Introduction to Machine Learning

October 30, 2018

1 Lecture 1

linear function

$$y = f_{\mathbf{W}}(\mathbf{X}) = f(\mathbf{X}, \mathbf{W}) = \mathbf{W}^{\mathrm{T}}\mathbf{X}$$

linear classifier (perception model)

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq 0$$

$$\mathbf{x}_i \cdot \mathbf{w} + b < 0$$

linear regression in 1 dimension

$$y^i = \mathbf{W}^{\mathrm{T}}\mathbf{X}^i + \epsilon^i$$

where ϵ is the noise(loss).

Loss function: sum of squared errors

$$L(\mathbf{W}) = \sum_{i=1}^{N} (\epsilon^i)^2$$

$$L(w_0, w_1) = \sum_{i=1}^{N}$$

$$\frac{\partial L(w_0, w_1)}{\partial w_0} = \sum_{i=1}^{N} \frac{\partial [y^i - (w_0 x_0^i + w_1 x_1^i)]^2}{\partial w_0} = -2 \sum_{i=1}^{N} (y^i - (w_0 x_0^i + w_1 x_1^i)) x_0^i = 0$$

$$\sum_{i=1}^{N} y^i x_0^i = w_0 \sum_{i=1}^{N} x_0^i x_0^i + w_1 \sum_{i=1}^{N} x_1^i x_0^i$$

as follow, the partial gradient of w_1 would be

$$\sum_{i=1}^{N} y^i x_1^i = w_0 \sum_{i=1}^{N} x_0^i x_1^i + w_1 \sum_{i=1}^{N} x_1^i x_1^i$$

Therefore

$$\begin{bmatrix} \sum_{i=1}^{N} y^i x_0^i \\ \sum_{i=1}^{N} y^i x_1^i \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} x_0^i x_0^i & \sum_{i=1}^{N} x_0^i x_1^i \\ \sum_{i=1}^{N} x_0^i x_1^i & \sum_{i=1}^{N} x_1^i x_1^i \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} \tag{1}$$

Formally, it could conclude that

$$\mathbf{X}^{\mathrm{T}}\mathbf{y} = \mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{w}$$

$$\mathbf{w} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$$

Here still need to add the trace version(more generalized version):

$$\begin{aligned}
\hat{\mathbf{Y}} &= \mathbf{X}\mathbf{w} \\
Loss &= (\mathbf{Y} - \mathbf{X}\mathbf{w})^{\mathbf{2}} \\
&= (\mathbf{Y} - \mathbf{X}\mathbf{w})^{\mathbf{T}}(\mathbf{Y} - \mathbf{X}\mathbf{w}) \\
&= \mathbf{Y}^{\mathbf{T}}\mathbf{Y} - \mathbf{w}^{\mathbf{T}}\mathbf{X}^{\mathbf{T}}\mathbf{Y} - \mathbf{Y}^{\mathbf{T}}\mathbf{X}\mathbf{w} + \mathbf{w}^{\mathbf{T}}\mathbf{X}^{\mathbf{T}}\mathbf{X}\mathbf{w} \\
Tr[\frac{\partial}{\partial \mathbf{w}} Loss] &= -\mathbf{X}^{\mathbf{T}}\mathbf{Y} - \mathbf{X}^{\mathbf{T}}\mathbf{Y} + \mathbf{X}^{\mathbf{T}}\mathbf{X}\mathbf{w} + \mathbf{X}^{\mathbf{T}}\mathbf{X}\mathbf{w} \\
&= 0 \\
\mathbf{w} &= (X^T X)^{-1} X^T Y
\end{aligned}$$

Due to the matrix derivatives:

$$\begin{aligned}
\mathrm{Tr[ABC]} &= \mathrm{Tr[CAB]} \\
\frac{\partial}{\partial \mathrm{A}} \mathrm{Tr[A^T B]} &= B \\
\frac{\partial}{\partial \mathrm{A}} \mathrm{Tr[A^T BAC]} &= BAC + B^T A C^T
\end{aligned}$$

Least squares solution, vector form

$$\begin{aligned}
L(\mathbf{w}) &= (\mathbf{y} - \mathbf{X}\mathbf{w})^t(\mathbf{y} - \mathbf{X}\mathbf{w}) \\
1 &= 2
\end{aligned}$$

Why the gradient could be equal to 0

Hessian matrix is a square matrix of second-order partial derivatives of scalar-valued function, or scalar field.

$$\mathbf{H}(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \tag{2}$$

$$\mathbf{H} = \begin{vmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial x^2} \end{vmatrix}$$

In the 2 dimension, when $\mathbf{H} > 0$: if $\frac{\partial^2 f}{\partial x^2} > 0$, then point$(x_0, y_0)$ is the local min point. If $\frac{\partial^2 f}{\partial x^2} < 0$, then point$(x_0, y_0)$ is the local max point.
when $\mathbf{H} < 0$, then point(x_0, y_0) is the stationary point.
when $\mathbf{H} = 0$, second order cannot decide the point property, then consider it in higher order Taylor's Expansion.
In the example,$\mathbf{H} = 4(x_0^i)^2(x_1^i)^2 - 4(x_1^i x_0^i)(x_0^i x_1^i) = 0$, and $\frac{\partial^2 f}{\partial x^2} = 4(x_0^i)^2(x_1^i)^2 > 0$. Therefore, it is a local min point for the loss function.
In higher dimension space (multi-variables), $\mathbf{H}(f)$ should be a positive definite matrix($(\nabla \mathbf{x})^{\mathrm{T}} \mathbf{H}(f) \nabla \mathbf{x} \geq 0$ for any $\nabla \mathbf{x}$).
The more detail of Hessian matrix could look up Taylor expansion.

Generalized linear regression

$$L(\mathbf{w}) = \sum_{i=1}^{N} (y^i - \mathbf{w}^{\mathrm{T}} \phi(\mathrm{x}^i))^{\mathrm{T}}$$

where $\phi(\mathbf{x}^i)$ is a polynomial function for $\mathbf{x}^i$

normalization

L2 norm(euclidean) norm:

$$||\mathbf{w}||_2 = \sqrt{\sum_{d=1}^{D} w_d^2} = \sqrt{< \mathbf{w}, \mathbf{w} >}$$

L1 norm(manhattan) norm:

$$||\mathbf{w}||_1 = \sum_{d=1}^{D} |w_d|$$

Lp norm, $p > 1$:

$$||\mathbf{w}||_p = (\sum_{d=1}^{D} w_d^p)^{\frac{1}{p}}$$

Ridge regression: L2-regularized linear regression

$$\begin{aligned} L(\mathbf{w}) &= \epsilon^{\mathrm{T}}\epsilon + \lambda \mathbf{w}^{\mathrm{T}}\mathbf{w} = \mathbf{y}^{\mathrm{T}}\mathbf{y} - 2\mathbf{y}^{\mathrm{T}}\mathbf{X}\mathbf{w} + \mathbf{w}^{\mathrm{T}}(\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda \mathbf{I})\mathbf{w} \\ \nabla L(\mathbf{w}^*) &= 0 \\ \mathbf{w}^* &= (\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda \mathbf{I})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y} \end{aligned}$$

In some case, the matrix cannot be inversed, then add $\lambda\mathbf{I}$ to make it invertible..

invertible matrix

Lasso regression: L1-regularized linear regression

suitable for small sample, large dimension.

It could shrink some coefficient into 0, helpful for feature selection.

The optimization would be gradient descent, LARS, PGD.

Logistic regression

sigmoid function:

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

Given training set:$\{(\mathbf{x}^1, y^1), \ldots, (\mathbf{x}^N, y^N)\}, \mathbf{x} \in \mathbb{R}^D, y \in \{0, 1\}$ The ML function would be:

$$
\begin{aligned}
p(y^1, \ldots, y^n | \mathbf{x}^1, \ldots, \mathbf{x}^N) &= \prod_{i=1}^{N} P(y^i | \mathbf{x}^i) \\
&= \prod_{i=1}^{N} \sigma(\mathbf{w}^T \mathbf{x}^i)^{y^i} (1 - \sigma(\mathbf{w}^T \mathbf{x}^i))^{1-y^i} \\
logP(\mathbf{y}|\mathbf{X}; \mathbf{w}) &= \sum_{i=1}^{N} y^i log\sigma(\mathbf{w}^T \mathbf{x}^i) + (1 - y^i) log(1 - \sigma(\mathbf{w}^T \mathbf{x}^i))
\end{aligned}
$$

quadratic loss is trying to close the distance, while logistic is trying to close the classification.

multiple classes

Softmax function:

$$
P(y = c|\mathbf{x}; \mathbf{W}) = \frac{\exp(\mathbf{w}_c^T \mathbf{x})}{\sum_{c'=1}^{C} \exp(\mathbf{w}_{c'}^T \mathbf{x})} = g_c(\mathbf{x}, \mathbf{W})
$$

Likelihood function of training sample: $(\mathbf{y}^i, \mathbf{x}^i)$

$$
P(\mathbf{y}^i | \mathbf{x}^i; \mathbf{w}) = \prod_{c=1}^{C} (g_c(\mathbf{x}, \mathbf{W}))^{\mathbf{y}_c^i}
$$

Optimization criterion:

$$
L(\mathbf{W}) = -\sum_{i=1}^{N} \sum_{c=1}^{C} \mathbf{y}_c^i \log(g_c(\mathbf{x}, \mathbf{W}))
$$

Mapping data to higher-dimensional space

while the data cannot be linear detected, the method is mapping the data to higher-dimensional space.

$$
L(\mathbf{W}') = -\sum_{i=1}^{N} \sum_{c=1}^{C} \mathbf{y}_c^i \log(g_c(\phi(\mathbf{x}), \mathbf{W}))
$$

Optimization to loss function

Gradient-based optimization

$$
\begin{aligned}
\frac{\partial L(\mathbf{w})}{\partial w_k} &= -\sum_{i=1}^{N} \left[y^i \frac{1}{g(\mathbf{w}^T \mathbf{x}^i)} \frac{\partial g(\mathbf{w}^T \mathbf{x}^i)}{\partial w_k} + (1 - y^i) \frac{1}{1 - g(\mathbf{w}^T \mathbf{x}^i)} (-\frac{\partial g(\mathbf{w}^T \mathbf{x}^i)}{\partial w_k}) \right] \\
&= -\sum_{i=1}^{N} [y^i - g(\mathbf{w}^T \mathbf{x}^i)] \mathbf{x}_k^i
\end{aligned}
$$

This is for the non-linear system of binary classification.

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$
$$Initial: \mathbf{x_0}$$
$$Update: \mathbf{x}_{i+1} = \mathbf{x}_i - \alpha \nabla f(\mathbf{x_i})$$

It always works for **convex function**. But it is hard to set α. second-order methods (newton method) First order Taylor series approximation:

$$f(x) \approx f(a) + (x-a)f'(a) + e(x)$$

Second order Taylor series approximation:

$$\begin{aligned} f(x) &= f(a) + (x-a)f'(a) + \frac{1}{2}(x-a)^2 f''(a) + e(x) \\ q'(x) &= f'(x_i) + (x-x_i)f''(x_i) = 0 \\ x_{i+1} &= x_i - \frac{f'(x_i)}{f''(x_i)} \end{aligned}$$

For the higher dimension

$$\begin{aligned} f(\mathbf{x}) &= f(\mathbf{x}_i) + (\mathbf{x}-\mathbf{x}_i)\nabla f(\mathbf{x}_i) + \frac{1}{2}(\mathbf{x}-\mathbf{x}_i)^T \mathbf{H}(\mathbf{x}-\mathbf{x}_i) \\ \mathbf{H_{i,j}} &= \frac{\partial^2 f}{\partial x_i \partial x_j} \\ \nabla q(\mathbf{x}) &= 0 \\ \nabla f(\mathbf{x}_i) + (\mathbf{x}-\mathbf{x}_i)^T \mathbf{H}(\mathbf{x_i}) &= 0 \\ \mathbf{x}_{i+1} &= \mathbf{x}_i - (\mathbf{H}(\mathbf{x_i}))^{-1} \nabla f(\mathbf{x}_i) \end{aligned}$$

Here is the hessian matrix for logistic loss function:

$$\begin{aligned} \frac{\partial^2 L(\mathbf{w})}{\partial_{w_k} \partial_{w_j}} &= \frac{\partial(-\sum_{i=1}^N [y^i - g(\mathbf{w}^T\mathbf{x}^i)]\mathbf{x}_k^i)}{\partial w_j} \\ &= \sum_{i=1}^N \mathbf{x}_k^i \frac{\partial g(\mathbf{w}^T\mathbf{x}^i)}{\partial w_j} \\ &= \sum_{i=1}^N \mathbf{x}_k^i g(\mathbf{w}^T\mathbf{x}^i)(1 - g(\mathbf{w}^T\mathbf{x}^i))\mathbf{x}_j^i \end{aligned}$$

Perception

Given $f(x) = sign(wx + b)$ as perception which is a discriminant. The distance between any point x_0 and the boundary is $\frac{|w \cdot x_0 + b|}{||w||}$.
For the wrong classified data (x_i, y_i): $-y_i(w \cdot x_i + b) > 0$. Therefore, the distance between wrong classified data (x_i, y_i) and the boundary is $-\frac{y_i(w \cdot x_i + b)}{||w||}$.
Then the loss function would be

$$L(w, b) = -\sum_{x_i \in M} y_i(w \cdot x_i + b)$$

Where the $\frac{1}{||w||}$ is ignored. The reasons: 1) $||w||$ is only a scalar, which does not influence the vector w direction 2)The perception training end condition is that loss $L(w, b) = 0$, so the $||w||$ does not influence that.
For the training, the update would be:

$$w \leftarrow w + \eta y_i x_i$$

$$b \leftarrow b + \eta y_i$$

Dual Property

For fast calculation.

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i$$

$$b = \sum_{i=1}^{N} \alpha_i y_i$$

to be continued.

SVM

Given the Discriminant: $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$

Functional Margins

$$\mathbf{w}^T\mathbf{x}^i \gg 0, if y^i = 1$$
$$\mathbf{w}^T\mathbf{x}^i \ll 0, if y^i = -1$$
$$y^i(\mathbf{w}^T\mathbf{x}^i) \gg 0$$

According to that $\mathbf{w}$ **is vertical to the boundary**(Due to $\mathbf{w} \cdot \mathbf{x} = -b \quad \forall \mathbf{x}$) and $\mathbf{x} = \mathbf{x}_\perp + \gamma \frac{\mathbf{w}}{|\mathbf{w}|}$, the distance between the point and the boundary would be $\gamma = \frac{\mathbf{w^T x} + b}{|\mathbf{w}|}$

Support Vector

The vectors (cases) that define the hyperplane are the support vectors.
Here we define that, for the positive support vector, $\mathbf{w}^T\mathbf{x}_+ + b = +1$ and $\mathbf{w}^T\mathbf{x}_- + b = -1$
In that way, the margin could be given as follow

$$\frac{\mathbf{w}^T(\mathbf{x}_+ - \mathbf{x}_-)}{||\mathbf{w}||} = \frac{2}{||\mathbf{w}||}$$

Then we needs to maximum the margin.

$$\begin{aligned} \max_{\mathbf{w}} \frac{2}{||\mathbf{w}||} &\rightarrow \min_{\mathbf{w}} ||\mathbf{w}||^2 \\ \texttt{s.t.} \quad y^i(\mathbf{w}^T\mathbf{x}^i + b) &\geq 1 \quad \forall i \end{aligned}$$

Dual

$$\begin{aligned} \min \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha^i\alpha^j y^i y^j < \mathbf{x}^i, \mathbf{x}^j > \\ \texttt{s.t.} \quad y^i\left(\sum_{j=1}^{N} \alpha^j y^j < \mathbf{x}^i, \mathbf{x}^j > +b\right) \geq 1 \quad \forall i \end{aligned}$$

Penalty constant

$$\begin{aligned} \min_{\mathbf{w},\xi} ||\mathbf{w}||^2 + C\sum_{i=1}^{N} \xi^i \\ s.t. : y^i(\mathbf{w}^T\mathbf{x}^+ b) \geq 1 - \xi^i, \forall i \\ \xi^i \geq 0, \forall i \end{aligned}$$

When misclassification when $\xi > 1$.
$\sum_i \xi^i$: upper bound on number of errors.
C: Hyper-parameter.
Rewrite the first constraint: $y^i h_{\mathbf{w},b}(\mathbf{x}) \geq 1 - \xi^i$
Then combined all constraint:

$$\begin{aligned} \xi^i &= [1 - y^i h_{\mathbf{w},b}(\mathbf{x})]_+ \\ &= \max(1 - y^i h_{\mathbf{w},b}(\mathbf{x}), 0) \end{aligned}$$

Then the loss function would be:

$$\begin{aligned} L(\mathbf{w}) &= \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N} \max(1 - y^i h_{\mathbf{w},b}(\mathbf{x}), 0) \\ &\propto \lambda||\mathbf{w}||^2 + \sum_{i=1}^{N} \max(1 - y^i h_{\mathbf{w},b}(\mathbf{x}), 0) \end{aligned}$$

where $\lambda||\mathbf{w}||^2$ is the regularizer and $\max(1 - y^i h_{\mathbf{w},b}(\mathbf{x}), 0)$ is the additive loss. In that way, the Hinge loss($Yf(x)$) would be 0 when $Yf(x) \geq 0$. Compared to logistic regression and linear regression, it is a good feature.

Kernel

ϕ is invariant to nuisance factors, sensitive to semantic variations (Encoder).
General idea: the original feature space can always be mapped to some higher dimensional feature space where training set is trainable.
$\phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \to \begin{pmatrix} r \\ \theta \end{pmatrix}$ would applied to the linearly separable in polar coordinates.
Kernel: $K(\mathbf{x}, \mathbf{y}) =< \phi(\mathbf{x}), \phi(\mathbf{y}) >$

Polynomial Kernel:$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x^T y} + 1)^p$, which $\phi(\mathbf{x}) = \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ 1 \end{bmatrix}$

Condition for kernel trick

Mercer Kernel:
1. Symmetric $k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_j, \mathbf{x}_i)$
2. Positive definite, $\alpha^T K\alpha \geq 0$ for all $\alpha \in \mathbb{R}^N$, where K is the $N \times N$Gram matrix with entries $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ Gram matrix is a positive semidefinite matrix
Then $k(,)$ is a valid kernel.

SVM before kernel

Optimization:

$$\min_{\alpha} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha^i \alpha^j y^i y^j < \mathbf{x}^i, \mathbf{x}^j >$$

$$s.t. : y^i \left(\sum_{j=1}^{N} \alpha^j y^j < \mathbf{x}^j, \mathbf{x}^i > +b \right) \geq 1, \forall i, \alpha \in \mathbb{R}^N \to O(N^3)$$

The classifier forms:

$$f(\mathbf{x}) =< \mathbf{w}, \mathbf{x} > +b = \sum_{i=1}^{N} \alpha^i y^i < \mathbf{x}^i, \mathbf{x} > +b$$

Linear kernel: $\mathbf{K}(\mathbf{x}, \mathbf{y}) = \mathbf{x}^{\mathbf{T}}\mathbf{y}$
Polynomial kernel: $\mathbf{K}(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^{\mathbf{T}}\mathbf{y} + \mathbf{1})^{\mathbf{p}}$
Radial Basis Function (a.k.a Gaussian) Kernel: $\mathbf{K}(\mathbf{x}, \mathbf{y}) = \exp(-\frac{1}{2\sigma^2}||\mathbf{x} - \mathbf{y}||^2)$

Radial Basis Function kernel expansion

$$\begin{aligned}
e^{-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2} &= e^{-\gamma(\mathbf{x}_i - \mathbf{x}_j)^2} = e^{-\gamma\mathbf{x}_i^2 + 2\gamma\mathbf{x}_i\mathbf{x}_j - \gamma\mathbf{x}_j^2} \\
&= e^{-\gamma\mathbf{x}_i^2 - \gamma\mathbf{x}_j^2}(1 + \frac{2\gamma\mathbf{x}_i\mathbf{x}_j}{1!} + \frac{(2\gamma\mathbf{x}_i\mathbf{x}_j)^2}{2!}) + \frac{(2\gamma\mathbf{x}_i\mathbf{x}_j)^3}{3!} + \ldots) \\
&= \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j) \\
where \quad \phi(\mathbf{x}) &= e^{-\gamma\mathbf{x}^2}[1, \sqrt{\frac{2\gamma}{1!}}\mathbf{x}, \sqrt{\frac{(2\gamma)^2}{2!}}\mathbf{x}^2, \sqrt{\frac{(2\gamma)^3}{3!}}\mathbf{x}^3, \ldots]^T
\end{aligned}$$

SVM after kernel

Optimization:

$$\min_{\alpha} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha^i \alpha^j y^i y^j K(\mathbf{x}^i, \mathbf{x}^j)$$

$$s.t. : y^i \left(\sum_{j=1}^{N} \alpha^j y^j K(\mathbf{x}^i, \mathbf{x}^j) + b \right) \geq 1, \forall i, \alpha \in \mathbb{R}^N \to O(N^3)$$

The classifier forms:

$$\begin{aligned} f(\mathbf{x}) &= \sum_{i=1}^{N} \alpha^i y^i K(\mathbf{x}^i, \mathbf{x}^j) + b \\ &= \sum_{\{i:\alpha^i \neq 0\}} w^i K(\mathbf{x}^i, \mathbf{x}^j) + b, w^i = y^i \alpha^i \end{aligned}$$

Compare with general $f(\mathbf{x}) = \sum_k w_k \phi_k(\mathbf{x})$

SVM learning method SMO

HoG/SIFT to image

Tree(not in lecture)

Decision tree

Entropy measures the uncertain of the random variable X
Here the distribution of X is $P(X = x_i) = p_i, i = 1, 2, \dots, n$
Entropy of X: $H(X) = -\sum_{i=1}^{n} p_i \log p_i$
Conditional Entropy: $H(Y|X) = \sum_{i=1}^{n} p_i H(Y|X = x_i)$
Information Gain is the difference between empirical entropy and empirical conditional entropy. $g(D, A) = H(D) - H(D|A)$
Information gain ratio:$g_R(D, A) = \frac{g(D,A)}{H_A(D)}$

ID3

using information gain

C4.5

using information gain ratio

Regression Tree

$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m)$

Classification Tree

K classes, and the probability is p_k

$$\begin{aligned} Gini(p) &= \sum_{k=1}^{K} p_k(1-p_k) = 1 - \sum_{k=1}^{K} p_k^2 \\ Gini(D,A) &= \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2 \end{aligned}$$

Ensemble learning

Generate a group of base-learners which has higher accuracy when **combined**.
Consider the error, $\mathbb{E}_{COM} = \frac{1}{M}\mathbb{E}_{AV}$

Bagging

pick subset of training data, then obtain weak learner.
Output final classifier by majority voting of the weak learner.

Boosting

Pick subset of training data using a sampling distribution, obtain weak learner(use weak learner to update sampling distribution).

Adaboost

Defines a classifier using an additive model(weighted voting):

$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \dots$$

Given: $(x^i, y^i), x^i \in X, y^i \in -1, 1, i = 1, \dots, N$
Initialize: $D_1(i) = \frac{1}{N}$ distribution on the sample
For $t = 1 \dots T$:
-Find classifier $h_t : X \rightarrow -1, 1 with smallest weighted error$

$$\epsilon = \frac{\sum_{i=1}^{N} D_t^i [y^i \neq_t (x^i)]}{\sum_i D_t^i}$$

-Update distribution:

$$
\begin{aligned}
D_{t+1}^{i} &= \frac{D_t^i}{Z_t} \times \{ \begin{matrix} \exp(-\alpha_t), \text{if} & y^i = h_t(x^i) \\ \exp(\alpha_t), \text{if} & y^i \neq h_t(x^i) \end{matrix} \\
&= \frac{D_t^i}{Z_t} \exp(-\alpha_t y^i h_t(x^i)) \\
a_t &= \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon} \\
Z_t &= \sum_i D_t^i \exp(-\alpha_t y^i h_t(x^i)) \\
\text{Final classifier :}\mathbf{H}(x) &= \text{sign}(\sum_t \alpha_t h_t(x))
\end{aligned}
$$

Posterior

$$
\begin{aligned}
\frac{\partial}{\partial f(\mathbf{x})} \mathbb{E}[e^{-\tilde{y} f(\mathbf{x})} | \mathbf{x}] &= \frac{\partial}{\partial f(\mathbf{x})} [p(\tilde{y}=1|\mathbf{x}) e^{-f(\mathbf{x})} + p(\tilde{y}=-1|\mathbf{x}) e^{f(\mathbf{x})}] \\
&= -p(\tilde{y}=1|\mathbf{x}) e^{-f(\mathbf{x})} + p(\tilde{y}=-1|\mathbf{x}) e^{f(\mathbf{x})} \\
&= 0 \Rightarrow \frac{p(\tilde{y}=1|\mathbf{x})}{p(\tilde{y}=-1|\mathbf{x})} = e^{2f(\mathbf{x})} \\
f(x) &= \frac{1}{2} \log(\frac{P(y=1|x)}{P(y=-1|x)})
\end{aligned}
$$