]}}{\partial z_j^{[M-1]}} \\
&= \sum_{m=1}^{n_M} \frac{\partial L(\theta)}{\partial z_m^{[M]}} \frac{\partial z_m^{[M]}}{\partial z_j^{[M-1]}} \\
&= \sum_{m=1}^{n_M} \frac{\partial L(\theta)}{\partial z_m^{[M]}} \left(\frac{\partial}{\partial z_j^{[M-1]}} \sum_{s=1}^{n_{M-1}} w_{ms}^{[M]} \sigma(z_s^{[M-1]}) + b_m^{[M]} \right)
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{m=1}^{n_M} \frac{\partial L(\theta)}{\partial z_m^{[M]}} w_{mj}^{[M]} \sigma'(z_j^{[M-1]}) \\
&= \sigma'(z_j^{[M-1]}) \left((W^{[M]})^T \frac{\partial L(\theta)}{\partial z^{[M]}} \right)_j \\
&= \left((W^{[M]})^T \frac{\partial L(\theta)}{\partial z^{[M]}} \right)_j a_j^{[M-1]} (1 - a_j^{[M-1]}), \\
&\vdots \\
\frac{\partial L(\theta)}{\partial z_j^{[1]}} &= \left((W^{[2]})^T \frac{\partial L(\theta)}{\partial z^{[2]}} \right)_j a_j^{[1]} (1 - a_j^{[1]})
\end{aligned}
$$

From above we can see that we *back propagate* the derivative since we need $\partial z^{[M]}$ to yield $\partial z_j^{[M-1]}$. The partial derivative of $z_j^{[k]}$ can help us get the gradient w.r.t weight and bias.

$$
\begin{aligned}
\frac{\partial L(\theta)}{\partial b_j^{[k]}} &= \frac{\partial L(\theta)}{\partial z_j^{[k]}} \frac{\partial z_j^{[k]}}{\partial b_j^{[k]}} = \frac{\partial L(\theta)}{\partial z_j^{[k]}}, \\
\frac{\partial L(\theta)}{\partial w_{js}^{[k]}} &= \frac{\partial L(\theta)}{\partial z_j^{[k]}} \frac{\partial z_j^{[k]}}{\partial w_{js}^{[k]}} = \frac{\partial L(\theta)}{\partial z_j^{[k]}} a_s^{[k]}
\end{aligned}
$$

And hence we can back propagate each $\partial b_j^{[k]}$ and $\partial w_{js}^{[k]}$.

For the case with multiple training instances (x_i, y_i), $i = 1, \ldots, N$. The loss function becomes

$$
L(\theta) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} \| y_i - a_i^{[[M]} \|_2^2
$$

The computation are all the same but the notation will become a little bit complicated.

2 Simple Regression Model

2.1 Linear

2.1.1 One Dimension

Each instance has one attribute x corresponds to one label y.

$$
y = wx + b, \quad y,\ x,\ w,\ b \in \mathbb{R}
$$

2.1.2 Multiple Dimensions

Each instance has multiple attributes $x_1, \ldots, x_n$ and multiple labels $y_1, \ldots, y_m$.

$$
\begin{aligned}
&\boldsymbol{y} = \boldsymbol{w}\boldsymbol{x} + \boldsymbol{b}, \\
&\boldsymbol{x} \in \mathbb{R}^n, \\
&\boldsymbol{w} \in \mathbb{R}^{m \times n}, \\
&\boldsymbol{y},\ \boldsymbol{b} \in \mathbb{R}^m,
\end{aligned}
$$

$$\begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix} = \begin{pmatrix} w_{11} & \dots & w_{1_n} \\ \vdots & \ddots & \vdots \\ w_{m1} & \dots & w_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$$

2.2 Nonlinear(Polynomial Model)

2.2.1 One Dimension

Each instance has one attribute x correponds to one label y.

$$\begin{aligned} y &= w_2 x^2 + w_1 x + b, \\ &\vdots \\ y &= w_k x^k + w_{k-1} x^{k-1} + \cdots + w_1 x + b \end{aligned}$$

2.2.2 Multiple Dimensions

Each instance has multiple attributes $x_1, \dots, x_n$ and multiple labels $y_1, \dots, y_m$.

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{w}_2 \boldsymbol{x}^2 + \boldsymbol{w}_1 \boldsymbol{x} + \boldsymbol{b}, \\ &\vdots \\ \boldsymbol{y} &= \boldsymbol{w}_k \boldsymbol{x}^k + \boldsymbol{w}_{k-1} \boldsymbol{x}^{k-1} + \cdots + \boldsymbol{w}_1 \boldsymbol{x} + \boldsymbol{b} \end{aligned}$$

where

$$\boldsymbol{x}^p = \begin{pmatrix} x_1^p \\ \vdots \\ x_n^p \end{pmatrix}, \ \boldsymbol{w}_p \in \mathbb{R}^{m \times n}, \ \boldsymbol{y}, \boldsymbol{b} \in \mathbb{R}^m$$

3 Regression Loss Functions

3.1 L^p Norm

Let $\boldsymbol{y}(\boldsymbol{w}, \boldsymbol{b}) : \mathbb{R}^n \to \mathbb{R}^m$ be a model defined by $\boldsymbol{w}$ and $\boldsymbol{b}$ that map $\boldsymbol{x} \in \mathbb{R}^n$ into $\boldsymbol{y} \in \mathbb{R}^m$. Mathematically the L^1 norm is

$$\|\boldsymbol{y}(\boldsymbol{w}, \boldsymbol{b}) - \hat{\boldsymbol{y}}\|_1 = \sum_{k=1}^{m} \left| y_k(\boldsymbol{w}, \boldsymbol{b}) - \hat{y}_k \right|$$

The L^p norm is

$$\|\boldsymbol{y}(\boldsymbol{w}, \boldsymbol{d}) - \hat{\boldsymbol{y}}\|_p = \left(\sum_{k=1}^{m} \left(y_k(\boldsymbol{w}, \boldsymbol{b}) - \hat{y}_k \right)^p \right)^{1/p}$$

The L^∞ norm is

$$\|\boldsymbol{y}(\boldsymbol{w}, \boldsymbol{b}) - \hat{\boldsymbol{y}}\|_\infty = \max_k |y_k(\boldsymbol{w}, \boldsymbol{b}) - \hat{y}_k|$$

3.2 Mean Absolute Error(L^1 Loss)

Suppose there are N instances $\{\boldsymbol{x}_i\}_{i=1}^N$. The MAE is defined by

$$L(\boldsymbol{w}, \boldsymbol{b}) = \frac{\sum_{n=1}^{N} \|\boldsymbol{y}^n(\boldsymbol{w}, \boldsymbol{b}) - \hat{\boldsymbol{y}^n}\|_1}{N}$$

3.3 Mean Square Error(L^2 Loss)

Suppose there are N instances $\{\boldsymbol{x}_i\}_{i=1}^N$. The MSE is defined by

$$L(\boldsymbol{w}, \boldsymbol{b}) = \frac{\sum_{n=1}^N \|\boldsymbol{y}^n(\boldsymbol{w}, \boldsymbol{b}) - \hat{\boldsymbol{y}^n}\|_2^2}{N}$$

3.4 Regularization

Add new term

$$\tilde{L}(\boldsymbol{w}, \boldsymbol{b}) = L(\boldsymbol{w}, \boldsymbol{b}) + \lambda \|\boldsymbol{w}\|_2^2$$

The last term will make loss function smoother since we also minimize weights.

4 Calssification Loss Functions

4.1 Hinge Loss

Let $g(x)$ be a classifier that defined by a score function $f(x)$

$$g(x) = \begin{cases} 1 & \text{if } f(x) > 0 \\ -1 & \text{if } f(x) < 0 \end{cases}$$

Suppose there are N data points $x_1, \ldots, x_N$ with labels $\hat{y}_1, \ldots, \hat{y}_N \in \{-1, 1\}$. The hinge loss of g is defined to be

$$l\big(g(x_n), \hat{y}_n\big) = \max\{0, 1 - \hat{y}_n f(x_n)\}$$

When $f(x_n) \geq 1$ or $f(x_n) \leq -1$, both implies $\hat{y}_n f(x_n) \geq 1$. This means $l\big(g(x_n), \hat{y}_n\big) = 0$. When $f(x_n) \in (-1, 1)$, we have $\hat{y}_n f(x_n) \in [0, 1)$. Hence $l\big(g(x_n), \hat{y}_n\big) = 1 - \hat{y}_n f(x_n)$.

4.2 Cross Entropy

Let $p(x)$ be an unknown distribution and we use $q(x)$ to esitmate it. The cross entropy of $q(x)$ relative to $p(x)$ is

$$H(p, q) = -\mathrm{E}_p[\ln q] = -\sum_x p(x) \ln q(x)$$

Roughly speaking, this stands for the average amount of information to transimit when we use $q(x)$ as $p(x)$. Compare this with only use $p(x)$, the additional information will be transimitted is called the KL divergenc between $p(x)$ and $q(x)$

$$\mathrm{KL}(p\|q) = H(p, q) - H(p) = \left(-\sum_x p(x) \ln q(x)\right) - \left(-\sum_x p(x) \ln p(x)\right)$$

where $H(p) = -\sum_x p(x) \ln p(x)$ is the entropy of $p(x)$ alone.

5 Gradient Descent

5.1 Vanilla Gradient Descent

Compute gradient for the entire training set and then update parameters

$$\theta^t = \theta^{t-1} - \alpha \nabla L(\theta^{t-1})$$

5.2 Stochastic Gradient Descent

Compute gradient for one training data and then update parameters subsequently.

$$\theta^t = \theta^{t-1} - \alpha \nabla L_i(\theta^{t-1})$$

where L_i means the loss function applies on only one instance.

5.2.1 Randomly Choose and Return

$\{x_1, \ldots, x_N\}$, $L(\theta) = \frac{1}{N}\sum_{i=1}^{N} L_i(\theta)$. Choose x_i uniformly randomly from $\{x_1, \ldots, x_N\}$.

$$\theta^1 = \theta^0 - \alpha \nabla L_i(\theta^0)$$

x_i will not be pulled out from instances. Choose new x_j uniformly randomly again.

$$\theta^2 = \theta^1 - \alpha \nabla L_j(\theta^1)$$

5.2.2 Shuffle and Not Return

Randomly shuffle $\{1, \ldots, N\}$ into $\{k_1, \ldots, k_N\}$. For i for 1 to N,

$$\theta^t = \theta^{t-1} - \alpha \nabla L_{k_i}(\theta^{i-1})$$

One process go trough all instances is an "epoch".

5.3 Mini-Batch Gradient Descent

Split instances into m subsets, each subset (or mini-batch) has n instances. Update θ per mini-batch.

$$\theta^t = \theta^{t-1} - \alpha \frac{1}{m} \sum_{i=1}^{m} \nabla L_i(\theta)$$

After going through all batches, we complete one "epoch".

6 Tuning Learning Rate(Optimizer)

6.1 Momentum

Let v_k be the variable that stores previous move, i.e. the momentum. In the beginning, $v_0 = 0$.

Initialize θ_0 and let $v_0 = 0$.

$$\begin{aligned}
&\theta_1 = \theta_0 + v_1, \quad v_1 = \lambda v_0 - \alpha \nabla L(\theta_0) = -\alpha \nabla L(\theta_0), \\
&\theta_2 = \theta_1 + v_2, \quad v_2 = \lambda v_1 - \alpha \nabla L(\theta_1) = -\lambda\alpha \nabla L(\theta_0) - \alpha \nabla L(\theta_1), \\
&\quad \vdots \\
&\theta_{t+1} = \theta_t + v_{t+1}, \quad v_{t+1} = \lambda v_t - \alpha \nabla L(\theta_t)
\end{aligned}$$

Briefly, momentum method perturb current gradient by previous gradient(momentum).

6.2 Nesterov Accelerated Gradient(NAG)

Similar to momentum

Initialize θ_0 and let $v_0 = 0$.

$$\begin{aligned}
&\theta_1 = \theta_0 + v_1, \quad v_1 = \lambda v_0 - \alpha \nabla L(\theta_0 + \lambda v_0) = -\alpha \nabla L(\theta_0), \\
&\theta_2 = \theta_1 + v_2, \quad v_2 = \lambda v_1 - \alpha \nabla L(\theta_1 + \lambda v_1) = -\lambda\alpha \nabla L(\theta_0) - \alpha \nabla L(\theta_1 + \lambda v_1), \\
&\quad \vdots \\
&\theta_{t+1} = \theta_t + v_{t+1}, \quad v_{t+1} = \lambda v_t - \alpha \nabla L(\theta_t + \lambda v_t)
\end{aligned}$$

Here, instead of perturbing current gradient, we perturb current parameter by previous gradient.

6.3 Adagrad(Adaptive Gradient)

Use first derivative to estimate second derivative.

$$\begin{aligned}
\alpha &\leftarrow \frac{\alpha}{\sqrt{\sum_{i=0}^{t} \left(\nabla L(\theta_i)\right)^2}}, \\
\theta_t &\leftarrow \theta_{t-1} - \frac{\alpha}{\sqrt{\sum_{i=0}^{t} \left(\nabla L(\theta_i)\right)^2}} \nabla L(\theta^{t-1})
\end{aligned}$$

In practice we will add ϵ in the denominator to avoid dividing by zero

$$\theta_t \leftarrow \theta_{t-1} - \frac{\alpha}{\sqrt{\sum_{i=0}^{t} \left(\nabla L(\theta_i)\right)^2 + \epsilon}} \nabla L(\theta^{t-1})$$

6.4 Adadelta

This method needs the average of gradients $\mathrm{E}[\nabla L(\theta)^2]_t$ at step t and the average of parameter update $\mathrm{E}[\Delta\theta^2]_t$. Initialize $\mathrm{E}[\nabla L(\theta)^2]_0 = 0,\ \mathrm{E}[\Delta\theta^2]_0 = 0$ and choose a decay rate ρ, learning rate α and small ϵ.

I. $\theta_1 = \theta_0 + \Delta\theta_0,\ \Delta\theta_0 = -\dfrac{\alpha}{\sqrt{\mathrm{E}[\nabla L(\theta)^2]_0 + \epsilon}} \nabla L(\theta_0)$

II. $\mathrm{E}[\Delta\theta^2]_0 = 0$

II. $\mathrm{E}[\nabla L(\theta)^2]_1 = \rho \mathrm{E}[\nabla L(\theta)^2]_0 + (1-\rho) \nabla L(\theta_1)^2$

II. $\theta_2 = \theta_1 + \Delta\theta_1,\ \Delta\theta_1 = -\dfrac{\sqrt{\mathrm{E}[\Delta\theta^2]_0 + \epsilon}}{\sqrt{\mathrm{E}[\nabla L(\theta)^2]_1 + \epsilon}} \nabla L(\theta_1) \equiv -\dfrac{RMS[\Delta\theta]_0}{RMS[\nabla L(\theta)]_1} \nabla L(\theta_1)$

III. $\mathrm{E}[\Delta\theta^2]_1 = \rho \mathrm{E}[\Delta\theta^2]_0 + (1-\rho) \Delta\theta_1^2$

III. $\mathrm{E}[\nabla L(\theta)^2]_2 = \rho \mathrm{E}[\nabla L(\theta)^2]_1 + (1-\rho) \nabla L(\theta_2)^2$

III. $\theta_3 = \theta_2 + \Delta\theta_2,\ \Delta\theta_2 = -\dfrac{RMS[\Delta\theta]_1}{RMS[\nabla L(\theta)]_2} \nabla L(\theta_2)$

#. $\mathrm{E}[\Delta\theta^2]_{t-2} = \rho \mathrm{E}[\Delta\theta^2]_{t-3} + (1-\rho) \Delta\theta_{t-2}^2$

#. $\mathrm{E}[\nabla L(\theta)^2]_t = \rho \mathrm{E}[\nabla L(\theta)^2]_{t-1} + (1-\rho) \nabla L(\theta_t)^2$

$$\#.\ \theta_t = \theta_{t-1} + \Delta\theta_{t-1},\ \Delta\theta_{t-1} = -\frac{RMS[\Delta\theta]_{t-2}}{RMS[\nabla L(\theta)]_{t-1}}\nabla L(\theta_{t-1})$$

6.5 RMSprop(Root Mean Square Propagation)

Manually determine a weight β.

$$\begin{aligned}
\theta_1 &\leftarrow \theta_0 - \frac{\alpha}{\sigma_0}\nabla L(\theta_0), \quad \sigma_0 = \nabla L(\theta_0),\\
\theta_2 &\leftarrow \theta_1 - \frac{\alpha}{\sigma_2}\nabla L(\theta_1), \quad \sigma_1 = \sqrt{\beta(\sigma_0)^2 + (1-\beta)\big(\nabla L(\theta_1)\big)^2 + \epsilon},\\
&\vdots\\
\theta_{t+1} &\leftarrow \theta_t - \frac{\alpha}{\sigma_t}\nabla L(\theta_t), \quad \sigma_t = \sqrt{\beta(\sigma_{t-1})^2 + (1-\beta)\big(\nabla L(\theta_t)\big)^2 + \epsilon}
\end{aligned}$$

6.6 Adam(RMSprop+Momentum)

Two weight numbers β_1 and β_2. Two moment vectors v_k and σ_k. In the beginning $v_0 = 0$ and $\sigma_0 = 0$.

$$\begin{aligned}
\theta_1 &= \theta_0 - \alpha\frac{\sigma_1}{\sqrt{v_1}+\epsilon}, \quad \sigma_1 = \frac{\beta_1\sigma_0 + (1-\beta_1)\nabla L(\theta_0)}{1-\beta_1}, \quad v_1 = \frac{\beta_2 v_0 + (1-\beta_2)\big(\nabla L(\theta_0)\big)^2}{1-\beta_2},\\
\theta_2 &= \theta_1 - \alpha\frac{\sigma_2}{\sqrt{v_2}+\epsilon}, \quad \sigma_2 = \frac{\beta_1\sigma_1 + (1-\beta_1)\nabla L(\theta_1)}{1-\beta_1^2}, \quad v_2 = \frac{\beta_2 v_1 + (1-\beta_2)\big(\nabla L(\theta_1)\big)^2}{1-\beta_2^2},\\
&\vdots\\
\theta_{t+1} &= \theta_t - \alpha\frac{\sigma_{t+1}}{\sqrt{v_{t+1}}+\epsilon}, \quad \sigma_{t+1} = \frac{\beta_1\sigma_t + (1-\beta_1)\nabla L(\theta_t)}{1-\beta_1^t}, \quad v_{t+1} = \frac{\beta_2 v_t + (1-\beta_2)\big(\nabla L(\theta_t)\big)^2}{1-\beta_2^t}
\end{aligned}$$

7 Probabilistic Generative Model

$P(A|B) = \frac{P(A\cap B)}{P(B)} = \frac{P(A,B)}{P(B)}$: the probability of A given B. By definition,

$$P(A|B)\frac{P(B)}{P(A)} = \frac{P(A,B)}{P(A)} = P(B|A)$$

There are two classes C_1, C_2. The probability of x is

$$P(x) = P(x|C_1)P(C_1) + P(x|C_2)P(C_2)$$

The posterior probability

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

8 Basic Statistic Notions

8.1 Expected Value $E[X] = \mu$

$$E[X] = \begin{cases} \sum x_i p_i & \text{if discrete} \\ \int_{\mathbb{R}} x f(x) dx & \text{if continuous} \end{cases}$$

8.2 Variance $var[X] = \sigma_X^2$

$$var[X] = \begin{cases} \sum (x_i - \mu)^2 p_i & \text{if discrete} \\ \int_{\mathbb{R}} (x-\mu)^2 f(x) dx & \text{if continuous} \end{cases}$$

If X is continuous, $var[X] = E[(X - E[X])^2] = E[X^2] - E[X]^2$.

8.3 Standard Deviation σ_X

$$\sigma_X = \begin{cases} \sqrt{\sum (x_i - \mu)^2 p_i} & \text{if discrete} \\ \sqrt{\int_{\mathbb{R}} (x-\mu)^2 f(x) dx} & \text{if continuous} \end{cases}$$

Note that $\sigma_X = \sqrt{var[X]}$. If X is discrete and each $p_i = \frac{1}{N}$, $\sigma_X = \sqrt{\frac{\sum (x_i - \mu)^2}{N}}$.

8.4 Covariance $cov[X, Y]$

8.4.1 Two Variables

X, Y are random variables with space $\mathbb{R}$.

$$cov[X,Y] = \begin{cases} \sum p_i (x_i - \mu_X)(y_i - \mu_Y) & \text{if discrete} \\ \int_{\mathbb{R}\times\mathbb{R}} (x-\mu_X)(y-\mu_Y) f(x,y) & \text{if continuous} \end{cases}$$

Note that $cov[X, X] = var[X]$ and covariance matrix $\Sigma = \begin{pmatrix} cov[X,X] & cov[X,Y] \\ cov[Y,X] & cov[Y,Y] \end{pmatrix}$

8.4.2 More Variables

$X_1, X_2, \ldots, X_n$ are random variables with space $\mathbb{R}$.
The covariance matrix Σ is

$$\Sigma = \begin{pmatrix} cov[X_1,X_1] & cov[X_1,X_2] & \cdots & cov[X_1,X_n] \\ cov[X_2,X_1] & cov[X_2,X_2] & \cdots & cov[X_2,X_n] \\ \vdots & \vdots & \ddots & \vdots \\ cov[X_n,X_1] & cov[X_n,X_2] & \cdots & cov[X_n,X_n] \end{pmatrix}$$

Or let $X = (X_1, X_2, \ldots, X_n)$, $\Sigma = cov[X, X] = E[(X - E[X])(X - E[X])^T]$.

8.5 Correlation $corr[X, Y] = \rho_{X,Y}$

$corr[X, Y] = \frac{cov[X,Y]}{\sigma_X \sigma_Y} \in [-1, 1]$.

8.6 Assume sample come from one distribution

8.6.1 Gaussian Distribution

$$f_{\mu,\Sigma}(x) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

where x vector, D for dimension, μ mean vector, Σ covariance matrix.
An unknown $f_{\mu,\Sigma}(x)$ samples n points $x_1, \ldots, x_n$. The likelihood function

$$L(\mu, \Sigma) = f_{\mu,\Sigma}(x_1) f_{\mu,\Sigma}(x_2) \cdots f_{\mu,\Sigma}(x_n)$$

Higher value of the likelihood function means higher probability to sample $x_1, \ldots, x_n$. The maximum likelihood estimator (μ^*, Σ^*) is

$$\mu^*, \Sigma^* = \arg\max_{\mu,\Sigma} L(\mu, \Sigma)$$

Also, for $l(\mu, \Sigma) = \ln L(\mu, \Sigma)$,

$$\mu^*, \Sigma^* = \arg\max_{\mu,\Sigma} l(\mu, \Sigma)$$

Then Gaussian distribution

$$\mu^* = \frac{1}{n} \sum_1^n x_i, \quad \Sigma^* = \frac{1}{n} \sum_1^n (x_i - \mu^*)(x_i - \mu^*)^T$$

8.7 Two Classes C_1, C_2

C_1 has $\{x_i\}_1^{70}$, C_2 has $\{x_i\}_{71}^{180}$. The posterior probability of an unknown object x is

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

The terms $P(C_k)$ are easy to get. The terms $P(x|C_k)$ is based on the distribution we assumed. If it's Gaussian,

$$P(x|C_1) = f_{\mu_1^*,\Sigma_1^*}(x), \quad P(x|C_2) = f_{\mu_2^*,\Sigma_2^*}(x)$$

We can modify Σ^* by

$$\Sigma^* = \frac{70}{70+110}\Sigma_1^* + \frac{110}{70+110}\Sigma_2^*$$

And let C_1, C_2 share this Σ^*, the likelihood function becomes

$$L(\mu_1, \mu_2, \Sigma) = f_{\mu_1,\Sigma}(x_1) \cdots f_{\mu_1,\Sigma}(x_{70}) f_{\mu_2,\Sigma}(x_{71}) \cdots f_{\mu_2,\Sigma}(x_{180})$$

$\mu_1^*, \mu_2^*, \Sigma^*$ are maximum likelihood estimators. The new model will be a linear classifier. Explain in the following.

Let $z = \ln \frac{P(x|C_1)P(C_1)}{P(x|C_2)P(C_2)}$. Then

$$P(C_1|x) = \frac{1}{1+e^{-z}} = \sigma(z), \quad \text{the sigmoid function.}$$

$P(x|C_k) = f_{\mu_k^*,\Sigma^*}(x)$, $k = 1, 2$. So

$$z = \ln \frac{f_{\mu_1^*,\Sigma^*}(x)}{f_{\mu_2^*,\Sigma^*}(x)} + \ln \frac{P(C_1)}{P(C_2)}$$

Put the details in and compute, we get

$$z = (\mu_1^* - \mu_2^*)^T \Sigma^{*-1} x - \frac{1}{2}(\mu_1^*)^T \Sigma^{*-1} \mu_1^* + \frac{1}{2}(\mu_2^*)^T \Sigma^{*-1} \mu_2^* + \ln \frac{P(C_1)}{P(C_2)}$$

Let $w^T = (\mu_1^* - \mu_2^*)^T \Sigma^{*-1}$, $b = -\frac{1}{2}(\mu_1^*)^T \Sigma^{*-1} \mu_1^* + \frac{1}{2}(\mu_2^*)^T \Sigma^{*-1} \mu_2^* + \ln \frac{P(C_1)}{P(C_2)}$. Then

$$P(C_k|x) = \sigma(w \cdot x + b)$$

9 Logistic Regression

Put the function of linear regression into sigmoid function, the output value will lie in $(0, 1)$.

$$f_{w,b}(x) = \sigma(w \cdot x + b) = \sigma\left(\sum_i w_i x_i + b\right)$$

In the training set $\{(x_k, \hat{y}_k)\}_k$, $\hat{y}_k \in \{0, 1\}$. 1 for class C_1, 0 for class C_2. If $(x_1, x_2, x_3, \dots)$ corresponds to $(1, 1, 0, \dots)$. The loss function

$$L(w, b) = f_{w,b}(x_1) f_{w,b}(x_2) \big(1 - f_{w,b}(x_3)\big) \dots, \quad w^*, b^* = \arg\max_{w,b} L(w, b)$$

Note that

$$w^*, b^* = \arg\min - \ln L(w, b)$$

And

$$\begin{aligned} - \ln L(w, b) &= - \ln f_{w,b}(x_1) - \ln f_{w,b}(x_2) - \ln(1 - f_{w,b}(x_3)) \dots \\ &= \sum_k - \left[\hat{y}_k \ln f_{w,b}(x_k) + (1 - \hat{y}_k) \ln \big(1 - f_{w,b}(x_k)\big)\right] \end{aligned}$$

The relation in the brackets [ ] is called the cross entropy between two Bernoulli distribution.

9.1 Comparison with Linear Regression

Simple computation shows that

$$\frac{\partial - \ln L(w, b)}{\partial w_i} = \sum_k - \big(\hat{y}_k - f_{w,b}(x_k)\big) x_{k,i}$$

So w_i will update in the same way with linear regression. The only difference between them is the range of output. Logistic regression lies in $(0, 1)$ while linear regression can be any real number.

10 Principal Component Analysis(PCA)

10.1 Settings

Let $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^p$ be a sample of n observations of p dimensional vectors. The first principal component of this sample is a real variable transformed from X

$$\mathbf{z}_1 = \mathbf{X}\mathbf{a}_1^T$$

where the vector $\mathbf{a}_1 = (a_{11}, a_{21}, \ldots, a_{p1}) \in \mathbb{R}^p$ is chosen so that $\|\mathbf{a}_1\|_2 = 1$ and $var[\mathbf{z}_1]$ is maximized.
The k^{th} principal component of the sample is

$$\mathbf{z}_k = \mathbf{X}\mathbf{a}_k^T, \quad k = 2, \ldots, p$$

where the vector $\mathbf{a}_k = (a_{1k}, a_{2k}, \ldots, a_{pk}) \in \mathbb{R}^p$ is chosen so that $\|\mathbf{a}_k\|_2 = 1$, $var[\mathbf{z}_k]$ is maximized and $cov[\mathbf{z}_k, \mathbf{z}_l] = 0$ for $k > l \geq 1$.

Geometrically, at first step, PCA finds a unit vector $\mathbf{a}_1 \in \mathbb{R}^p$ that the projections of sample points on it has maximum variance. $\mathbf{z}_1$ is actually a variable of projection length of sample points on $\mathbf{a}_1$. Secondly, PCA finds the second component $\mathbf{z}_2$ in the same logic but with further restriction that $\mathbf{z}_2$ has no relation with $\mathbf{z}_1$, i.e. $cov[\mathbf{z}_2, \mathbf{z}_1] = 0$.

10.2 Computations

To find $\mathbf{a}_1$, first note that

$$\begin{aligned}
var[\mathbf{z}_1] =& \mathrm{E}[\mathbf{z}_1^2] - \mathrm{E}[\mathbf{z}_1]^2 \\
=& \sum_{i,j=1}^{p} a_{i1}a_{j_1}\mathrm{E}[X_iX_j] - \sum_{i,j=1}^{p} a_{i1}a_{j1}\mathrm{E}[X_i]\mathrm{E}[X_j] \\
=& \sum_{i,j=1}^{p} a_{i1}a_{j1}cov[X_i, X_j] \\
=& (a_{11}, a_{21}, \ldots, a_{p1}) \begin{pmatrix} cov[X_1, X_1] & cov[X_1, X_2] & \ldots & cov[X_1, X_p] \\ cov[X_2, X_1] & cov[X_2, X_2] & \ldots & cov[X_2, X_p] \\ \vdots & \vdots & \ddots & \vdots \\ cov[X_p, X_1] & cov[X_p, X_2] & \ldots & cov[X_p, X_p] \end{pmatrix} \begin{pmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{p1} \end{pmatrix} \\
=& \mathbf{a}_1\Sigma\mathbf{a}_1^T
\end{aligned}$$

where Σ is the covariance matrix of X. The problem becomes

$$\max(\mathbf{a}_1\Sigma\mathbf{a}_1^T) \quad s.t. \quad \|\mathbf{a}_1\|_2 = 1$$

Let λ be Lagrange multiplier and let $g(\mathbf{a}_1) = \mathbf{a}_1\Sigma\mathbf{a}_1^T - \lambda(\mathbf{a}_1\mathbf{a}_1^T - 1)$. Then we need

$$\nabla g(\mathbf{a}_1) = \mathbf{a}_1\Sigma^T - \lambda\mathbf{a}_1 = (\Sigma\mathbf{a}_1^T - \lambda\mathbf{a}_1^T)^T = 0$$

Therefore, $\mathbf{a}_1^T$ is the eigenvector of Σ corresponding to eigenvalue $\lambda \equiv \lambda_1$. Furthermore, we have maximized $var[\mathbf{z}_1] = \mathbf{a}_1\Sigma\mathbf{a}_1^T = \lambda_1$, this means λ_1 is the largest eigenvalue of Σ.

For the second component, note that $cov[\mathbf{z}_2, \mathbf{z}_1] = \mathbf{a}_2\Sigma\mathbf{a}_1^T = \mathbf{a}_2\lambda_1\mathbf{a}_1^T = 0$. For another multipliers ϕ and λ and another $g(\mathbf{a}_2) = \mathbf{a}_2\Sigma\mathbf{a}_2^T - \lambda(\mathbf{a}_2\mathbf{a}_2^T - 1) - \phi\lambda_1\mathbf{a}_2\mathbf{a}_1^T$, we need

$$\nabla g(\mathbf{a}_2) = (\Sigma\mathbf{a}_2^T - \lambda\mathbf{a}_2^T - \phi\lambda_1\mathbf{a}_1^T)^T = 0$$

Multiply $\mathbf{a}_1$ both side, we get

$$\begin{aligned}
&\mathbf{a}_1\Sigma\mathbf{a}_2^T - \mathbf{a}_1\lambda\mathbf{a}_2^T - \phi\lambda_1\mathbf{a}_1\mathbf{a}_1^T \\
=&(\mathbf{a}_2\Sigma\mathbf{a}_1)^T - \lambda(\mathbf{a_2}\mathbf{a}_1)^T - \phi\lambda_1 = 0
\end{aligned}$$

So ϕ must be zero. Hence

$$\Sigma \mathbf{a}_2^T = \lambda \mathbf{a}_2^T$$

$\mathbf{a}_2^T$ is also an eigenvector and has $\lambda \equiv \lambda_2$ as its eigenvalue. And, again, $var[\mathbf{z}_2] = \mathbf{a}_2 \Sigma \mathbf{a}_2^T = \lambda_2$ is the second largest eigenvalue.

In general,

$$var[\mathbf{z}_k] = \mathbf{a}_k \Sigma \mathbf{a}_k^T = \lambda_k$$

The k^{th} largest eigenvalue of Σ is the variance of the k^{th} PC.

11 K-means clustering

Given N data points $\{x_n\}_{n=1}^N \subset \mathbb{R}^D$. Initialize K prototype vectors $\{\mu_k\}_{k=1}^K$. Each μ_k corresponds to the mean of the k^{th} cluster. Let r_{nk} be indicator variable with respect to x_n and μ_k.

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min \|x_n - \mu_k\|^2 \\ 0 & \text{otherwise} \end{cases}$$

Then update μ_k,

$$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}$$

Keep this procedure until

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|x_n - \mu_k\|^2$$

converge.

12 Mixture of Gaussian(Soft Clustering)

Give probabilities that an instance belongs to each cluster instead of assigning only one. Let $g(x; m, \sigma)$ be the probability of a point x based on a Gaussian Distribution with mean m and variance σ. Suppose there is a distribution generated by randomly selecting one of K Gaussians. We randomly draw a point from this distribution. Let p_k be the probability of choosing the k^{th} Gaussian. Then

$$p(x) = \sum_{k=1}^{K} p_k g(x; m_k, \sigma_k)$$

Now we want to find p_k, σ_k and m_k that maximize $p(x)$.
E step: compute the probability that point x_n is generated by distribution k, i stands for steps

$$p^{(i)}(k|x_n) = \frac{p_k^{(i)} g(x_n; m_k^{(i)}, \sigma_k^{(i)})}{\sum_{j=1}^{K} p_j^{(i)} g(x_n; m_j^{(i)}, \sigma_j^{(i)})}$$

M step: update p_k, σ_k and m_k.

$$m_k^{(i+1)} = \frac{\sum_{n=1}^{N} p^{(i)}(k|x_n) x_n}{\sum_{n=1}^{N} p^{(i)}(k|x_n)}$$

$$\sigma_k^{(i+1)} = \sqrt{\frac{\sum_{n=1}^N p^{(i)}(k|x_n)\|x_n - n_k^{(i+1)}\|^2}{\sum_{n=1}^N p^{(i)}(k|x_n)}}$$
$$p_k^{(i+1)} = \frac{1}{N}\sum_{n=1}^N p^{(i)}(k|x_n)$$

13 Information Theory

13.1 Information

Core concept:

> "Highly improbable events bring more information to us while certain events bring no information."

The information of an event x will therefore depends on its probability distribution $p(x)$. Let $h(\cdot)$ be the monotonic function of $p(x)$ that returns information. If x and y are unrelated events, we hope the information they take are also unrelated, so

$$h(x,y) = h(x) + h(y)$$
$$p(x,y) = p(x)p(y)$$

Note that we can interpret $h(p(x))$ as $h(x)$ and $h(p(x,y))$ as $h(x,y)$. This means we can define

$$h(x) = -\log_2 p(x)$$

Then $h(x)$ satisfies $2^{h(x)} = 1/p(x)$. We can interpret this as

> "$h(x)$ is the amount of bits that being enough for representing $1/p(x)$ in binary."

13.2 Entropy

Let x be the state that transmitted from a sender to a receiver. Intuitively, the average amount of the information that x carries is obtained by taking the expectation of information $h(x)$ with respect to the p.d.f. $p(x)$

$$\sum_x p(x)h(x) = -\sum_x p(x)\log_2 p(x)$$

This is called the *entropy* of the random variable x and denote it by $\mathrm{H}[x]$. Since $\lim_{p\to 0} p\ln p = 0$, we just take $p(x)\ln p(x) = 0$ when we encounter $p(x) = 0$ for some x. The *noiseless coding theorem*(not actually understand what the hell is this) states that entropy is a lower bound of the amount of bits that a random variable can transmits.

In practice, we use $\ln p(x)$ instead of $\log_2 p(x)$. That is

$$h(x) = -\ln p(x)$$
$$\mathrm{H}[x] = -\sum_x p(x)\ln p(x)$$

In this situation, we said the information is measured in the units of 'nats'.

13.3 Maximize Entropy in Discrete Case

Let $X = \{x_i\}_{i=1}^{M}$ be a discrete random variable and let p be the distribution of X. For the problem

$$\max \left(-\sum_i p(x_i) \ln p(x_i) \right) \quad \textit{subject to} \quad \sum_i p(x_i) = 1$$

The Lagrangian function is

$$\tilde{\mathrm{H}} = -\sum_i p(x_i) \ln p(x_i) + \lambda \left(\sum_i p(x_i) - 1 \right)$$

From $\partial \tilde{H} / \partial p(x_k) = -(\ln p(x_k) + 1) + \lambda = 0$, we have $\lambda = \ln p(x_k) + 1$, $\forall\ k = 1, \ldots, M$. Since $\sum_k p(x_k) = M \cdot e^{\lambda - 1} = 1$, $\lambda = \ln(1/M) + 1$. We have

$$p(x_k) = 1/M,\ \forall\ k$$

The entropy becomes $\mathrm{H}[x] = \ln M$.

13.4 Differential Entropy(Entrpoy in Continuous Case)

Let X be a continuous random variable and p be the distribution of X. By M.V.T, we know there exists some x_i such that

$$\int_{i\Delta}^{(i+1)\Delta} p(x)dx = p(x_i)\Delta$$

where Δ is the length of one partition of X. Now for any $x \in [i\Delta, (i+1)\Delta]$, we can use $p(x_i)\Delta$ to estimate its probability as long as Δ small enough. Here comes an entropy

$$\begin{aligned}
\mathrm{H}_\Delta &= -\sum_i p(x_i)\Delta \ln(p(x_i)\Delta) \\
&= -\sum_i p(x_i)\Delta \ln(p(x_i)\Delta) + \sum_i p(x_i)\Delta \ln \Delta - \sum_i p(x_i)\Delta \ln \Delta \\
&= -\sum_i p(x_i)\Delta \ln p(x_i) - \ln \Delta
\end{aligned}$$

Note that $\sum_i p(x_i)\Delta = 1$. Take out the first term of right hand side,

$$\lim_{\Delta \to 0} -\sum_i p(x_i)\Delta \ln p(x_i) = -\int p(x) \ln p(x) dx$$

This integral is called the *differential entropy*. The equation between H_Δ and the differential entropy shows the fact the we need lots of bits to describe a continuous variable.

When it comes to multiple dimension we also have

$$\mathrm{H}[\mathbf{x}] = -\int p(\mathbf{x}) \ln p(\mathbf{x}) d\mathbf{x}$$

13.5 Gaussian Maximize Differential Entropy

We want to maximize

$$\mathrm{H}[x] = -\int p(x)\ln p(x)dx$$

with three constraints

$$\int_{-\infty}^{\infty} p(x)dx = 1$$
$$\int_{-\infty}^{\infty} xp(x)dx = \mu$$
$$\int_{-\infty}^{\infty} (x-\mu)^2 p(x)dx = \sigma^2$$

Here comes the Lagrangian function

$$-\int_{\mathbb{R}} p(x)\ln p(x)dx + \lambda_1\left(\int_{\mathbb{R}} p(x)dx - 1\right) + \lambda_2\left(\int_{\mathbb{R}} xp(x)dx - \mu\right) + \lambda_3\left(\int_{\mathbb{R}} (x-\mu)^2 p(x)dx - \sigma^2\right)$$

This is a functional $F[p]$. The derivative of a functional is denoted by $\frac{\delta F}{\delta p}$ and is defined to satisfy

$$\int \frac{\delta F[p]}{\delta p}\phi(x)dx = \lim_{\epsilon\to 0}\frac{F[p+\epsilon\phi] - F[p]}{\epsilon} = \left[\frac{d}{d\epsilon}F[p+\epsilon\phi]\right]_{\epsilon=0}$$

where $\phi(x)$ is a variation term. Followed by the definition

$$\int \frac{\delta F[p]}{\delta p}\phi(x)dx = \int \left(-(\ln p(x)+1) + \lambda_1 + \lambda_2 x + \lambda_3(x-\mu)^2\right)\phi(x)dx$$

We have the actual form of $\frac{\delta F[p]}{\delta p}$ and can let it be zero then get

$$p(x) = e^{-1+\lambda_1+\lambda_2 x+\lambda(x-\mu)^2}$$

Substitute this result back to three constraints leading to

$$p(x) = \frac{1}{(2\pi\sigma^2)^{1/2}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Furthermore,

$$\mathrm{H}[x] = \frac{1}{2}\{1+\ln(2\pi\sigma^2)\}$$

13.6 Kullback-Leibler Divergence

Let $p(x)$ be an unknown distribution and we use $q(x)$ to approximate it. This will cause additional amount of information when transmitting

$$\begin{aligned}\mathrm{KL}(p\|q) &= \left(-\int p(\mathbf{x})\ln q(\mathbf{x})d\mathbf{x}\right) - \left(-\int p(\mathbf{x})\ln p(\mathbf{x})d\mathbf{x}\right)\\ &= -\int p(\mathbf{x})\ln\left\{\frac{q(\mathbf{x})}{p(\mathbf{x})}\right\}d\mathbf{x}\end{aligned}$$

This is known as the *relative entropy* or *Kullback-Leiber divergence*, or *KL divergence* between the distributions $p(\mathbf{x})$ and $q(\mathbf{x})$. Note that $\mathrm{KL}(p\|q) \neq \mathrm{KL}(q\|p)$.

14 EM Algorithm

The goal of the EM algorithm is to find maximum likelihood solutions for models having latent variables. Consider a probabilistic model in which we collectively denote all of the observed variabels by $\mathbf{X}$ and all of the hidden variables by $\mathbf{Z}$. The joint distribution $p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$ is governed by a set of parameters denoted $\boldsymbol{\theta}$. Our goal is to maximize the likelihood function that is given by

$$p(\mathbf{X}|\boldsymbol{\theta}) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$$

This method based on the assumption that the computation of $p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$ is easier than $p(\mathbf{X}|\boldsymbol{\theta})$. Let $q(\mathbf{Z})$ be the distribution of $\mathbf{Z}$. Then is obvious that

$$\ln p(\mathbf{X}|\boldsymbol{\theta}) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln p(\mathbf{X}|\boldsymbol{\theta})$$

We've known that

$$\ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}) = \ln p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}) + \ln p(\mathbf{X}|\boldsymbol{\theta})$$

Then

$$\begin{aligned}
\ln p(\mathbf{X}|\boldsymbol{\theta}) &= \sum_{\mathbf{Z}} \ln q(\mathbf{Z}) p(\mathbf{X}|\boldsymbol{\theta}) \\
&= \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})}{q(\mathbf{Z})} \right\} - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \right\} \\
&\equiv \mathcal{L}(q, \boldsymbol{\theta}) + \mathrm{KL}(q\|p)
\end{aligned}$$

Since $\mathrm{KL}(q\|p) \geq 0$, $\mathcal{L}(q, \boldsymbol{\theta})$ is the lower bound of $\ln p(\mathbf{X}|\boldsymbol{\theta})$, i.e.

$$\mathcal{L}(q, \boldsymbol{\theta}) \leq \ln p(\mathbf{X}|\boldsymbol{\theta})$$

Note that $\mathrm{KL}(q\|p) = 0$ if and only if $q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta})$.

The EM algorithm is a two-stage iterative optimization technique for finding maximum likelihood solutions.

14.1 E Step

Let $\boldsymbol{\theta}^{\mathrm{old}}$ be the current parameter vector. Maximize $\mathcal{L}(q, \boldsymbol{\theta})$ with respect to $q(\mathbf{Z})$ with holding $\boldsymbol{\theta}^{\mathrm{old}}$ fixed. $\mathcal{L}(q, \boldsymbol{\theta})$ will reach its maximum $\ln p(\mathbf{X}|\boldsymbol{\theta})$ when $q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\mathrm{old}})$. That is,

$$\max \mathcal{L}(q, \boldsymbol{\theta}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\mathrm{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}^{\mathrm{old}}) - \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\mathrm{old}}) \ln p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\mathrm{old}})$$

14.2 M Step

Maximize $\mathcal{L}(q, \boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ with holding $q(\mathbf{Z})$ fixed. Here we get new parameter $\boldsymbol{\theta}^{\mathrm{new}}$. This will cause $\mathcal{L}(q, \boldsymbol{\theta})$ increase and then cause $\ln p(\mathbf{X}|\boldsymbol{\theta})$ increase subsequently. Furthermore, the $q(\mathbf{Z})$ in this step is determined in the previous step, $\mathrm{KL}(q, \boldsymbol{\theta}^{\mathrm{new}})$ will be nonzero. Hence $\ln p(\mathbf{X}|\boldsymbol{\theta})$ increases more than its lowe bound $\mathcal{L}(q, \boldsymbol{\theta})$. From above we have,

$$\mathcal{L}(q, \boldsymbol{\theta}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\mathrm{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}) - \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\mathrm{old}}) \ln p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\mathrm{old}})$$

$$= \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) + \text{constant}$$

Since $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) = q(\mathbf{Z})$, the last term is a constant. This tells us that we actually maximize the expectation of the complete-data log likelihood.

15 Optimization

15.1 Hessian Matrix

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a twice differentiable function, its Hessian matrix at $x \in \mathbb{R}^n$ is defined as

$$\nabla^2 f(x) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 x_2} & \frac{\partial^2 f f}{\partial x_1 x_3} & \cdots & \frac{\partial^2 f}{\partial x_1 x_n} \\ \frac{\partial^2 f}{\partial x_2 x_1} & \frac{\partial^2 f}{\partial x_2^2} & \frac{\partial^2 f}{\partial x_2 x_3} & \cdots & \frac{\partial^2 f}{\partial x_2 x_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n x_1} & \frac{\partial^2 f}{\partial x_n x_2} & \frac{\partial^2 f}{\partial x_n x_3} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix} \in \mathbb{R}^{n \times n}$$

15.1.1 Quadratic Function

Let $f : \mathbb{R}^n \to \mathbb{R}$ and $f(x) = \frac{1}{2} x^T H x + p^T x$ where $H \in \mathbb{R}^{n \times n}$ is a symmetric matrix and $p \in \mathbb{R}^n$. Then

$$\begin{aligned} \nabla f(x) &= Hx + p \\ \nabla^2 f(x) &= H \end{aligned}$$

If H is positive definite, then $x^* = -H^{-1}p$ is a unique solution.

15.1.2 Least-Square Problem

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2, \ A \in \mathbb{R}^{m \times n}, \ b \in \mathbb{R}^m$$

$$\begin{aligned} f(x) &= (Ax - b)(Ax - b)^T \\ &= x^T A^T A x - 2b^T A x + b^T b \\ \nabla f(x) &= 2A^T A x - 2A^T b \\ \nabla^2 f(x) &= 2A^T A \\ x^* &= (A^T A)^{-1} A^T b \end{aligned}$$

15.1.3 Newton's Method

Compute d^i satisfies

$$\nabla^2 f(x^i) d^i = -\nabla f(x^i)$$

Update

$$x^{i+1} = x^i + d^i$$

Until $\nabla f(x^i) = 0$.

15.2 Lagrangian Function

For the problem

$$\begin{aligned} &\min_{x\in\Omega} f(x) \\ &s.t. \quad g_i(x) \le 0,\ h_i(x) = 0,\ \forall\ i = 1,\dots,m \end{aligned}$$

Write $g(x) = (g_1(x),\dots,g_m(x))^T$ and $h(x) = (h_1(x),\dots,h_m(x))^T$. Let

$$\mathcal{L}(x,\alpha,\beta) = f(x) + \alpha^T g(x) + \beta^T h(x) \text{ and } \alpha \in \mathbb{R}^m \ge 0$$

For a fixed $\alpha \ge 0$ and a fixed β, if $\bar{x} \in \arg\min\{\mathcal{L}(x,\alpha,\beta) | x \in \mathbb{R}^n\}$ then

$$\left.\frac{\partial \mathcal{L}(x,\alpha,\beta)}{\partial x}\right|_{x=\bar{x}} = \nabla f(\bar{x}) + \alpha^T \nabla g(\bar{x}) + \beta^T \nabla h(\bar{x}) = 0$$

15.2.1 Duality

Consider

$$\max_{\alpha,\beta} \min_{x\in\Omega} \mathcal{L}(x,\alpha,\beta) \text{ s.t. } \alpha \ge 0$$

Let $\theta(\alpha,\beta) = \inf_{x\in\Omega} \mathcal{L}(x,\alpha,\beta)$. Then

$$\max_{\alpha,\beta} \theta(\alpha,\beta) \text{ s.t. } \alpha \ge 0$$

For any $\bar{x} \in \Omega$ and $(\bar{\alpha} \ge 0, \bar{\beta})$ be solutions of the above problems, respectively, since $\bar{\alpha}^T g(\bar{x}) \le 0$ and $h(\bar{x}) = 0$, we have $f(\bar{x}) \ge \theta(\bar{\alpha},\bar{\beta})$.

Theorem 15.1. *If the equality happens, the $\bar{x}$ and $(\bar{\alpha} \ge 0, \bar{\beta})$ solve the primal and dual problem, respectively. In this case,*

$$\mathbf{0} \le \bar{\alpha} \perp g(x) \le \mathbf{0}$$

Furthermore, for these $\bar{x}$ and $(\bar{\alpha},\bar{\beta})$,

$$\begin{aligned} f(\bar{x}) &= \theta(\bar{\alpha},\bar{\beta}) \\ &= \inf_{x\in\Omega}\{f(x) + \bar{\alpha}^T g(x) + \bar{\beta}^T h(x)\} \\ &\le f(\bar{x}) + \bar{\alpha}^T g(\bar{x}) + \bar{\beta}^T h(\bar{x}) \\ &= f(\bar{x}) + \bar{\alpha}^T g(\bar{x}) \\ &\le f(\bar{x}) \end{aligned}$$

This implies that

$$\bar{\alpha}^T g(\bar{x}) = 0$$

15.2.2 Karush-Kuhn-Tucker Condition(KKT Condition)

This is a summary of solve both primal form and dual form. Find $\bar{x} \in \Omega$, $\bar{\alpha},\bar{\beta} \in \mathbb{R}^m$ such that

$$\begin{aligned} \textbf{Stationarity} \quad & \nabla f(\bar{x}) + \bar{\alpha}^T \nabla g(\bar{x}) + \bar{\beta}^T \nabla h(\bar{x}) = 0 \\ \textbf{Complmentary Slackness} \quad & \bar{\alpha}^T g(\bar{x}) = 0 \\ \textbf{Primal Feasibility} \quad & h(\bar{x}) = 0,\ g(\bar{x}) \le 0 \\ \textbf{Dual Feasibility} \quad & \bar{\alpha} \ge 0 \end{aligned}$$

15.2.3 Dual Linear Problem

For the primal linear problem

$$\min_{x\in\mathbb{R}^n} p^T x \quad \text{s.t.} \quad Ax \geq b,\ x \geq 0$$

Consider

$$\max_{\alpha_1,\alpha_2\geq \mathbf{0}} \min_{x\in\mathbb{R}^n} \mathcal{L}(x,\alpha,\beta) = p^T x + \alpha_1^T(b - Ax) + \alpha_2^T(-x)$$

When $\bar{x} \in \arg\min\{\mathcal{L}(x,\alpha,\beta)|x \in \Omega\}$, the gradient $\nabla\mathcal{L}(\bar{x},\alpha,\beta)$ vanish

$$p - A^T\alpha_1 - \alpha_2 = 0$$

Then we have the dual problem

$$\max_{\alpha_1,\alpha_2\geq \mathbf{0}} b^T\alpha_1 \quad \text{s.t.} \quad p - A^T\alpha_1 - \alpha_2 = 0$$

Since α_2 is a slack variable, it's equivalent to

$$\max b^T\alpha \quad \text{s.t.} \quad A^T\alpha \leq p,\ \alpha \geq 0$$

15.2.4 Least Square Problem

For $\min_{x\in\mathbb{R}^n} \|Ax - b\|_2^2$ where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. It's obvious that

$$x^* \in \left\{ \arg\min\{\|Ax - b\|_2^2\} \right. \Rightarrow A^T Ax = A^T b$$

Consider the problem

$$\min \mathbf{0}^T x \quad \text{s.t.} \quad A^T Ax = A^T b$$

Then for

$$\max_{\alpha} \min \mathbf{0}^T x + \alpha^T(A^T b - A^T Ax) \quad \text{s.t.} \quad \alpha \in \mathbb{R}^m$$

and the dual problem

$$\max_{\alpha} b^T A\alpha \quad \text{s.t.} \quad (A^T A)^T\alpha = \mathbf{0}$$

The constraint has a trivial solution $\alpha = \mathbf{0}$ and the objective function has value 0. The objective function of the primal problem and the dual problem have the same value. This implies that $A^T Ax = A^T b$ must have a solution. Otherwise the dual problem won't have optimal solution.

15.2.5 Quadratic Problem

The primal problem

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}x^T Qx + p^T x \quad \text{s.t.} \quad Ax \leq b$$

For

$$\max_{\alpha\geq 0} \min_{x\in\mathbb{R}^n} \frac{1}{2}x^T Qx + p^T x + \alpha^T(Ax - b) \quad \text{s.t.} \quad \alpha \geq 0$$

the gradient needs vanishing

$$Qx + p + A^T\alpha = 0 \Rightarrow x = -Q^{-1}(A^T\alpha + p)$$

Substitute back and we have the dual form

$$\max -\frac{1}{2}(p^T + \alpha^T A)Q^{-1}(A^T\alpha + p) - \alpha^T b \quad \text{s.t.} \quad \alpha \geq \mathbf{0}$$

16 Support Vector Machine

Suppose there are N data points $\mathbf{x}_1, \ldots, \mathbf{x}_N$ and let $t_1, \ldots, t_N$ be their labels where $t_k \in \{-1, 1\}$ for all k. We want to find suitable $\phi(\mathbf{x}_k)$, weight $\mathbf{w}$ and bias b that satisfies

$$\mathbf{w}^T\phi(\mathbf{x}_k) + b = \begin{cases} > 0 & \text{if } t_k = 1 \\ < 0 & \text{if } t_k = -1 \end{cases} \qquad \text{for all } k$$

This is a linear classifier in feature space. The distance of each data point to this hyper plane is

$$\frac{|\mathbf{w}^T\phi(\mathbf{x}_k) + b|}{\|\mathbf{w}\|}$$

According to *statistical learning theory*(not actually understand), we want to maximize the minimal distance

$$\arg\max_{\mathbf{w},b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_k |\mathbf{w}^T\phi(\mathbf{x}_k) + b| \right\}$$

Since $t_k(\mathbf{w}^T\phi(\mathbf{x}_k) + b) > 0$ and $t_k \in \{-1, 1\}$, the problem is equivalent to

$$\arg\max_{\mathbf{w},b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_k t_k(\mathbf{w}^T\phi(\mathbf{x}_k) + b) \right\}$$

Observe that any rescaling of $\mathbf{w}$ and b won't change the ultimate value, hence we can set

$$t_k(\mathbf{w}^T\phi(\mathbf{x}_k) + b) = 1$$

for those $\mathbf{x}_k$ that are closest to the hyper plane. Then other points yield

$$t_k(\mathbf{w}^T\phi(\mathbf{x}_k) + b) \geq 1$$

Now we simplify the problem into

$$\arg\max_{\mathbf{w},b} \frac{1}{\|\mathbf{w}\|} \quad \text{s.t.} \quad t_k(\mathbf{w}^T\phi(\mathbf{x}_k) + b) \geq 1 \text{ for all } k$$

It's equivalent to

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{s.t.} \quad t_k(\mathbf{w}^T\phi(\mathbf{x}_k) + b) \geq 1 \text{ for all } k$$

And gives the Lagrangian function

$$\mathcal{L}(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{k=1}^{N} a_k\big[1 - t_k(\mathbf{w}^T\phi(\mathbf{x}_k) + b)\big]$$

where $\mathbf{a} = (a_1, \ldots, a_N)^T \geq \mathbf{0}$. For

$$\max_{\mathbf{a}} \min_{\mathbf{w},b} \mathcal{L}(\mathbf{w}, b, \mathbf{a}) \quad \text{s.t.} \quad \mathbf{a} \geq 0$$

let gradient vanish with respect to $\mathbf{w}$ and b

$$\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}, b, \mathbf{a}) = \mathbf{w} - \sum_{k=1}^{N} a_k t_k \phi(\mathbf{x}_k) = 0$$

$$\nabla_b \mathcal{L}(\mathbf{w}, b, \mathbf{a}) = \sum_{k=1}^{N} a_k t_k = 0$$

Substitute back then yield the dual form

$$\max_{\mathbf{a}} \sum_{k=1}^{N} a_k - \frac{1}{2} \sum_{k=1}^{N} \sum_{m=1}^{N} a_k a_m t_k t_m \phi(\mathbf{x}_k)^T \phi(\mathbf{x}_m)$$

subject to

$$a_k \geq 0, \quad k = 1, \ldots, N$$
$$\sum_{k=1}^{N} a_k t_k = 0.$$

Let $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$ stands for the kernel function and let $y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b$. When $\mathbf{w}$ is the solution, $\mathbf{w} = \sum_k a_k t_k \phi(\mathbf{x}_k)$. Put this into $y(\mathbf{x})$ and have

$$y(\mathbf{x}) = \sum_{k=1}^{N} a_k t_k k(\mathbf{x}_k, \mathbf{x}) + b$$

here we express the classifier in terms of $\{a_k\}$ and the kernel function $k(\mathbf{x}, \mathbf{x}')$.

16.1 Apply KKT Condition

We have a primal problem

$$\arg\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{s.t.} \quad t_k y(\mathbf{x}_k) - 1 \geq 0 \text{ for all } k$$

and a dual problem

$$\max_{\mathbf{a}} \sum_{k=1}^{N} a_k - \frac{1}{2} \sum_{k=1}^{N} \sum_{m=1}^{N} a_k a_m t_k t_m k(\mathbf{x}_k, \mathbf{x}_m) \quad \text{s.t.} \quad a_k \geq 0, \ \forall \ k \text{ and } \sum_{k=1}^{N} a_k t_k = 0$$

The KKT condition needs further constraint that

$$a_k \{t_k y(\mathbf{x}_k) - 1\} = 0$$

This implies $a_k = 0$ or $t_k y(\mathbf{x}_k) = 1$. Any data point has $a_k = 0$ will not contribute to the classifier. The rest data points have $t_k y(\mathbf{x}_k) = 1$ are called *support vector*.This tells us that we only need support vectors to predict new point though we need whole data to train. Mathematically, choose one support vector $\mathbf{x}'$, we can get b by

$$t' y(\mathbf{x}') = t' \left\{ \sum_m a_m t_m k(\mathbf{x}_m, \mathbf{x}') + b \right\} = 1$$

where m stands for the index of support vector. In practical, multiply each side by one label t_k of one support vector and have

$$b = t_k - \left\{ \sum_m a_m t_m k(\mathbf{x}_m, \mathbf{x}_k) \right\} \text{ for all } k$$

Take the average of all possible b as the final one

$$b = \frac{1}{M}\sum_k \left(t_k - \left\{ \sum_m a_m t_m k(\mathbf{x}_m, \mathbf{x}_k) \right\} \right)$$

where M is the number of support vectors and k and m are both the index of support vector.

17 Soft Margin Support Vector Machine

The original SVM is

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{s.t.} \quad t_k y(\mathbf{x}_k) \geq 1 \text{ for all } k$$

To allow some points can be misclassified, we introduce each point a *slack variable* ξ_k that is defined by

$$\xi_k = \begin{cases} 0 & \text{if } t_k y(\mathbf{x}_k) \geq 1 \\ |t_k - y(\mathbf{x}_k)| & \text{otherwise} \end{cases}$$

Look deeper into this definition. If $0 < t_k y(\mathbf{x}_k) < 1$, we have $0 < y(\mathbf{x}_k) < 1$ or $-1 < y(\mathbf{x}_k) < 0$. Hence $0 < \xi_k = 1 - t_k y(\mathbf{x}_k) < 1$. If $t_k y(\mathbf{x}_k) = 0$, $\xi_k = 1 - t_k y(\mathbf{x}_k) = 1$. If $t_k y(\mathbf{x}_k) < 0$, $\xi_k = 1 - t_k y(\mathbf{x}_k) > 1$. In summary

$$\xi_k \begin{cases} = 0 & \text{if } t_k y(\mathbf{x}_k) \in [1, \infty) \\ = 1 - t_k y(\mathbf{x}_k) & \begin{cases} \in (0,1) & \text{if } t_k y(\mathbf{x}_k) \in (0,1) \\ = 1 & \text{if } t_k y(\mathbf{x}_k) = 0 \\ > 1 & \text{if } t_k y(\mathbf{x}_k) \in (-\infty, 0) \end{cases} \end{cases}$$

Replace the constrain with

$$t_k y(\mathbf{x}_k) \geq 1 - \xi_k \text{ for all } k$$

This is so called *soft margin*.

When there exist outliers, they'll have extremely large ξ_k. To avoid this, here comes the soft SVM that also minimize slack vairable

$$\min_{\mathbf{w},b,\xi} C\sum_{k=1}^{N} \xi_k + \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{s.t.} \quad t_k y(\mathbf{x}_k) \geq 1 - \xi_k \text{ for all } k$$

where C is some constant. Briefly speaking, slack vairables relax some points' constarint, their $t_k y(\mathbf{x}_k)$ only needs to larger than some value smaller than 1. That's why we call this *soft margin*.

17.1 Apply KKT Condition

The Lagrangian of soft SVM is

$$\mathcal{L}(\mathbf{w}, b, \xi, \mathbf{a}, \mu) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{k=1}^{N} \xi_k - \sum_{k=1}^{N} a_k\{t_k y(\mathbf{x}_k) - 1 + \xi_k\} - \sum_{k=1}^{N} \mu_k \xi_k$$

where $a_k, \mu_k \geq 0$, $t_k y(\mathbf{x}_k) - 1 + \xi_k \geq 0$ and note that slack variables are non negative $\xi_k \geq 0$. The KKT conditions are

Dual Feasibility	$a_k \geq 0$,	$\mu_k \geq 0$
Primal Feasibility	$t_k y(\mathbf{x}_k) - 1 + \xi_k \geq 0$,	$\xi_k \geq 0$
Complmentary Slackness	$a_k(t_k y(\mathbf{x}_k) - 1 + \xi_k) = 0$,	$\mu_k \xi_k = 0$

Use $y(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) + b$ and compute gradients of $\mathcal{L}$ with respect to $\mathbf{w}$, b and ξ

$$\nabla_{\mathbf{w}}\mathcal{L} = \mathbf{w} - \sum_{k=1}^{N} a_k t_k \phi(\mathbf{x}_n) = 0$$
$$\nabla_b \mathcal{L} = \sum_{k=1}^{N} a_k t_k = 0$$
$$\nabla_{\xi_k}\mathcal{L} = a_k - C - \mu_k = 0$$

Substitute back and get the dual form

$$\max_{\mathbf{a}} \sum_{k=1}^{N} a_k - \frac{1}{2}\sum_{k=1}^{N}\sum_{m=1}^{N} a_k a_m t_k t_m k(x_k, x_m) \quad \text{s.t.} \quad 0 \le a_k \le C,\ \forall\ k \text{ and } \sum_{k=1}^{N} a_k t_k = 0$$

It's the same as normal SVM, the only difference is the constraint of a_k, which is known as the *box constraint*.

18 Kalman Filter

19 References

[1] **Hung-Yi Lee, National Taiwan University.**
`http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML20.html`

[2] **Yuh-Jye Lee, National Chiao Tung University.**
`https://ocw.nctu.edu.tw/course_detail.php?bgid=1&gid=1&nid=563`

[3] **Christopher Bishop.** *Pattern Recognition and Machine Learning,* 2006

[4] **Aaron Courville, Ian Goodfellow, and Yoshua Bengio.** *Deep Learning,* 2015

[5] **Sebastian Ruder.** *An overview of gradient descent optimization alogrithms.*
`https://ruder.io/optimizing-gradient-descent/index.html`