Machine Learning in Practice: a Crash Course

Lecture 6: Linear Classification

胡津铭
DolphinDB

Recap

- Our goal (supervised learning):
  - To learn a set of discriminant functions

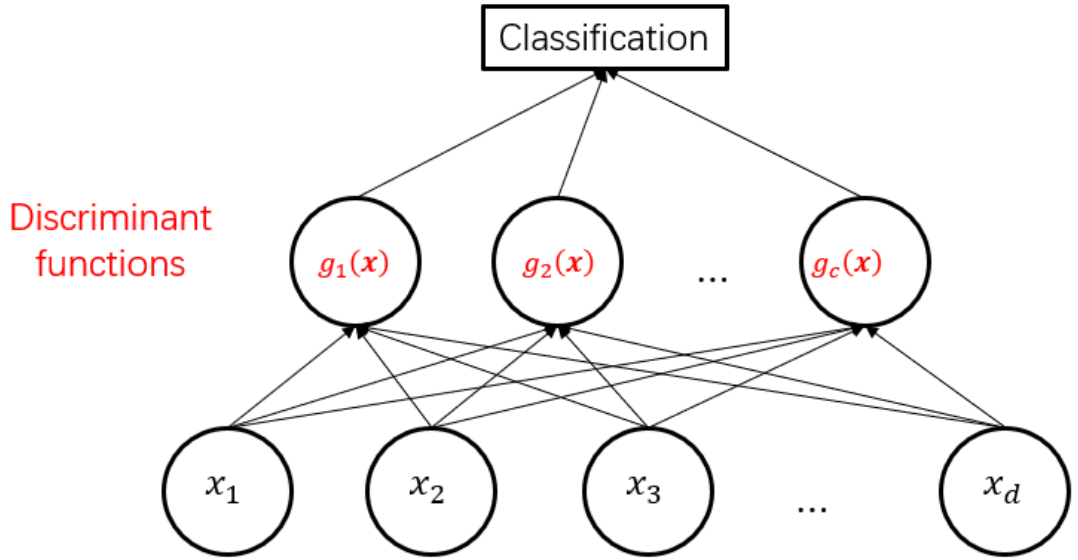
- Bayesian framework
  - We could design an optimal classifier if we knew:
    - $P(y_i)$: priors and $P(x\,|y_i)$: likelihodd
    - Using training data to estimate $P(y_i)$and $P(x\,|y_i)$
    - $P(y_i|x)$ is computed and be used as the discriminant functions

- Other possible ways?
  - Directly learning discriminant functions from the training data
  - Linear Methods for Regression

Today

- Logistic Regression
- Support Vector Machine
- Loss Function

Discriminant Functions and Classifiers



- Set of discriminant functions: $g_i(\boldsymbol{x}),\ i = 1, \cdots, c$

$$g(\boldsymbol{x}) = (XX^T + \lambda \boldsymbol{I})^{-1} X\boldsymbol{y}$$

- Classifier assigns a feature vector $\boldsymbol{x}$ to class y_i if:

$$g_i(\boldsymbol{x}) > g_j(\boldsymbol{x}), \qquad \forall j \neq i$$

Linear Regression of an Indicator Matrix

$$g(\boldsymbol{x}) = (XX^T + \lambda \boldsymbol{I})^{-1} X \boldsymbol{y}$$

$$\mathbf{y} = \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 2 \\ \vdots \\ 2 \\ \vdots \\ c \\ \vdots \\ c \end{bmatrix}$$

What is the problem with this approach?

$$Y = \begin{bmatrix} 1 & 0 & & 0 \\ \vdots & \vdots & & \vdots \\ 1 & 0 & & 0 \\ 0 & 1 & & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & & 1 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & & 1 \end{bmatrix}$$

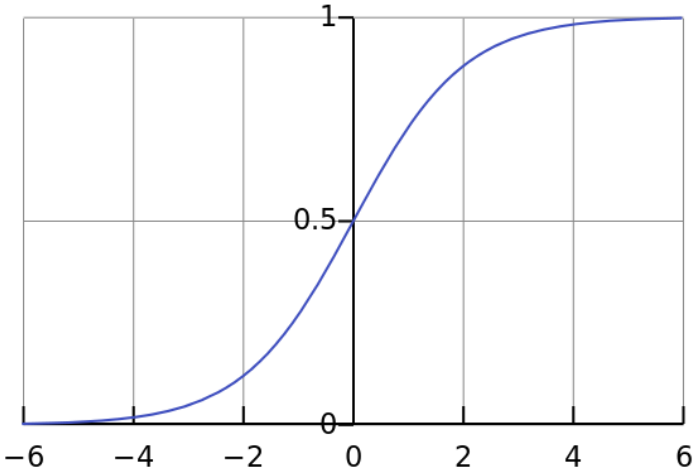
- One VS. Rest

Sigmoid function (logistic function)

- Linear Regression for classification can give us a result
  - However, it cannot give us a probability. When do we want a probability?
  - What is the assumption by linear regression in terms of loss function/metric?
- $\sigma(t) = \frac{e^t}{1+e^t} = \frac{1}{1+e^{-t}}$
  - It is the cumulative distribution function (CDF) of the standard logistic distribution.
  - While the input can have any value from $-\infty$ to $+\infty$, the output takes only values between 0 and 1, hence is interpretable as probability

$$\sigma: \ \mathrm{R} \rightarrow (0,1)$$

S-shaped



Logistic Regression

- **Logistic Regression (LR)** is a classification model used to describe the relationship between a **categorical** dependent variable and one or several independent variables by estimating **probabilities** using **sigmoid function**.
- $P(y_i = 1|\boldsymbol{x}_i, \boldsymbol{a}) = \sigma(\boldsymbol{a}^T\boldsymbol{x}_i) = \frac{1}{1+e^{-\boldsymbol{a}^T\boldsymbol{x}_i}}$
- $P(y_i = -1|\boldsymbol{x}_i, \boldsymbol{a}) = 1 - \sigma(\boldsymbol{a}^T\boldsymbol{x}_i) = 1 - \frac{1}{1+e^{-\boldsymbol{a}^T\boldsymbol{x}_i}} = \frac{1}{1+e^{\boldsymbol{a}^T\boldsymbol{x}_i}}$
- $P(y_i = \pm 1|\boldsymbol{x}_i, \boldsymbol{a}) = \sigma(y_i\boldsymbol{a}^T\boldsymbol{x}_i) = \frac{1}{1+e^{-y_i\boldsymbol{a}^T\boldsymbol{x}_i}}$

Logistic Regression: Representation

- $P(y_i = \pm 1 | \boldsymbol{x}_i, \boldsymbol{a}) = \sigma(y_i \boldsymbol{a}^T \boldsymbol{x}_i) = \frac{1}{1+e^{-y_i \boldsymbol{a}^T x_i}}$
- Is this a linear model or non-linear model? Hint(Consider the decision boundary)

Logistic Regression: Evaluation

- What metrics to use?
  - Accuracy, precision, recall, ……
- What is the **loss function**?
- Maximum likelihood estimation
- $P(y_i = \pm 1 | \boldsymbol{x}_i, \boldsymbol{a}) = \sigma(y_i \boldsymbol{a}^T \boldsymbol{x}_i) = \frac{1}{1+e^{-y_i \boldsymbol{a}^T \boldsymbol{x}_i}}$
- $P(D) = \prod_{i \in I} \sigma(y_i \boldsymbol{a}^T \boldsymbol{x}_i)$
- $l\big(P(D)\big) = \sum_{i \in I} \log\big(\sigma(y_i \boldsymbol{a}^T \boldsymbol{x}_i)\big) = -\sum_{i \in I} \log\left(1 + e^{-y_i \boldsymbol{a}^T \boldsymbol{x}_i}\right)$
- Why log here?
- Is this a convex function?

Logistic Regression: Optimization

- $l\big(P(D)\big) = \sum_{i \in I} \log\big(\sigma(y_i \boldsymbol{a}^T \boldsymbol{x}_i)\big) = -\sum_{i \in I} \log\left(1 + e^{-y_i \boldsymbol{a}^T \boldsymbol{x}_i}\right)$
- $l(a) = -\sum_{i \in I} \log\left(1 + e^{-y_i \boldsymbol{a}^T \boldsymbol{x}_i}\right)$
- How to optimize it?
- Gradient Descent!

Assumptions behind logistic regression

- $l(a) = -\sum_{i \in I} \log\left(1 + e^{-y_i \boldsymbol{a}^T \boldsymbol{x}_i}\right)$
- It assumes the event($y_i = \pm 1$) happens with a probability p, and conforms to a Binomial Distribution (二项分布)
  - Is this a reasonable assumption? Compared with Linear Regression?
- It assumes the data can be linearly separated.
  - If it cannot be linearly separated, then approximate it with a linear separation.
- Another viewpoint: from Naïve Bayes to Logistic
  - We want to get $\log\left(\frac{P(\boldsymbol{y_i}|\, x)}{P(\boldsymbol{y_j}|\, x)}\right)$ for classification
  - In Naïve Bayes, we assume features are conditionally independent, so $P(x_1, \cdots x_p|y) = P(x_1|y) \cdots P(x_p|y)$
  - In logistics, we assume $\log\left(\frac{P(\boldsymbol{y_i}|\, x)}{P(\boldsymbol{y_j}|\, x)}\right) = a^T x$

Regularized Logistic Regression

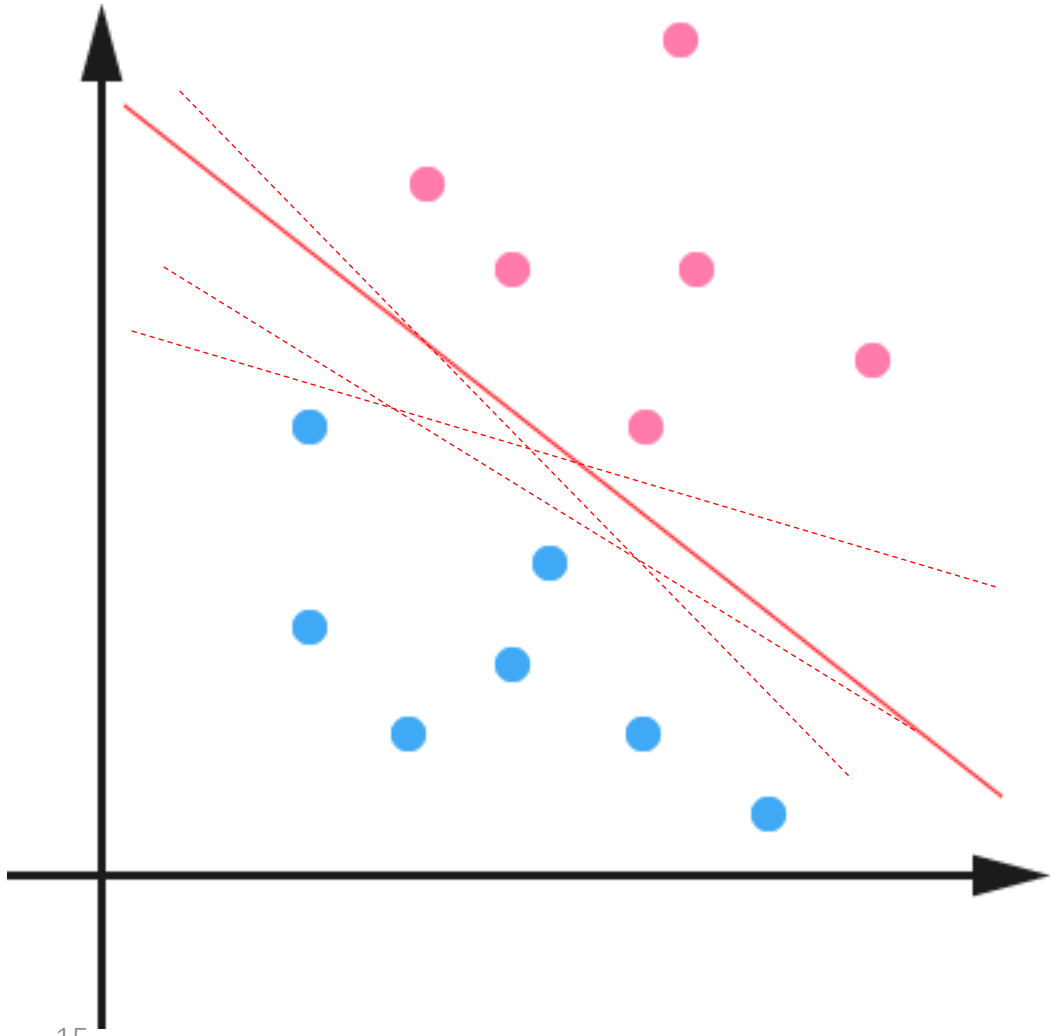
- $E(\boldsymbol{a}) = \sum_{i \in I} \log\left(1 + e^{-y_i \boldsymbol{a}^T \boldsymbol{x}_i}\right) + \lambda \sum_{j=1}^{p} |a_j|$
- L2-regularizer
- L1-regularizer (Sparse Logistic Regression)

Pros and Cons

- Pros:
  - Binomial distribution is a good assumption for classification
    - Consequently, logistic function and its extension to multi-class classification **softmax** is one of the first-choice for classification.
  - Provide a probability
    - Probability is necessary for some applications, such as ads
  - Low computation, easy to optimize
    - Works pretty well in large-scale dataset, tens of millions of features and hundreads of millions of data
  - Support online learning
- Cons:
  - Linear model is still too simple
    - High bias, low variance

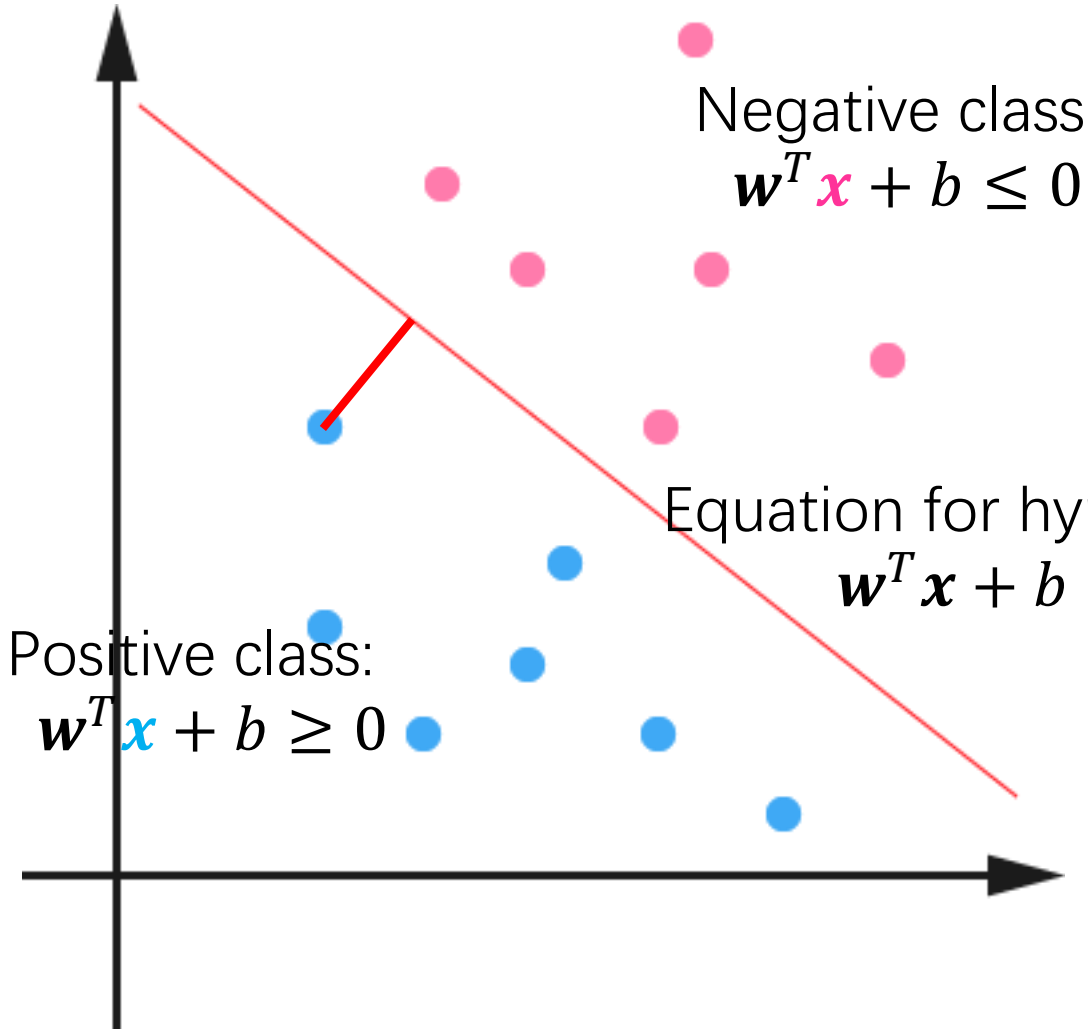
Support Vector Machine

Non-uniqueness of hyperplane classifier



Which one is better?

Binary Classification



Negative class:
$\boldsymbol{w}^T \boldsymbol{x} + b \leq 0$

Equation for hyperplane:
$\boldsymbol{w}^T \boldsymbol{x} + b = 0$

Positive class:
$\boldsymbol{w}^T \boldsymbol{x} + b \geq 0$

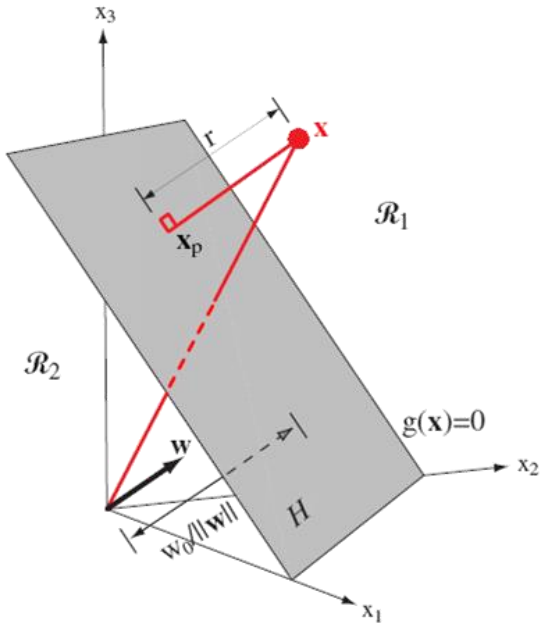
Geometrical Margin

- Define γ as the distance from $\boldsymbol{x}$ to the hyperplane
  - Computation: let the projection of $\boldsymbol{x}$ into the hyperplane be $\boldsymbol{x_0}$, then we have

$$\boldsymbol{x} = \boldsymbol{x}_0 + y\gamma \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$$

$$\boldsymbol{w}^T \left(\boldsymbol{x} - y\gamma \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \right) + b = 0$$

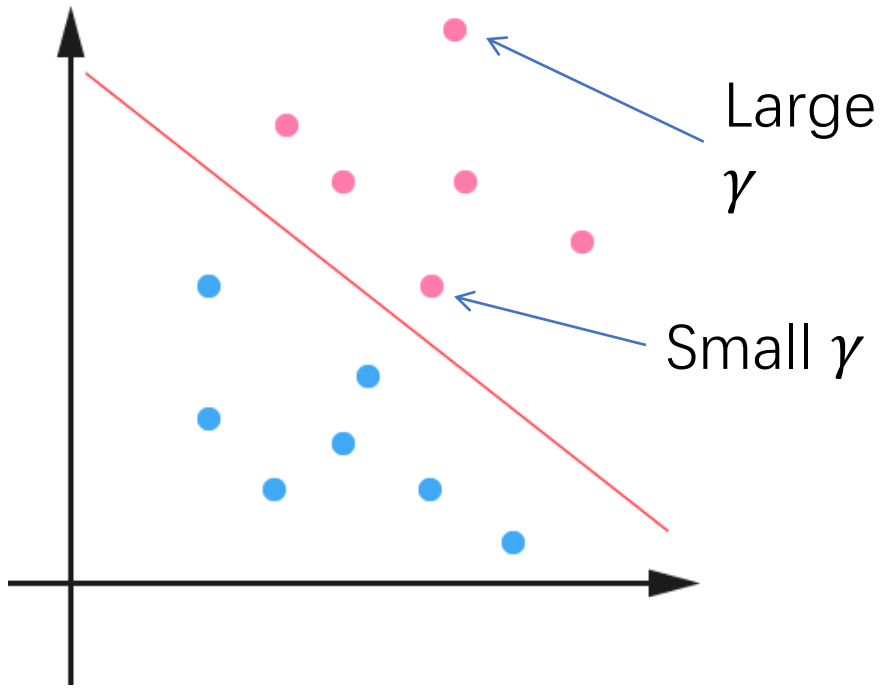
$$\gamma = y \frac{\boldsymbol{w}^T \boldsymbol{x} + b}{\|\boldsymbol{w}\|}$$



- γ: **geometrical margin**

Geometrical Margin

$$\gamma = y\frac{\boldsymbol{w}^T\boldsymbol{x} + b}{\|\boldsymbol{w}\|}$$



If the hyperplane moves a little, points with small γ will be affected, but points with large γ won't

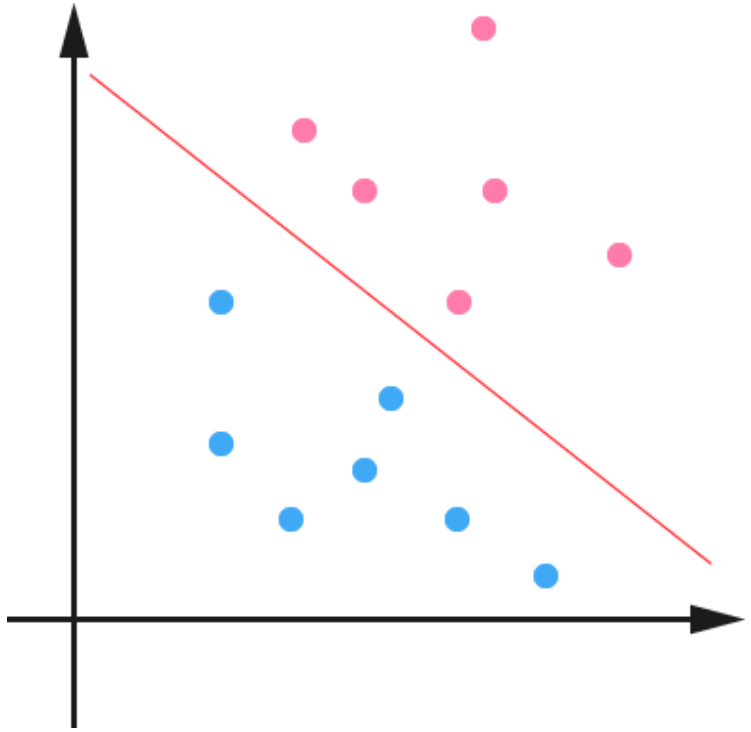
Maximum Margin Classifier

- Define the margin of a dataset be the minimum margin of each data point
- Maximum margin classifier tries to achieve the maximum possible margin for a given dataset
  - Thus maximize the *confidence* of classifying the dataset

- Goal: Find the hyperplane with the largest margin

Why Maximum Margin?

- Intuitively this feels safest
- If we've made a small error in the locationof the boundary, this gives us least chance of causing misclassification
- There're some theories (using VC dimension) that is related to the proposition that this is a good thing.
- If there are some noises in the data, the result may move a little bit, but still falls in the right range
- Empirically it works very well.



Maximum Margin Classifier: Representation

- $\max_{\boldsymbol{w},b} \gamma = \max_{\boldsymbol{w},b} \frac{y(\boldsymbol{w}^T \boldsymbol{x}+b)}{\|\boldsymbol{w}\|}$
- $s.t., \gamma_i \geq \gamma$
- $y(\boldsymbol{w}^T \boldsymbol{x} + b)$ can be made arbitrarily large without changing the hyperplane, so we simply fix it at $y(\boldsymbol{w}^T \boldsymbol{x} + b) = 1$

Maximum Margin Classifier: Representation

- $\max_{\boldsymbol{w},b} \frac{y(\boldsymbol{w}^T\boldsymbol{x}+b)}{\|\boldsymbol{w}\|}$ → $\max_{\boldsymbol{w},b} \frac{1}{\|\boldsymbol{w}\|}$ → $\min_{\boldsymbol{w},b} \|\boldsymbol{w}\|$

$$s.t., \gamma_i = \frac{y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)}{\|\boldsymbol{w}\|} \geq \gamma$$

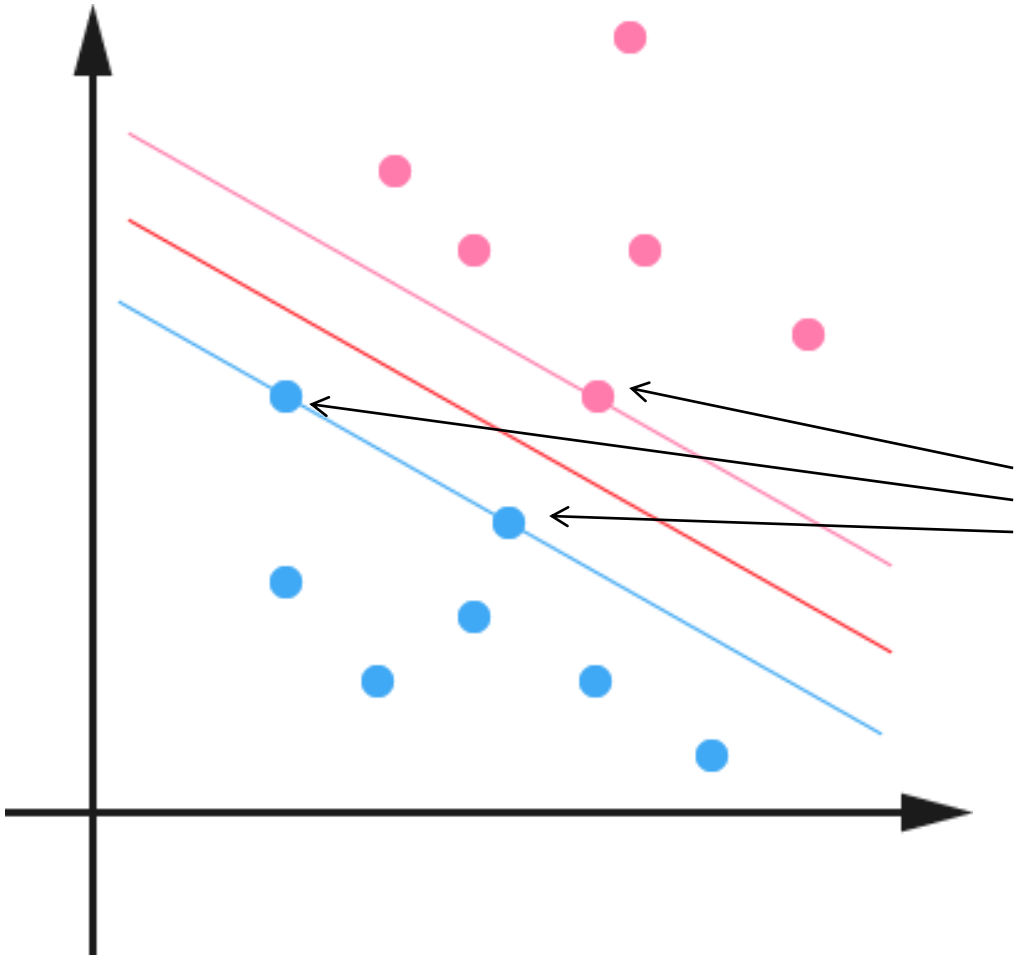
↓

$$y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq \gamma\|\boldsymbol{w}\| = 1$$

Maximum Margin Classifier

- $\min_{\boldsymbol{w},b} \frac{1}{2} \|\boldsymbol{w}\|^2$
- $y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) \geq 1$
- Square and a coefficient $\frac{1}{2}$ are added for the convenience of the derivation of optimization, and the minimizer of $\|\boldsymbol{w}\|$ and $\frac{1}{2} \|\boldsymbol{w}\|^2$ is obviously the same.
- How to optimize?
  - Convex optimization with constraint

Support Vector Machine



Hyper plane of maximum margin is *support*ed by those points (vectors) on the margin. Those are called Support Vectors.

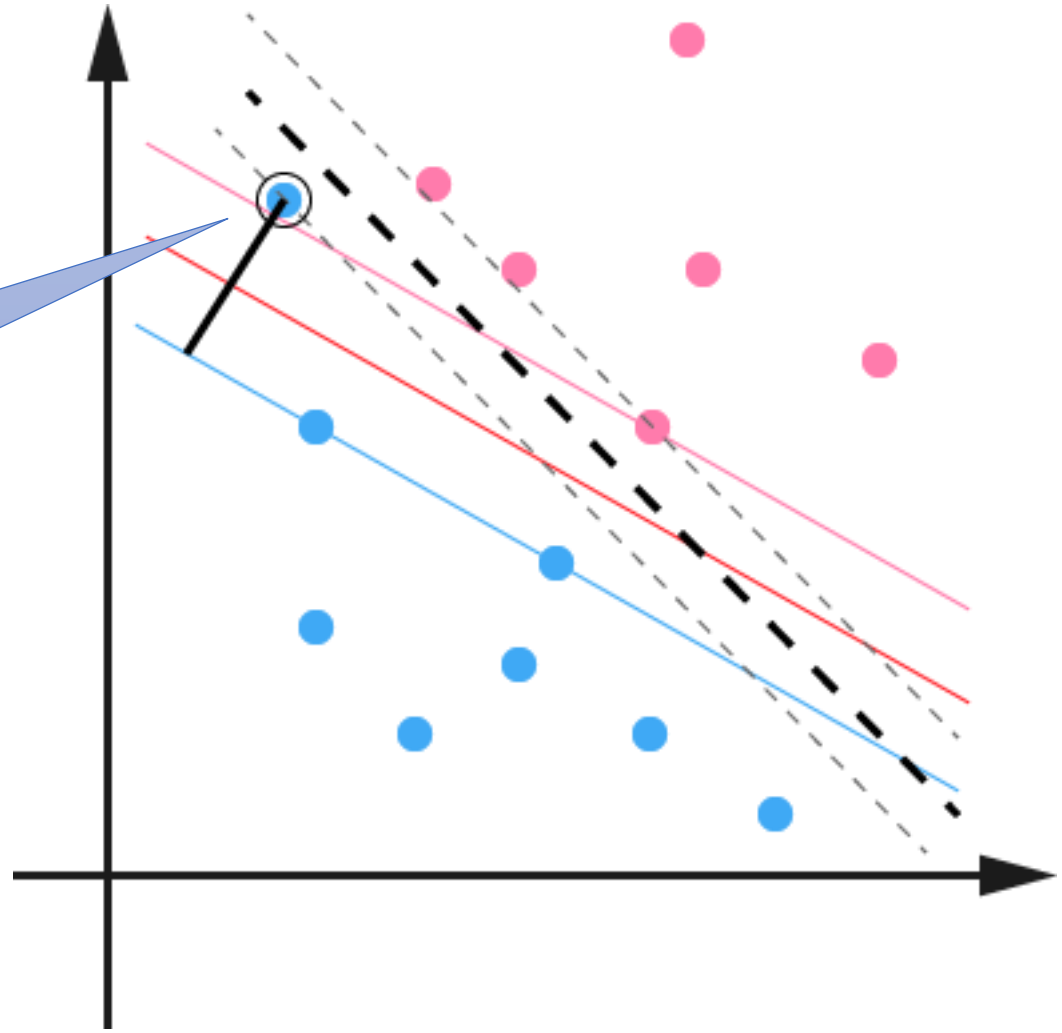
Non-support vectors can move freely without affecting the position of the hyperplane as long as they don't exceed the margin.

Weakness of the Original Model

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2$$
$$y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1$$

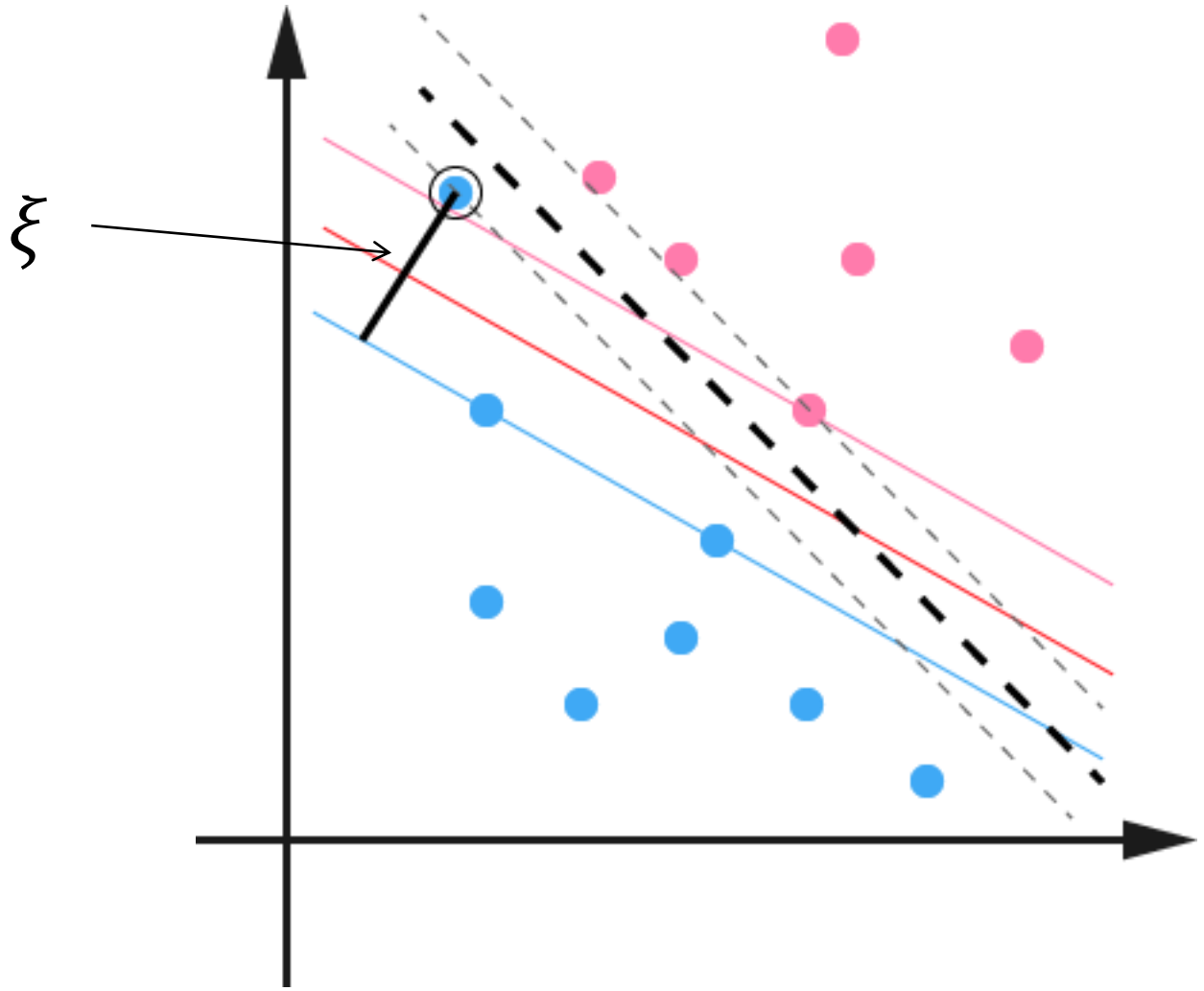
When an outlier appear, the optimal hyperplane may be pushed far away from its original/correct place. The resultant margin will also be smaller than before.

- Red Solid: the original hyperplane
- Dark dashed: the new hyperplane



Slack Variables

Assign a slack variable ξ to each data point. That means we allow the point to deviate from the correct margin by a distance of ξ (Actually $\|\boldsymbol{w}\|\xi$ when considering geometrical margin).



New Objective Function

Slack variables can't be arbitrarily large, we want to minimize the sum of all slack variables

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$y(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

New Objective Function

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$y(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

- We would pay a cost of the objective function being increased by $C\ \xi_i$. The parameter C controls the relative weighting between the twin goals of making the $\|\boldsymbol{w}\|^2$small (makes the margin large) and of ensuring that most examples have functional margin at least 1.

Assumptions behind SVM

- The max margin classifier is a good classifier
  - If we find the max margin classifier in the training data, the generalization of this model should be good
  - Maybe it is also a max margin classifier in the test data, or at least it can avoid some noises in the training/testing data
- Is this a linear model? (Consider the decision boundary)

Pros and Cons of SVM

- Pros
  - Max margin works well for classification
  - A so-called kernel trick fits well with SVM (we may mention it in the following lectures)
- Cons
  - Linear boundary
  - When the feature space is very large, say millions of features, then the *measure*（测度）is degraded. As a result, most points may be approximately the same near to the boundary, which makes the max margin sort of meaningless
  - No probability (though can add a sigmoid-like function to get a probability)

Unconstrained Optimization of SVM

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$
$$y(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i$$
$$\xi_i \geq 0$$

$$\xi_i \geq 1 - y(\boldsymbol{w}^T\boldsymbol{x}_i + b) \qquad \xi_i = \max[1 - y(\boldsymbol{w}^T\boldsymbol{x}_i + b), 0]$$

$$\min_{\boldsymbol{w},b}\left\{\sum_{i=1}^{n}\underbrace{\max[1 - y(\boldsymbol{w}^T\boldsymbol{x}_i + b), 0]}_{\text{Loss function}} + \underbrace{\frac{1}{2C}\|\boldsymbol{w}\|^2}_{\text{Regularizer}}\right\}$$

$\ell(f) = \max[1 - yf, 0]$ Hinge loss

Linear regression: $E(\boldsymbol{a}) = \sum_{i\in I}\underbrace{(y_i - \boldsymbol{a}^T\boldsymbol{x}_i)^2}_{\text{Loss function}}$

$\ell(f) = (y - f)^2 = (1 - yf)^2$ Square loss

Logistic regression: $E(\boldsymbol{a}) = \sum_{i\in I}\underbrace{\log\left(1 + e^{-y_i\boldsymbol{a}^T\boldsymbol{x}_i}\right)}_{\text{Loss function}}$

$\ell(f) = \log\left(1 + e^{-yf}\right)$ Logistic loss

A General formulation of classifiers

$$\min_f \left\{ \sum_{i=1}^{n} \ell(f) + \lambda R(f) \right\}$$

Loss function

Regularizer

Square loss: $\ell(f) = (1 - yf)^2$	Ordinary regression
Logistic loss: $\ell(f) = \log\left(1 + e^{-yf}\right)$	Logistic regression
Hinge loss: $\ell(f) = \max[1 - yf, 0]$	SVM

L2-regularizer

L1-regularizer

Loss Function