Linear Classification and Support Vector Machine and Stochastic Gradient Descent

Prof.Mingkui Tan

South China University of Technology
Southern Artificial Intelligence Laboratory(SAIL)

September 19, 2017

Content

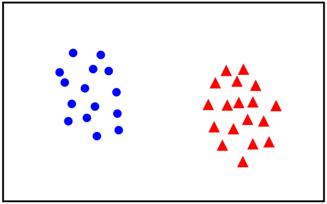
Contents

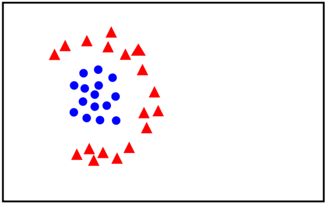
Binary Classification

Given training data $(\mathbf{x}_i, y_i)$ for $i = 1 \ldots N$, with $\mathbf{x}_i \in R^n$ and $y_i \in -1, 1$,learn a classfier $f(\mathbf{x})$ such that

$$f(\mathbf{x}_i) \begin{cases} \geq 0 & y_i = +1 \\ < 0 & y_i = -1 \end{cases}$$

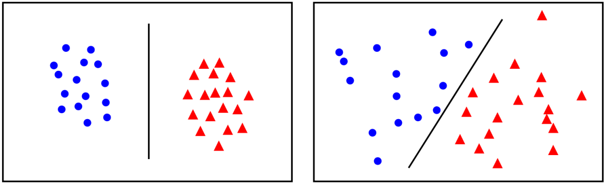
i.e. $y_i f(\mathbf{x}_i) > 0$ for a correct classification



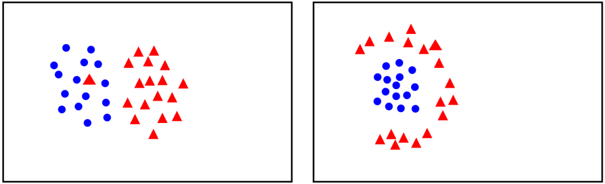


Linear Separability

linearly separable



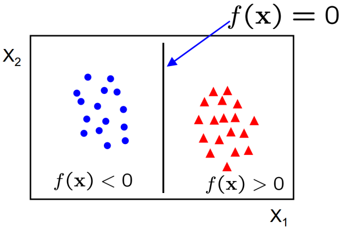
not linearly separable



Linear Classifiers

A linear classifier has the form:

$$f(x) = \mathbf{w}^\top \mathbf{x} + b$$

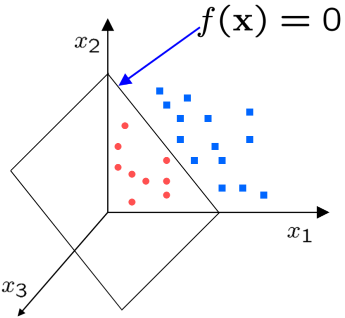


- In 2D the discriminant is a line
- $\mathbf{w}$ is the normal to the line, and b the bias
- $\mathbf{w}$ is known as the weight vector

Linear Classifiers

A linear classifier has the form:

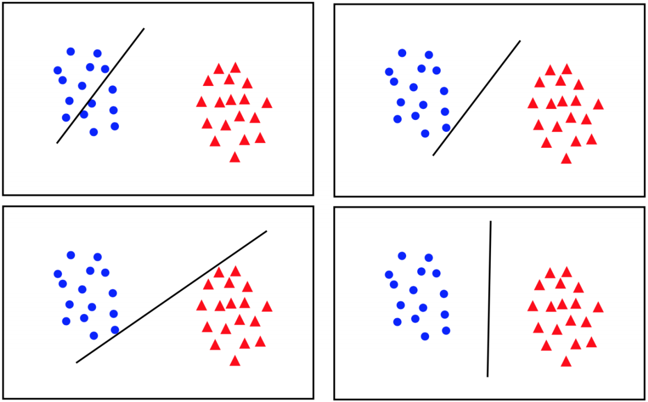
$$f(x) = \mathbf{w}^\top \mathbf{x} + b$$



- In $3D$ the discriminant is a plane, and in nD it is a hyperplane

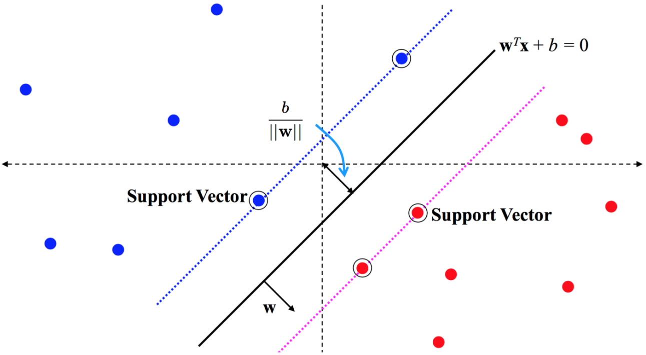
Contents

What's a Good Decision Boundary?



- Maximum margin solution: most stable under perturbations of the inputs

Max-margin Methods



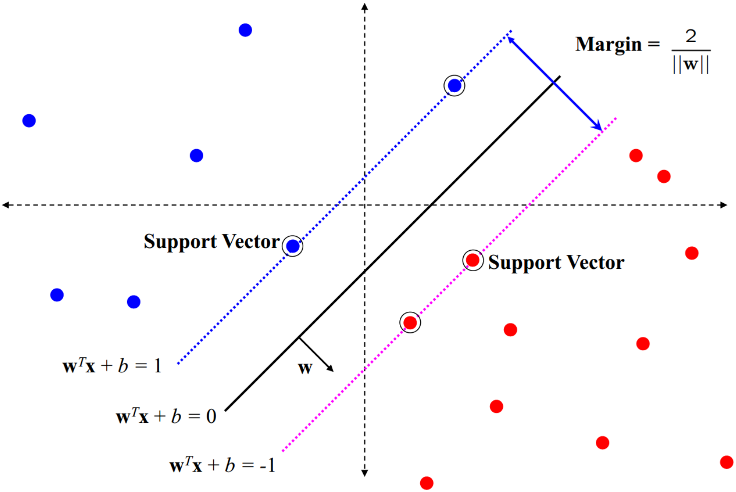
- "margin" == min dist. to decision boundary. Maximize it!

SVM-sketch Derivation

- Choose normalization such that $\mathbf{w}^\top \mathbf{x}_+ + b = +1$ and $\mathbf{w}^\top \mathbf{x}_- + b = -1$ for the positive and negative support vectors respectively
- Then the magin is given by

$$\frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot (\mathbf{x}_+ - \mathbf{x}_-) = \frac{\mathbf{w}^\top (\mathbf{x}_+ - \mathbf{x}_-)}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|} \tag{1}$$

Support Vector Machine



Basic Support Vector Machine

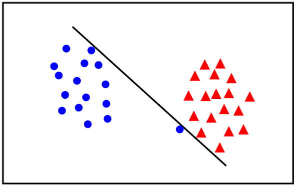
- Learning the SVM can be formulated as an optimization:

$$\max_{\mathbf{w},b} \frac{2}{\|\mathbf{w}\|}$$
$$s.t. \quad \mathbf{w}^\top \mathbf{x}_i + b \begin{cases} \geqslant 1 & y_i = +1 \\ \leqslant -1 & y_i = -1 \end{cases}$$

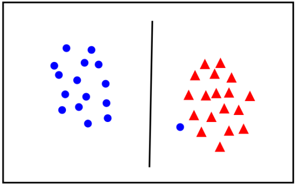
- Or equivalently:

$$\min_{\mathbf{w},b} \frac{\|\mathbf{w}\|^2}{2}$$
$$s.t. \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geqslant 1, \qquad i = 1, 2, \ldots, n.$$

Linear Separability Again: What is The Best w?



• the points can be linearly separated but there is a very narrow margin

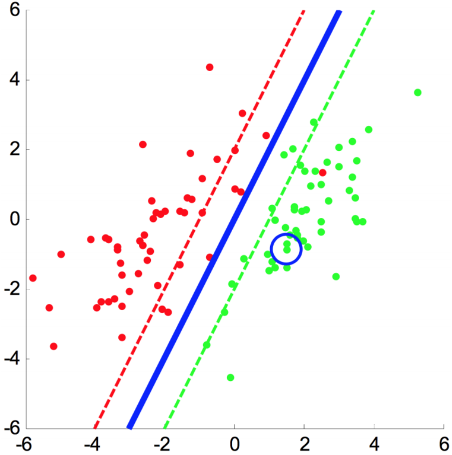


• but possibly the large margin solution is better, even though one constraint is violated

In general there is a trade off between the margin and the number of mistakes on the training data

Linear Separability Again: What is The Best **w**?

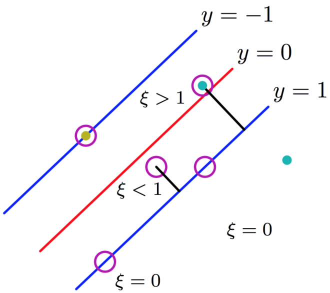
Moreover, training data may not be linearly separable!



A Relaxed Formulation

Introduce variable $\xi_i \geqslant 0$, for each i, which represents how much example i is on wrong side of margin boundary

- If $\xi_i = 0$ then it is ok
- If $0 < \xi_i < 1$ it is correctly classified, but with a smaller margin than $\frac{1}{\|\mathbf{w}\|}$
- If $\xi_i > 1$ then it is incorrectly classified



Soft Margin Formulation

The optimization problem becomes:

$$\min_{\mathbf{w},b} \quad \frac{\|\mathbf{w}\|^2}{2} + C \sum_{i=1}^{N} \xi_i$$
$$s.t. \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geqslant 1 - \boldsymbol{\xi}_i, \qquad i = 1, 2, \ldots, n.$$

Hinge Loss

Hinge loss:

$$Hinge\ loss = \xi_i = \max(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)) \tag{2}$$

The optimization problem becomes:

$$\min_{\mathbf{w},b} \quad \frac{\|\mathbf{w}\|^2}{2} + C \sum_{i=1}^{N} \max(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)) \tag{3}$$

Dual Problem

An optimization problem can be considered in two ways, primal problem and dual problem

- for primal problem of basic SVM:

$$\min_{\mathbf{w},b} \frac{\|\mathbf{w}\|^2}{2}$$
$$s.t. \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geqslant 1, \qquad i = 1, 2, \ldots, n.$$

- its Lagrange function is:

$$\mathcal{L}(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} a_i(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)) \tag{4}$$

Dual Problem

- its Lagrange "dual function" is:

$$\mathbf{D}(\mathbf{a}) = \inf \boldsymbol{\mathcal{L}}(\mathbf{w}, b, \mathbf{a})$$

  Dual function gives the lower bound of the optimal value of primal problem

- dual problem: the best lower bound dual function can get

$$\max_{\mathbf{a}} \mathbf{D}(\mathbf{a}) \tag{5}$$

Dual Problem

- setting partial derivative of $\mathbf{D}$ with respect to $\mathbf{w}$ to 0:

$$\begin{aligned}\nabla_w \mathbf{D} &= \mathbf{w} - \sum_{i=1}^{n} a_i y_i \mathbf{x}_i = 0 \\ \rightarrow \mathbf{w} &= \sum_{i=1}^{n} a_i y_i \mathbf{x}_i = 0\end{aligned} \tag{6}$$

- setting partial derivative of $\mathcal{L}$ with respect to b to 0:

$$\nabla_b \mathbf{D} = \sum_{i=1}^{n} a_i y_i = 0 \tag{7}$$

Dual Problem

- adding (6), (7) to (4):

$$
\begin{aligned}
\mathcal{L}(\mathbf{w}, b, \mathbf{a}) &= \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} a_i(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)) \\
&= \frac{1}{2}\mathbf{w}^\top\mathbf{w} + \sum_{i=1}^{n} a_i - \sum_{i=1}^{n} \mathbf{w}^\top a_i y_i \mathbf{x}_i - \sum_{i=1}^{n} a_i y_i \\
&= \frac{1}{2}\mathbf{w}^\top\mathbf{w} + \sum_{i=1}^{n} a_i - \mathbf{w}^\top\mathbf{w} - 0 \\
&= \sum_{i=1}^{n} a_i - \frac{1}{2}[\sum_{i=1}^{n} a_i y_i \mathbf{x}_i]^\top[\sum_{j=1}^{n} a_i y_i \mathbf{x}_i] \\
&= \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j
\end{aligned}
$$

Dual Problem

- finally, we can get the dual problem:

$$\max_{\mathbf{a}} \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j$$

$$\text{s.t.} \quad \sum_{i=1}^{n} a_i y_i = 0,$$

$$a_i \geqslant 0, \quad i = 1, 2, \ldots, n.$$

Contents

Gradient Descent

- True gradient descent is a batch algorithm, slow but sure

$$w^+ = w - \gamma \boxed{\frac{1}{n} \sum_i \nabla_w L(f_w(x_i), y_i)} \quad \nabla E_n(f_w)$$

learning rate or gain

Stochastic Optimization Motivation

- Information is redundant amongst samples
- Sufficient samples means we can afford more frequent, noisy updates
- Never-ending stream means we should not wait for all data
- Tracking non-stationary data means that the target is moving

Stochastic Optimization

Idea: estimate function and gradient from a small, current subsample of your data and with enough iterations and data, you will converge to the true minimum

- Better for large datasets and often faster convergence
- Hard to reach high accuracy
- Best classical methods can not handle stochastic approximation
- Theoretical definitions for convergence not as well defined

Stochastic Gradient Descent (SGD)

- Randomized gradient estimate to minimize the function using a single randomly picked example

  Instead of ∇f ,use $\tilde{\nabla} f$,where $\mathbf{E}[\tilde{\nabla} f] = \nabla f$
- The resulting update is of the form:

  $$\mathbf{w}^{+} = \mathbf{w} - \gamma \nabla_{\mathbf{w}} \boldsymbol{\mathcal{L}}(f_{\mathbf{w}}(x_i, y_i))$$
- Although random noise is introduced, it behaves like gradient descent in its expectation

Stochastic Gradient Descent

SGD works similar as GD, but more quickly by estimating gradient from a few examples at a time

Gradient Descent:

1.while True{
2.   loss= f(params,x_i, y_i);
3. d_w= gradient;
4.   params -=learning rate $\cdot d_w$;
5.   if (stopping condition is met){
6.   return params;
7.   }
8.}

GD

Stochastic Gradient Descent:

1.for (x_i, y_i) in training set{
2.   loss= f(params,x_i, y_i);
3. d_w= gradient;
4.   params -=learning rate $\cdot d_w$;
5.   if (stopping condition is met){
6.   return params;
7.   }
8.}

SGD

The Benefits of SGD

- Gradient is easy to calculate (instantaneous)
- Less prone to local minima
- Small memory footprint
- Get to a reasonable solution quickly
- Works for non-stationary environments as well as online settings
- Can be used for more complex models and error surfaces

Importance of Learning Rate

- Learning rate has a large impact on convergence

  Too small $\rightarrow$ too slow

  Too large $\rightarrow$ oscillatory and may even diverge

- Should learning rate be fixed or adaptive?
- Is convergence necessary?

  Non-stationary: convergence may not be required

  Stationary: learning rate should decrease with time

  Robbins-Monroe sequence is adequate $\gamma_t = \frac{1}{t}$

Minibatch Stochastic Gradient Descent

- Rather than using a single point, use a random subset where the size is less than the original data size

$$\mathbf{w}^{+} = \mathbf{w} - \gamma \frac{1}{|\boldsymbol{\mathcal{S}}_k|} \sum_{i \in \boldsymbol{\mathcal{S}}_k} \nabla_{\mathbf{w}} \boldsymbol{\mathcal{L}}(f_{\mathbf{w}}(\mathbf{x}_i, y_i)),$$
$$where \quad \boldsymbol{\mathcal{S}}_k \subseteq [n]$$

- Like the single random sample, the full gradient is approximated via an unbiased noisy estimate
- Random subset reduces the variance by a factor of $\frac{1}{|\boldsymbol{\mathcal{S}}_k|}$, but is also $|\boldsymbol{\mathcal{S}}_k|$ times more expensive

Minibatch Stochastic Gradient Descent

MSGD works identically to SGD, except that we use more than one training example to make each estimate of the gradient

Stochastic Gradient Descent:

1.for (x_i, y_i) in training set{
2.   loss= f(params,x_i, y_i);
3. d_w= gradient;
4.   params -=learning rate $\cdot d_w$;
5.   if (stopping condition is met){
6.   return params;
7.   }
8.}

SGD

Minibatch Stochastic Gradient Descent

1.for(x_{batch}, y_{batch}) in training set{
2.   loss= f(params,x_{batch}, y_{batch});
3. d_w= gradient;
4.   params -=learning rate $\cdot d_w$;
5.   if (stopping condition is met){
6.   return params;
7.   }
8.}

MSGD

Example

- Optimization problem:

$$\min_{\mathbf{w},b} \quad f : \frac{\|\mathbf{w}\|^2}{2} + C\sum_{i=1}^{N} \max(0, 1 - y_i(\mathbf{w}^\top x_i + b))$$

- Gradient computation:

$$\nabla f = \begin{bmatrix} \nabla_{\mathbf{w}} f(\mathbf{w}, b) \\ \nabla_b f(\mathbf{w}, b) \end{bmatrix}$$

- Update costs:
  - Batch: $O(nd)$
  - Stochastic: $O(d)$
  - Mini-batch: $O(|\boldsymbol{\mathcal{S}}_k|d)$

Gradient Computation

$$\mathbf{w} = \begin{bmatrix} w_1 & \dots & w_n \end{bmatrix}^\top$$
$$\|\mathbf{w}\|^2 = \|\mathbf{w}\|_2^2 = w_1^2 + w_2^2 + \cdots + w_n^2$$

- numerator layout:

$$\begin{aligned} \frac{\partial(\|\mathbf{w}\|^2)}{\partial \mathbf{w}} &= \begin{bmatrix} \frac{\partial(w_1^2 + w_2^2 + \cdots + w_n^2)}{\partial w_1} & \dots & \frac{\partial(w_1^2 + w_2^2 + \cdots + w_n^2)}{\partial w_n} \end{bmatrix} \\ &= \begin{bmatrix} 2w_1 & \dots & 2w_n \end{bmatrix} \\ &= 2\mathbf{w}^\top \end{aligned}$$

- so:

$$\frac{1}{2} \cdot \frac{\partial(\|\mathbf{w}\|^2)}{\partial \mathbf{w}} = \mathbf{w}^\top \tag{8}$$

Gradient Computation

- if $1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b) >= 0$:

$$
\begin{aligned}
&\frac{\partial(\sum_{i=1}^{N} \max(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)))}{\partial \mathbf{w}} \\
&= \frac{\partial(\sum_{i=1}^{N} -y_i(\mathbf{w}^\top \mathbf{x}_i + b))}{\partial \mathbf{w}} \\
&= -\frac{\partial(\sum_{i=1}^{N} y_i \mathbf{w}^\top \mathbf{x}_i)}{\partial \mathbf{w}} \\
&= -\sum_{i=1}^{N} y_i \begin{bmatrix} \frac{\partial(w_1 x_{i1} + \cdots + w_n x_{in})}{\partial w_1} & \ldots & \frac{\partial(w_1 x_{i1} + \cdots + w_n x_{in})}{\partial w_n} \end{bmatrix} \\
&= -\sum_{i=1}^{N} y_i \cdot \begin{bmatrix} x_{i1} & \ldots & x_{in} \end{bmatrix} \\
&= -\mathbf{y}^\top \cdot \mathbf{X}
\end{aligned}
$$

Gradient Computation

- if $1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b) < 0$:

$$\frac{\partial(\sum_{i=1}^{N} \max(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)))}{\partial \mathbf{w}} = 0$$

- so:

$$\frac{\partial(\sum_{i=1}^{N} max(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)))}{\partial \mathbf{w}} =$$

$$\begin{cases} -\mathbf{y}^\top \mathbf{X} & 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b) >= 0 \\ 0 & 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b) < 0 \end{cases}$$

Gradient Computation

- At last we have:

$$\frac{\partial f(\mathbf{w}, b)}{\mathbf{w}} = \begin{cases} \mathbf{w}^\top - C\mathbf{y}^\top \mathbf{X} & 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b) >= 0 \\ \mathbf{w}^\top & 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b) < 0 \end{cases} \tag{9}$$

- and:

$$\frac{\partial f(\mathbf{w}, b)}{b} = \begin{cases} -C \sum_{i=1}^{N} y_i & 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b) >= 0 \\ 0 & 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b) < 0 \end{cases} \tag{10}$$

Example

- n=10000,d=20

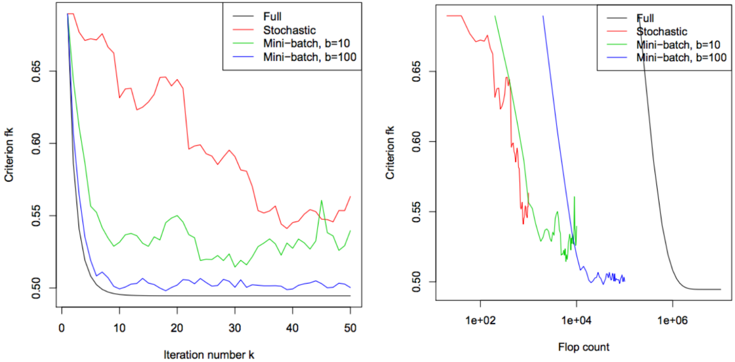


Figure: Iterations make better progress as mini-batch size is larger but also takes more computation time

SGD Recommendations

- Randomly shuffle training examples

  Although theory says you should randomly pick examples, it is easier to make a pass through your training set sequentially

  Shuffling before each iteration eliminates the effect of order

- Monitor both training cost and validation error

  Set aside samples for a decent validation set

  Compute the objective on the training set and validation set

  (expensive but better than overfitting or wasting computation)

SGD Recommendations

- Check gradient using finite differences
  - If computation is slightly incorrect can yield erratic and slow algorithm
  - Verify your code by slightly perturbing the parameter and inspecting differences between the two gradients
- Experiment with the learning rates using small sample of training set
  - SGD convergence rates are independent from sample size
  - Use traditional optimization algorithms as a reference point

SGD Recommendations

- Leverage sparsity of the training examples
  For very high-dimensional vectors with few non zero coefficients, you only need to update the weight coefficients corresponding to nonzero pattern in $\mathbf{x}$
- Use learning rates of the form$\gamma_t = \gamma_0(1 + \gamma_0 \lambda t)^{-1}$
  Allows you to start from reasonable learning rates determined by testing on a small sample
  Works well in most situations if the initial point is slightly smaller than best value observed in training sample