Machine Learning

Anders Jonsson & Vicenç Gómez

Master in Intelligence Interactive Systems
2021-22

Lecture 2
Linear Models

Content

Content

Intuition

- There exists a set of objects or concepts that we want to analyze
- Key assumption: objects with similar features behave similarly
- Supervised learning: data comes with labels

Supervised learning problem

A supervised learning problem consists of:

- A domain set $\mathcal{X} = \mathcal{X}^1 \times \cdots \times \mathcal{X}^d$
- An unknown probability distribution $\mathcal{D}$ on $\mathcal{X}$
- A target set $\mathcal{Y}$
- An unknown labelling function $f : \mathcal{X} \to \mathcal{Y}$
- A training set $S = ((x_1, y_1), \ldots, (x_m, y_m))$ sampled from $\mathcal{D}$ and f

Supervised learning

Given a supervised learning problem, the learner chooses the following:

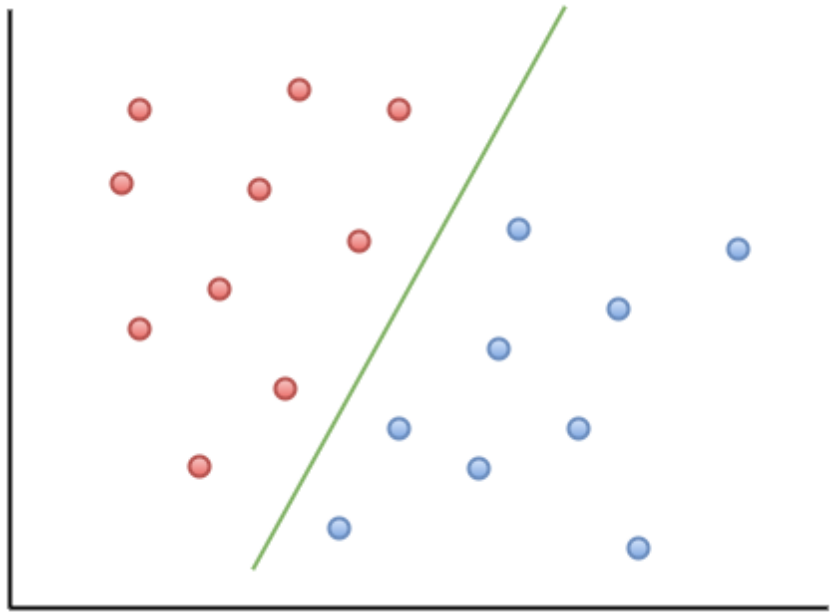
- A hypothesis class $\mathcal{H}$ of candidate labelling functions
- A loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$
- An algorithm $\mathcal{A}$ that minimizes the empirical risk

Target set

- Target set (label space) $\mathcal{Y}$ represents what we want to predict
- The machine learning problem depends on the form of $\mathcal{Y}$:

$\mathcal{Y} = \{c_1, \ldots, c_k\}$:	classification (target is a class)
$\mathcal{Y} = \mathbb{R}$:	regression (target is a real number)
$\mathcal{Y} = [0, 1]$:	logistic regression (target is a probability)

Linear models



- Simplest way to separate data is a line (or a hyperplane in higher dimensions)
- Hypothesis class: linear combination of features

Content

Perceptron

- Algorithm for binary classification: $\mathcal{Y} = \{-1, +1\}$
- Assume inputs $x = (x_1, \ldots, x_d)$ on d numerical features
- Compute a weighted score and output $+1$ if

$$\sum_{i=1}^{d} w_i x_i > \theta,$$

otherwise output -1

Perceptron

- Sign function $\text{sign}(x)$ outputs $+1$ if $x > 0$ and -1 otherwise
- Hypothesis h completely defined by weights $w_0, \ldots, w_d$:

$$h(x) = \text{sign}\left(\sum_{i=1}^{d} w_i x_i - \theta\right) = \text{sign}\left(\sum_{i=0}^{d} w_i x_i\right) = \text{sign}\left(w^\top x\right)$$

where $x_0 = 1$ is a dummy feature and $w_0 = -\theta$

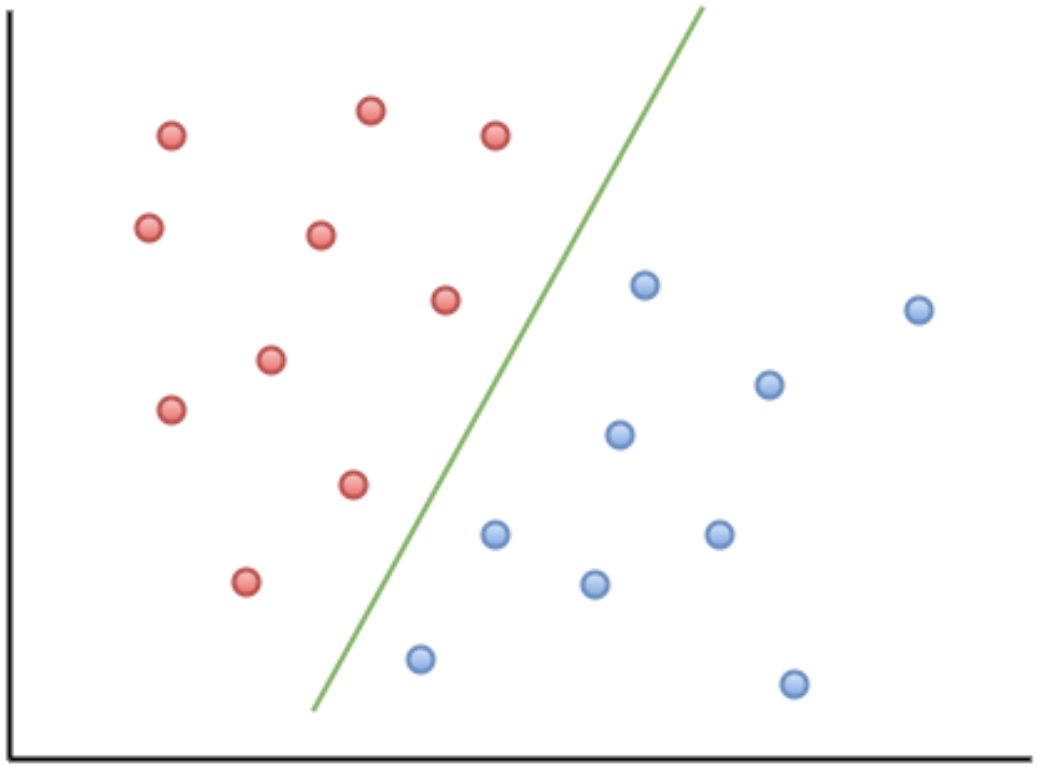
Perceptron

- Sign function $\mathrm{sign}(x)$ outputs $+1$ if $x > 0$ and -1 otherwise
- Hypothesis h completely defined by weights $w_0, \ldots, w_d$:

$$h(x) = \mathrm{sign}\left(\sum_{i=1}^{d} w_i x_i - \theta\right) = \mathrm{sign}\left(\sum_{i=0}^{d} w_i x_i\right) = \mathrm{sign}\left(w^\top x\right)$$

  where $x_0 = 1$ is a dummy feature and $w_0 = -\theta$
- $\mathcal{H}$ is infinite (weights are real-valued)
- Classification loss: $\ell(h(x_i), y_i) = [h(x_i) \neq y_i]$
- $L_S(h) = \frac{1}{m} \sum_{i=1}^{m} [h(x_i) \neq y_i]$

Linearly separable data



Perceptron learning algorithm (PLA)

Perceptron learning algorithm

1. Initialize weight vector $w_0 = 0$
2. Find a mistake (x_i, y_i) such that $h(x_i) = \text{sign}(w_0^\top x_i) \neq y_i$
3. Update weights as $w_1 \leftarrow w_0 + y_i x_i$
4. Repeat from 2. for weight vector w_t, $t = 1, 2, \ldots$

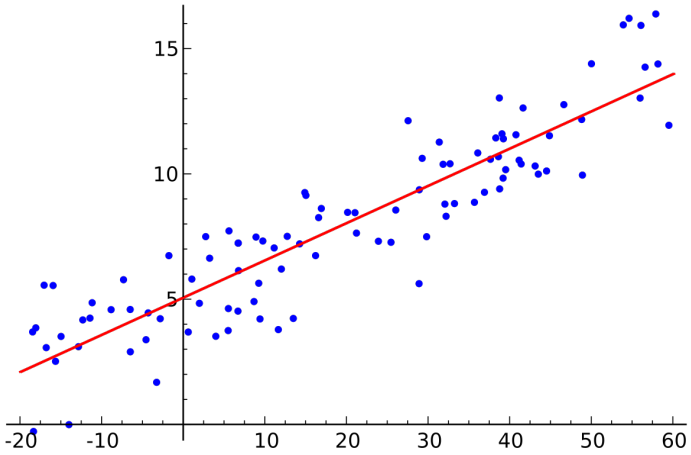
Properties

- If data is linearly separable, PLA guaranteed to converge to a hypothesis h_S (i.e. weight vector w_S) such that $L_S(h_S) = 0$
- If data is not linearly separable, PLA never converges
- Variants:
  - Fix number of iterations T, stop when $t > T$
  - Pocket algorithm: only update weight vector w_t when total number of mistakes decreases

Content

Linear regression

- Assumes regression problem ($\mathcal{Y} = \mathbb{R}$)
- Hypothesis $h(x) = \sum_{i=0}^{d} w_i x_i = w^\top x$
- Squared loss: $\ell(h(x_i), y_i) = (h(x_i) - y_i)^2$
- $L_S(h) = \frac{1}{m} \sum_{i=1}^{m} (h(x_i) - y_i)^2$ is the mean squared error (MSE)



Linear regression

- Hypothesis h defined by weight vector $w = (w_0, \ldots, w_d)$
- Find w that minimizes

$$L_S(w) = \frac{1}{m} \sum_{i=1}^{m} (h(x_i) - y_i)^2 = \frac{1}{m} \sum_{i=1}^{m} (w^\top x_i - y_i)^2$$

- $L_S(w)$ is continuous, differentiable, and convex

Linear regression

$$\left\{ \begin{array}{l} L_S(w) = \frac{1}{m} \sum_{i=1}^{m} (w^\top x_i - y_i)^2 = \frac{1}{m} \left\| \begin{pmatrix} w^\top x_1 - y_1 \\ \vdots \\ w^\top x_m - y_m \end{pmatrix} \right\|^2 \end{array} \right.$$

Linear regression

$$
\begin{cases}
L_S(w) = \dfrac{1}{m}\displaystyle\sum_{i=1}^{m}(w^\top x_i - y_i)^2 = \dfrac{1}{m}\left\|\begin{pmatrix} w^\top x_1 - y_1 \\ \vdots \\ w^\top x_m - y_m \end{pmatrix}\right\|^2 \\
\qquad = \dfrac{1}{m}\left\|\begin{pmatrix} — & x_1 & — \\ — & x_2 & — \\ & \vdots & \\ — & x_m & — \end{pmatrix}\begin{pmatrix} w_0 \\ \vdots \\ w_d \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}\right\|^2
\end{cases}
$$

Linear regression

$$
\begin{cases}
L_S(w) = \dfrac{1}{m}\displaystyle\sum_{i=1}^{m}(w^\top x_i - y_i)^2 = \dfrac{1}{m}\left\|\begin{pmatrix} w^\top x_1 - y_1 \\ \vdots \\ w^\top x_m - y_m \end{pmatrix}\right\|^2 \\
\qquad = \dfrac{1}{m}\left\|\begin{pmatrix} — & x_1 & — \\ — & x_2 & — \\ & \vdots & \\ — & x_m & — \end{pmatrix}\begin{pmatrix} w_0 \\ \vdots \\ w_d \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}\right\|^2 \\
\qquad = \dfrac{1}{m}\left\|Xw - y\right\|^2 = \dfrac{1}{m}\left(w^\top X^\top X w - 2w^\top X^\top y + y^\top y\right)
\end{cases}
$$

Linear regression

$$
\begin{cases}
L_S(w) = \dfrac{1}{m}\displaystyle\sum_{i=1}^{m}(w^\top x_i - y_i)^2 = \dfrac{1}{m}\left\|\begin{pmatrix} w^\top x_1 - y_1 \\ \vdots \\ w^\top x_m - y_m \end{pmatrix}\right\|^2 \\
\qquad = \dfrac{1}{m}\left\|\begin{pmatrix} — & x_1 & — \\ — & x_2 & — \\ & \vdots & \\ — & x_m & — \end{pmatrix}\begin{pmatrix} w_0 \\ \vdots \\ w_d \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}\right\|^2 \\
\qquad = \dfrac{1}{m}\left\|Xw - y\right\|^2 = \dfrac{1}{m}\left(w^\top X^\top X w - 2w^\top X^\top y + y^\top y\right)
\end{cases}
$$

Linear regression

$$
\begin{cases}
L_S(w) = \dfrac{1}{m}\displaystyle\sum_{i=1}^{m}(w^\top x_i - y_i)^2 = \dfrac{1}{m}\left\|\begin{pmatrix} w^\top x_1 - y_1 \\ \vdots \\ w^\top x_m - y_m \end{pmatrix}\right\|^2 \\
\qquad = \dfrac{1}{m}\left\|\begin{pmatrix} — & x_1 & — \\ — & x_2 & — \\ & \vdots & \\ — & x_m & — \end{pmatrix}\begin{pmatrix} w_0 \\ \vdots \\ w_d \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}\right\|^2 \\
\qquad = \dfrac{1}{m}\left\|Xw - y\right\|^2 = \dfrac{1}{m}\left(w^\top X^\top X w - 2w^\top X^\top y + y^\top y\right)
\end{cases}
$$

- X: $m \times (d+1)$ matrix of inputs
- y: $m \times 1$ vector of labels

Linear regression

- Minimize $L_S(w)$ $\Leftrightarrow$ set gradient $\nabla_w L_S(w)$ to 0

$$\nabla_w L_S(w) = \begin{pmatrix} \frac{\partial L_S(w)}{\partial w_0} \\ \vdots \\ \frac{\partial L_S(w)}{\partial w_d} \end{pmatrix}$$

Linear regression

- Minimize $L_S(w)$ $\Leftrightarrow$ set gradient $\nabla_w L_S(w)$ to 0

$$\nabla_w L_S(w) = \begin{pmatrix} \frac{\partial L_S(w)}{\partial w_0} \\ \vdots \\ \frac{\partial L_S(w)}{\partial w_d} \end{pmatrix}$$

- Single term $(w^\top x_1 - y_1)^2$, single weight w_0:

$$\frac{\partial (w^\top x_1 - y_1)^2}{\partial w_0} = 2(w^\top x_1 - y_1) x_{10}$$

Linear regression

- Minimize $L_S(w)$ $\Leftrightarrow$ set gradient $\nabla_w L_S(w)$ to 0

$$\nabla_w L_S(w) = \begin{pmatrix} \frac{\partial L_S(w)}{\partial w_0} \\ \vdots \\ \frac{\partial L_S(w)}{\partial w_d} \end{pmatrix}$$

- Single term $(w^\top x_1 - y_1)^2$, single weight w_0:

$$\frac{\partial (w^\top x_1 - y_1)^2}{\partial w_0} = 2(w^\top x_1 - y_1) x_{10}$$

- Gradient

$$\nabla_w L_S(w) = \frac{2}{m}(X^\top X w - X^\top y)$$

Linear regression

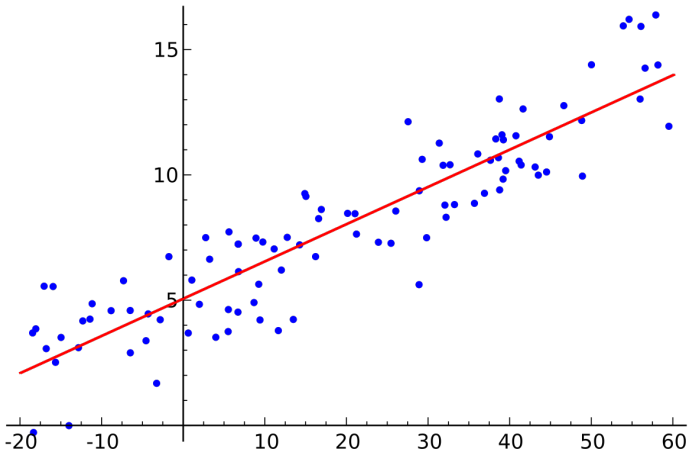
- $\nabla_w L_S(w) = 0 \quad \Leftrightarrow \quad X^\top X w = X^\top y$
- $X^\top X$ is a $(d+1) \times (d+1)$ matrix

Linear regression

- $\nabla_w L_S(w) = 0 \quad \Leftrightarrow \quad X^\top X w = X^\top y$
- $X^\top X$ is a $(d+1) \times (d+1)$ matrix

- $X^\top X$ invertible: Analytic solution $w_{\text{lin}} = (X^\top X)^{-1} X^\top y = X^\dagger y$
- $X^\dagger = (X^\top X)^{-1} X^\top$: pseudo-inverse of X
- In practice: use well-implemented $\dagger$ routine for computing $X^\dagger$

Hat matrix



- Approximate labels on inputs $x_1, \ldots, x_m$:

$$\hat{y} = Xw_{\text{lin}} = X(X^\top X)^{-1}X^\top y = Hy$$

- $H = X(X^\top X)^{-1}X^\top$ is called the hat matrix since it puts the "hat" on y

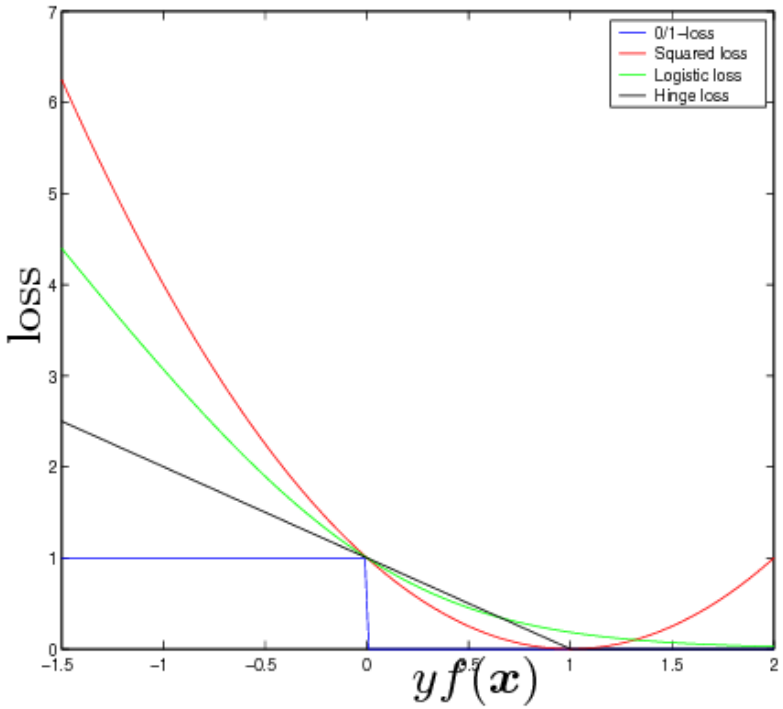
Is linear regression really “learning”?

- No, in the sense that w_{lin} has an analytical solution
- No algorithm necessary for iteratively improving the training loss
- Yes, in the sense that we achieve a small training loss $L_S(w)$
- Algorithm for computing the pseudo-inverse $X^\dagger = (X^\top X)^{-1} X^\top$

Linear classification vs. linear regression

- Minimizing $L_S(h) = \frac{1}{m}\sum_{i=1}^{m}[h(x_i) \neq y_i]$ is NP-hard
- Minimizing $L_S(h) = \frac{1}{m}\sum_{i=1}^{m}(h(x_i) - y_i)^2$ has an efficient analytical solution
- Idea: $\{-1, +1\} \subset \mathbb{R} \Rightarrow$ use linear regression for classification!
- On input x, predict label $\mathrm{sign}(w_{\mathrm{lin}}^{\top} x)$

Relationship between loss functions



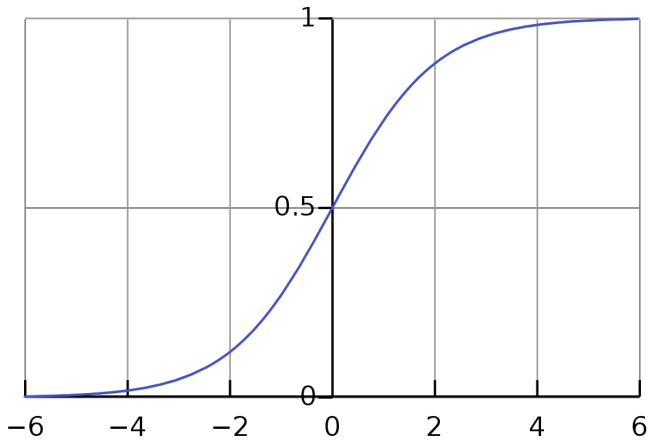
- For label $+1$, square loss upper bounds 0-1 loss! (same for -1)
- Sacrifice bound tightness for efficiency

Content

Logistic regression

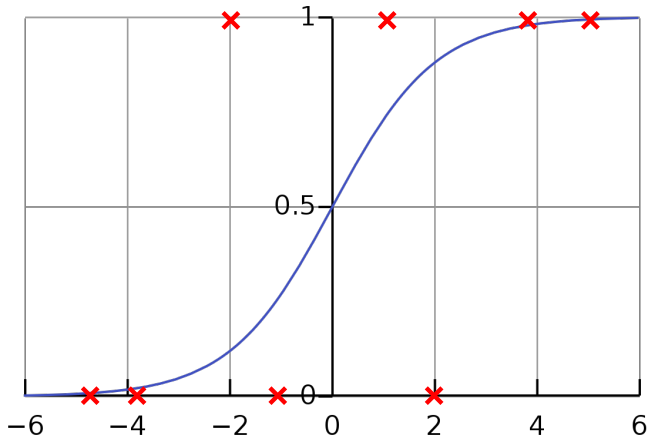
- Soft classification: estimate probability of belonging to class
- Hypothesis $h(x) = \theta(\sum_{i=0}^{d} w_i x_i) = \theta(w^\top x)$
- Logistic function

$$\theta(s) = \frac{1}{1 + e^{-s}}$$



Logistic regression

- Assume that there are two classes $\{-1, +1\}$
- Ideally, data would be on the form $((x_1, 0.8), \ldots, (x_m, 0.1))$, i.e. the probability of belonging to class $+1$
- However, data is usually on the form $((x_1, +1), \ldots, (x_m, -1))$
- We can view labels as hard probabilities, i.e. 0 or 1



Logistic loss

- Find w that minimizes

$$L_S(w) = \frac{1}{m} \sum_{i=1}^{m} \ell(h(x_i), y_i)$$

- Cross-entropy loss $\ell(h(x_i), y_i) = \ln(1 + \exp(-y_i w^\top x_i))$
- $L_S(w)$ is continuous, differentiable, and convex

Logistic loss

- Minimize $L_S(w) \Leftrightarrow$ Find w such that $\nabla_w L_S(w) = 0$
- Gradient

$$\nabla_w L_S(w) = \frac{1}{m} \sum_{i=1}^{m} \theta(-y_i w^\top x_i)(-y_i x_i)$$

- Unfortunately, no analytic solution to $\nabla_w L_S(w) = 0$

Logistic loss

- Minimize $L_S(w) \Leftrightarrow$ Find w such that $\nabla_w L_S(w) = 0$
- Gradient

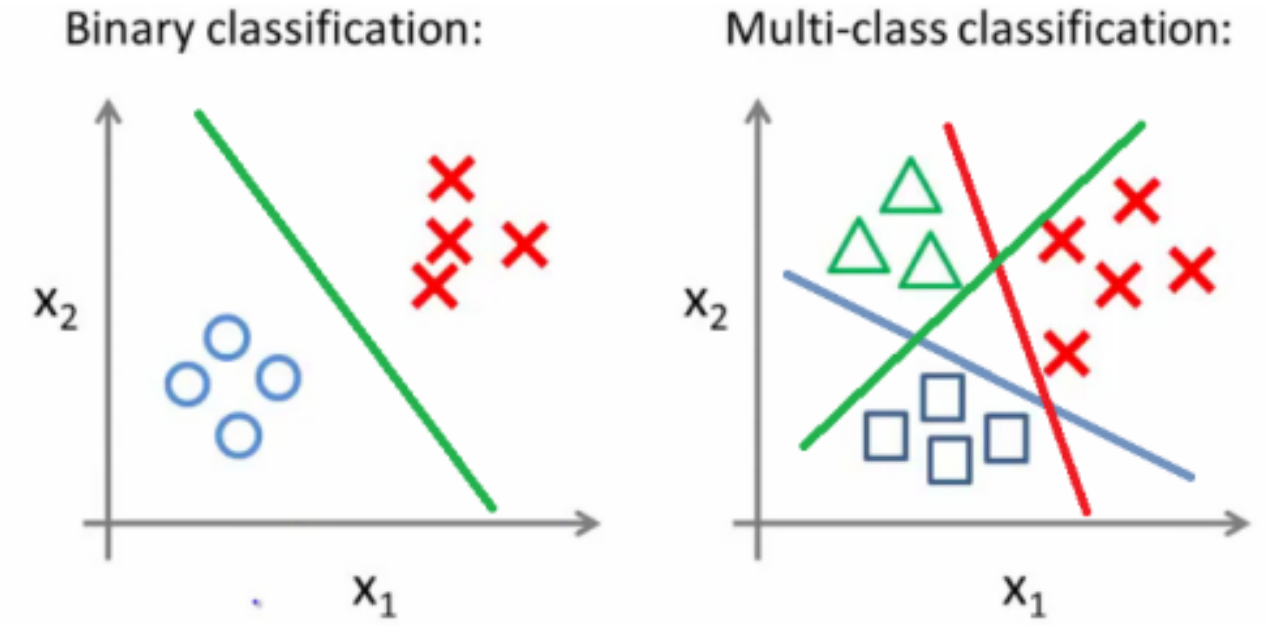
$$\nabla_w L_S(w) = \frac{1}{m} \sum_{i=1}^{m} \theta(-y_i w^\top x_i)(-y_i x_i)$$

- Unfortunately, no analytic solution to $\nabla_w L_S(w) = 0$
- (Stay tuned for optimization and gradient descent)

Multiclass classification

- So far we have mainly talked about binary classification, i.e. $|\mathcal{Y}| = 2$
- What if there are more than two classes, i.e. $2 < |\mathcal{Y}| = k$?
- Two main approaches:
  1. One-versus-all (OVA): binary classification of one class vs. rest
  2. One-versus-one (OVO): binary classification of pairs of classes
- In both cases, perform soft binary classification (i.e. logistic regression) and return most probable class

One-versus-all



Multiclass classification

Properties of one-versus-all (OVA):

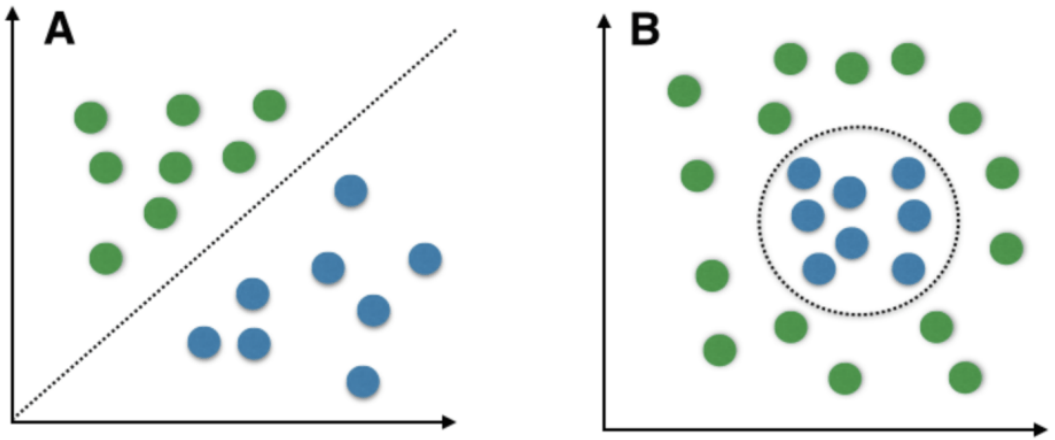
- Linear number of classifiers
- Imbalanced data!

Properties of one-versus-one (OVO):

- Quadratic number of classifiers
- More balanced data

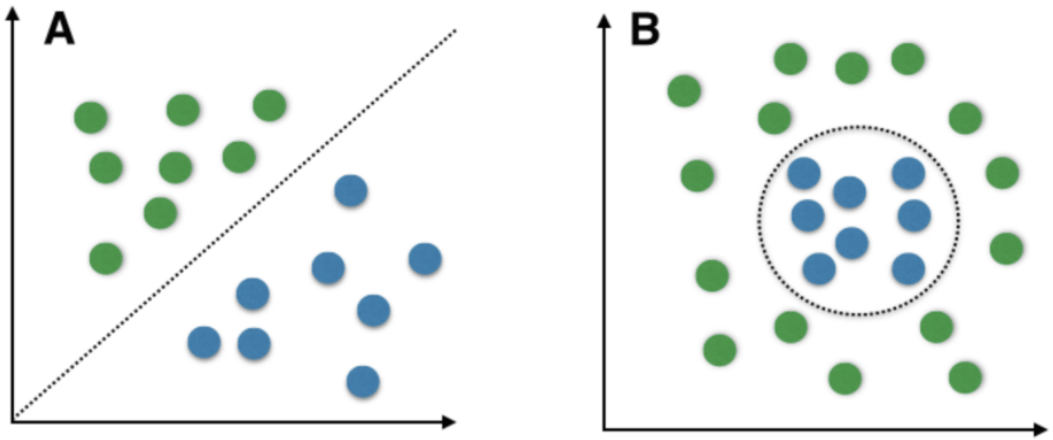
Content

Non-linear data



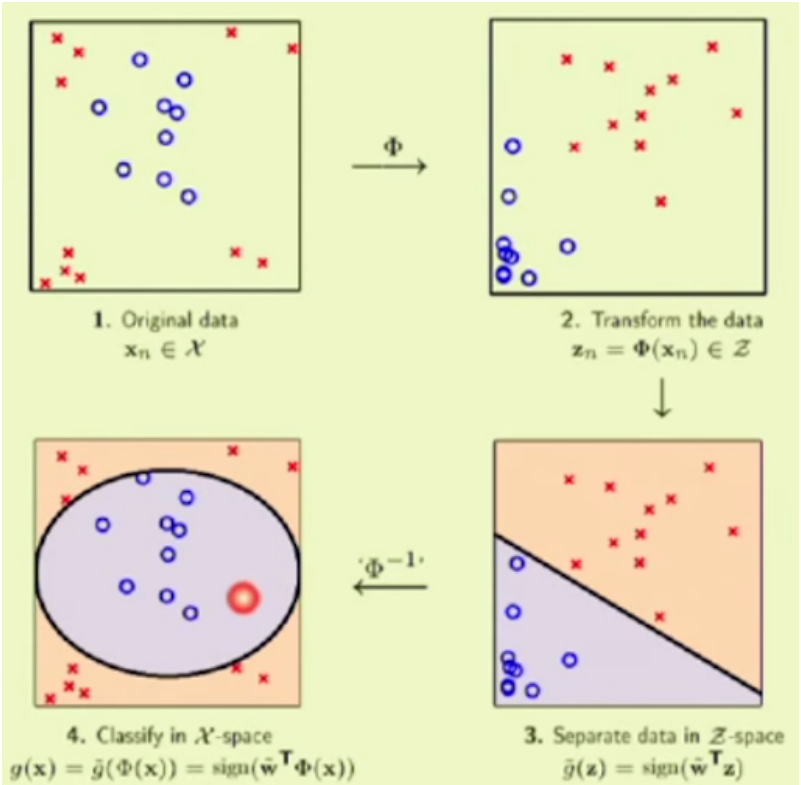
- Often data is not linearly separable at all
- Idea: derive circular perceptron, circular regression, etc.
- It would be better to take advantage of linear models!

Non-linear transforms



- Circular hypothesis: $h(x) = \text{sign}(0.6 - x_1^2 - x_2^2)$
- Linear in quadratic terms x_1^2 and x_2^2!
- Non-linear transform: introduce additional non-linear terms
- Quadratic transform: $z = \Phi(x) = (1, x_1, x_2, x_1 x_2, x_1^2, x_2^2)$

Non-linear transforms



- Transform each data point (x_i, y_i) to $(z_i = \Phi(x_i), y_i)$
- Apply linear algorithm in transformed space to find $\tilde{w}$
- Hypothesis on input x proportional to $\tilde{w}^\top \Phi(x)$

Price of non-linear transforms

- Let $\mathcal{H}_q$ be the hypothesis class of q-th order polynomials
- Then $\mathcal{H}_1 \subset \mathcal{H}_2 \subset \cdots \subset \mathcal{H}_q$
- However, number of features increases!
- The q-th order polynomial has $O(d^q)$ dimensions
- Increases memory requirements, running time of algorithms, etc.

Content

Multiclass classification

- Assume that a binary classification algorithm $\mathcal{A}$ runs in time N^3 on data of size N
- For a 10 class classification problem ($|\mathcal{Y}| = 10$), assume that there are exactly $N/10$ data points for each class
- What is the running time of OVA and OVO multiclass classification using algorithm $\mathcal{A}$?

One-versus-all (OVA)

- We have to build 10 classifiers in total

One-versus-all (OVA)

- We have to build 10 classifiers in total
- Each classifier uses all data, i.e. N data points

One-versus-all (OVA)

- We have to build 10 classifiers in total
- Each classifier uses all data, i.e. N data points
- For each classifier, the running time of algorithm $\mathcal{A}$ is N^3

One-versus-all (OVA)

- We have to build 10 classifiers in total
- Each classifier uses all data, i.e. N data points
- For each classifier, the running time of algorithm $\mathcal{A}$ is N^3
- Hence the total running time is $10N^3$

One-versus-one (OVO)

- We have to build $\binom{10}{2} = \frac{10 \cdot 9}{2} = 45$ classifiers in total

One-versus-one (OVO)

- We have to build $\begin{pmatrix} 10 \\ 2 \end{pmatrix} = \frac{10 \cdot 9}{2} = 45$ classifiers in total
- Each classifier uses only data points of 2 classes, i.e. $2N/10$

One-versus-one (OVO)

- We have to build $\begin{pmatrix} 10 \\ 2 \end{pmatrix} = \frac{10 \cdot 9}{2} = 45$ classifiers in total
- Each classifier uses only data points of 2 classes, i.e. $2N/10$
- For each classifier, the running time of algorithm $\mathcal{A}$ is $8N^3/1000$

One-versus-one (OVO)

- We have to build $\binom{10}{2} = \frac{10 \cdot 9}{2} = 45$ classifiers in total
- Each classifier uses only data points of 2 classes, i.e. $2N/10$
- For each classifier, the running time of algorithm $\mathcal{A}$ is $8N^3/1000$
- Hence the total running time is $45 \cdot 8N^3/1000 = \frac{9}{25}N^3$

Non-linear transforms

- Consider the hypothesis class $\mathcal{H}_2$ of second-order polynomials
- Let d be the number of dimensions of the original inputs $x \in \mathcal{X}$
- What is the number of dimensions of transformed inputs $z = \Phi(x)$?
- Recall that the features are $1, x_1, \ldots, x_d, x_1^2, x_1 x_2, x_2^2, \ldots$

Non-linear transforms

- $\mathcal{H}_2$ contains quadratic, linear and constant terms

Non-linear transforms

- $\mathcal{H}_2$ contains quadratic, linear and constant terms
- The number of quadratic terms is $\binom{d}{2} + d = \frac{d(d-1)+2d}{2} = \frac{d^2+d}{2}$

Non-linear transforms

- $\mathcal{H}_2$ contains quadratic, linear and constant terms
- The number of quadratic terms is $\binom{d}{2} + d = \frac{d(d-1)+2d}{2} = \frac{d^2+d}{2}$
- The number of linear terms is d

Non-linear transforms

- $\mathcal{H}_2$ contains quadratic, linear and constant terms
- The number of quadratic terms is $\binom{d}{2} + d = \frac{d(d-1)+2d}{2} = \frac{d^2+d}{2}$
- The number of linear terms is d
- The number of constant terms is 1 (the dummy feature $+1$)

Non-linear transforms

- $\mathcal{H}_2$ contains quadratic, linear and constant terms
- The number of quadratic terms is $\binom{d}{2} + d = \frac{d(d-1)+2d}{2} = \frac{d^2+d}{2}$
- The number of linear terms is d
- The number of constant terms is 1 (the dummy feature $+1$)
- In total, there are $\frac{d^2+d}{2} + d + 1 = \frac{d^2}{2} + \frac{3d}{2} + 1$ terms

Linear regression

The training loss for linear regression is given by

$$L_S(w) = \frac{1}{m}(w^\top X^\top X w - 2w^\top X^\top y + y^\top y). \tag{1}$$

Assuming that $X^\top X$ is invertible, show that $L_S(w)$ can be written as

$$L_S(w) = \frac{1}{m}((w - w_{\text{lin}})^\top (X^\top X)(w - w_{\text{lin}}) + y^\top (y - X w_{\text{lin}}), \tag{2}$$

where $w_{\text{lin}} = (X^\top X)^{-1} X^\top y$ is the weight vector that minimizes empirical risk

Linear regression

Multiplying $X^\top X$ with w_{lin} yields

$$(X^\top X)w_{\text{lin}} = (X^\top X)(X^\top X)^{-1}X^\top y = I X^\top y = X^\top y.$$

Using the properties of the transpose yields

$$w_{\text{lin}}^\top (X^\top X) = ((X^\top X)w_{\text{lin}})^\top = (X^\top y)^\top = y^\top X.$$

Linear regression

Show that simplifying (2) results in (1):

$$\begin{cases} mL_S(w) = (w - w_{\text{lin}})^\top (X^\top X)(w - w_{\text{lin}}) + y^\top (y - X w_{\text{lin}}) \end{cases}$$

Linear regression

Show that simplifying (2) results in (1):

$$
\begin{cases}
mL_S(w) = (w - w_{\text{lin}})^\top (X^\top X)(w - w_{\text{lin}}) + y^\top (y - X w_{\text{lin}}) \\
\quad = (w - w_{\text{lin}})^\top (X^\top X w - X^\top X w_{\text{lin}}) + y^\top (y - X w_{\text{lin}})
\end{cases}
$$

Linear regression

Show that simplifying (2) results in (1):

$$
\begin{aligned}
m L_S(w) &= (w - w_{\text{lin}})^\top (X^\top X)(w - w_{\text{lin}}) + y^\top (y - X w_{\text{lin}}) \\
&= (w - w_{\text{lin}})^\top (X^\top X w - X^\top X w_{\text{lin}}) + y^\top (y - X w_{\text{lin}}) \\
&= (w - w_{\text{lin}})^\top (X^\top X w - X^\top y) + y^\top (y - X w_{\text{lin}})
\end{aligned}
$$

Linear regression

Show that simplifying (2) results in (1):

$$
\begin{cases}
mL_S(w) = (w - w_{\text{lin}})^\top (X^\top X)(w - w_{\text{lin}}) + y^\top (y - X w_{\text{lin}}) \\
\quad = (w - w_{\text{lin}})^\top (X^\top X w - X^\top X w_{\text{lin}}) + y^\top (y - X w_{\text{lin}}) \\
\quad = (w - w_{\text{lin}})^\top (X^\top X w - X^\top y) + y^\top (y - X w_{\text{lin}}) \\
\quad = w^\top X^\top X w - w_{\text{lin}}^\top X^\top X w - w^\top X^\top y + w_{\text{lin}}^\top X^\top y + y^\top y - y^\top X w_{\text{lin}}
\end{cases}
$$

Linear regression

Show that simplifying (2) results in (1):

$$
\begin{cases}
mL_S(w) = (w - w_{\text{lin}})^\top (X^\top X)(w - w_{\text{lin}}) + y^\top (y - X w_{\text{lin}}) \\
\quad = (w - w_{\text{lin}})^\top (X^\top X w - X^\top X w_{\text{lin}}) + y^\top (y - X w_{\text{lin}}) \\
\quad = (w - w_{\text{lin}})^\top (X^\top X w - X^\top y) + y^\top (y - X w_{\text{lin}}) \\
\quad = w^\top X^\top X w - w_{\text{lin}}^\top X^\top X w - w^\top X^\top y + w_{\text{lin}}^\top X^\top y + y^\top y - y^\top X w_{\text{lin}} \\
\quad = w^\top X^\top X w - y^\top X w - w^\top X^\top y + w_{\text{lin}}^\top X^\top y + y^\top y - y^\top X w_{\text{lin}}
\end{cases}
$$

Linear regression

Show that simplifying (2) results in (1):

$$
\begin{cases}
mL_S(w) = (w - w_{\text{lin}})^\top (X^\top X)(w - w_{\text{lin}}) + y^\top (y - X w_{\text{lin}}) \\
\quad = (w - w_{\text{lin}})^\top (X^\top X w - X^\top X w_{\text{lin}}) + y^\top (y - X w_{\text{lin}}) \\
\quad = (w - w_{\text{lin}})^\top (X^\top X w - X^\top y) + y^\top (y - X w_{\text{lin}}) \\
\quad = w^\top X^\top X w - w_{\text{lin}}^\top X^\top X w - w^\top X^\top y + w_{\text{lin}}^\top X^\top y + y^\top y - y^\top X w_{\text{lin}} \\
\quad = w^\top X^\top X w - y^\top X w - w^\top X^\top y + w_{\text{lin}}^\top X^\top y + y^\top y - y^\top X w_{\text{lin}} \\
\quad = \{y^\top X w \text{ and } y^\top X w_{\text{lin}} \text{ are scalars } \Leftrightarrow y^\top X w = w^\top X^\top y\}
\end{cases}
$$

Linear regression

Show that simplifying (2) results in (1):

$$
\begin{cases}
mL_S(w) = (w - w_{\text{lin}})^\top (X^\top X)(w - w_{\text{lin}}) + y^\top (y - X w_{\text{lin}}) \\
\quad = (w - w_{\text{lin}})^\top (X^\top X w - X^\top X w_{\text{lin}}) + y^\top (y - X w_{\text{lin}}) \\
\quad = (w - w_{\text{lin}})^\top (X^\top X w - X^\top y) + y^\top (y - X w_{\text{lin}}) \\
\quad = w^\top X^\top X w - w_{\text{lin}}^\top X^\top X w - w^\top X^\top y + w_{\text{lin}}^\top X^\top y + y^\top y - y^\top X w_{\text{lin}} \\
\quad = w^\top X^\top X w - y^\top X w - w^\top X^\top y + w_{\text{lin}}^\top X^\top y + y^\top y - y^\top X w_{\text{lin}} \\
\quad = \{y^\top X w \text{ and } y^\top X w_{\text{lin}} \text{ are scalars } \Leftrightarrow y^\top X w = w^\top X^\top y\} \\
\quad = w^\top X^\top X w - w^\top X^\top y - w^\top X^\top y + \cancel{w_{\text{lin}}^\top X^\top y} + y^\top y - \cancel{w_{\text{lin}}^\top X^\top y}
\end{cases}
$$

Linear regression

Show that simplifying (2) results in (1):

$$
\begin{cases}
mL_S(w) = (w - w_{\text{lin}})^\top (X^\top X)(w - w_{\text{lin}}) + y^\top (y - X w_{\text{lin}}) \\
\quad = (w - w_{\text{lin}})^\top (X^\top X w - X^\top X w_{\text{lin}}) + y^\top (y - X w_{\text{lin}}) \\
\quad = (w - w_{\text{lin}})^\top (X^\top X w - X^\top y) + y^\top (y - X w_{\text{lin}}) \\
\quad = w^\top X^\top X w - w_{\text{lin}}^\top X^\top X w - w^\top X^\top y + w_{\text{lin}}^\top X^\top y + y^\top y - y^\top X w_{\text{lin}} \\
\quad = w^\top X^\top X w - y^\top X w - w^\top X^\top y + w_{\text{lin}}^\top X^\top y + y^\top y - y^\top X w_{\text{lin}} \\
\quad = \{y^\top X w \text{ and } y^\top X w_{\text{lin}} \text{ are scalars } \Leftrightarrow y^\top X w = w^\top X^\top y\} \\
\quad = w^\top X^\top X w - w^\top X^\top y - w^\top X^\top y + \cancel{w_{\text{lin}}^\top X^\top y} + y^\top y - \cancel{w_{\text{lin}}^\top X^\top y} \\
\quad = w^\top X^\top X w - 2 w^\top X^\top y + y^\top y
\end{cases}
$$

Linear regression

The matrix $X^\top X$ is positive definite, i.e. for any vector v,

$$v^\top (X^\top X) v \geq 0.$$

Trivially, the expression

$$L_S(w) = \frac{1}{m}((w - w_{\text{lin}})^\top (X^\top X)(w - w_{\text{lin}}) + y^\top (y - X w_{\text{lin}})$$

is minimized when $w - w_{\text{lin}} = 0$, i.e. when $w = w_{\text{lin}}$

Perceptron learning algorithm (PLA)

Perceptron learning algorithm

1. Initialize weight vector $w_0 = 0$
2. Find a mistake (x_i, y_i) such that $h(x_i) \neq y_i$
3. Update weights as $w_1 \leftarrow w_0 + y_i x_i$
4. Repeat from 2. for weight vector w_t, $t = 1, 2, \ldots$

Letting $R^2 = \max_i \|x_i\|^2$, show that after t iterations, $\|w_t\|^2 \leq tR^2$

Perceptron learning algorithm (PLA)

Consider a single step of PLA:

$$\begin{cases} \|w_{t+1}\|^2 = \|w_t + y_i x_i\|^2 \\ \\ \\ \\ \end{cases}$$

Perceptron learning algorithm (PLA)

Consider a single step of PLA:

$$\begin{cases} \|w_{t+1}\|^2 = \|w_t + y_i x_i\|^2 \\ \qquad\qquad = \|w_t\|^2 + 2y_i w_t^\top x_i + y_i^2 \|x_i\|^2 \end{cases}$$

Perceptron learning algorithm (PLA)

Consider a single step of PLA:

$$\begin{cases} \|w_{t+1}\|^2 = \|w_t + y_i x_i\|^2 \\ \qquad\qquad = \|w_t\|^2 + 2y_i w_t^\top x_i + y_i^2 \|x_i\|^2 \\ \qquad\qquad \leq \|w_t\|^2 + 0 + \|x_i\|^2 \end{cases}$$

Perceptron learning algorithm (PLA)

Consider a single step of PLA:

$$\begin{cases} \|w_{t+1}\|^2 = \|w_t + y_i x_i\|^2 \\ \qquad\qquad = \|w_t\|^2 + 2y_i w_t^\top x_i + y_i^2 \|x_i\|^2 \\ \qquad\qquad \leq \|w_t\|^2 + 0 + \|x_i\|^2 \\ \qquad\qquad \leq \|w_t\|^2 + R^2 \end{cases}$$

Perceptron learning algorithm (PLA)

Consider a single step of PLA:

$$\begin{cases} \|w_{t+1}\|^2 = \|w_t + y_i x_i\|^2 \\ \qquad\qquad = \|w_t\|^2 + 2y_i w_t^\top x_i + y_i^2 \|x_i\|^2 \\ \qquad\qquad \leq \|w_t\|^2 + 0 + \|x_i\|^2 \\ \qquad\qquad \leq \|w_t\|^2 + R^2 \end{cases}$$

Perceptron learning algorithm (PLA)

Consider a single step of PLA:

$$\begin{cases} \|w_{t+1}\|^2 = \|w_t + y_i x_i\|^2 \\ \qquad\qquad = \|w_t\|^2 + 2y_i w_t^\top x_i + y_i^2 \|x_i\|^2 \\ \qquad\qquad \leq \|w_t\|^2 + 0 + \|x_i\|^2 \\ \qquad\qquad \leq \|w_t\|^2 + R^2 \end{cases}$$

Hence after t iterations, $\|w_t\|^2 \leq \|w_0\|^2 + tR^2 = 0 + tR^2 = tR^2$

Perceptron learning algorithm (PLA)

Data linearly separable $\Rightarrow$ exists w_* such that $y_i w_*^\top x_i > 0,\ \forall i \in [m]$

Perceptron learning algorithm

1. Initialize weight vector $w_0 = 0$
2. Find a mistake (x_i, y_i) such that $h(x_i) \neq y_i$
3. Update weights as $w_1 \leftarrow w_0 + y_i x_i$
4. Repeat from 2. for weight vector w_t, $t = 1, 2, \ldots$

Letting $\rho = \min_i y_i \frac{w_*^\top}{\|w_*\|} x_i > 0$, show that after t iterations, $\frac{w_*^\top w_t}{\|w_*\|} \geq t\rho$

Perceptron learning algorithm (PLA)

Consider a single step of PLA:

$$\begin{cases} \dfrac{w_*^\top w_{t+1}}{\|w_*\|} = \dfrac{w_*^\top w_t}{\|w_*\|} + y_i \dfrac{w_*^\top}{\|w_*\|} x_i \end{cases}$$

Perceptron learning algorithm (PLA)

Consider a single step of PLA:

$$\begin{cases} \dfrac{w_*^\top w_{t+1}}{\|w_*\|} = \dfrac{w_*^\top w_t}{\|w_*\|} + y_i \dfrac{w_*^\top}{\|w_*\|} x_i \\ \qquad\qquad \geq \dfrac{w_*^\top w_t}{\|w_*\|} + \rho \end{cases}$$

Perceptron learning algorithm (PLA)

Consider a single step of PLA:

$$\begin{cases} \dfrac{w_*^\top w_{t+1}}{\|w_*\|} = \dfrac{w_*^\top w_t}{\|w_*\|} + y_i \dfrac{w_*^\top}{\|w_*\|} x_i \\ \qquad\qquad \geq \dfrac{w_*^\top w_t}{\|w_*\|} + \rho \end{cases}$$

Perceptron learning algorithm (PLA)

Consider a single step of PLA:

$$\begin{cases} \dfrac{w_*^\top w_{t+1}}{\|w_*\|} = \dfrac{w_*^\top w_t}{\|w_*\|} + y_i \dfrac{w_*^\top}{\|w_*\|} x_i \\ \qquad\qquad \geq \dfrac{w_*^\top w_t}{\|w_*\|} + \rho \end{cases}$$

Hence after t iterations, $\frac{w_*^\top w_t}{\|w_*\|} \geq \frac{w_*^\top w_0}{\|w_*\|} + t\rho = 0 + t\rho = t\rho$

Perceptron learning algorithm (PLA)

- Putting the two results together, we get

$$\frac{w_*^\top w_t}{\|w_*\|\|w_t\|} \geq \frac{t\rho}{\sqrt{t}R} = \sqrt{t}\frac{\rho}{R}$$

Perceptron learning algorithm (PLA)

- Putting the two results together, we get

$$\frac{w_*^\top w_t}{\|w_*\|\|w_t\|} \geq \frac{t\rho}{\sqrt{t}R} = \sqrt{t}\frac{\rho}{R}$$

- The scalar product of two unit vectors is upper bounded by 1:

$$1 \geq \frac{w_*^\top w_t}{\|w_*\|\|w_t\|} \geq \sqrt{t}\frac{\rho}{R}$$

Perceptron learning algorithm (PLA)

- Putting the two results together, we get

$$\frac{w_*^\top w_t}{\|w_*\|\|w_t\|} \geq \frac{t\rho}{\sqrt{t}R} = \sqrt{t}\frac{\rho}{R}$$

- The scalar product of two unit vectors is upper bounded by 1:

$$1 \geq \frac{w_*^\top w_t}{\|w_*\|\|w_t\|} \geq \sqrt{t}\frac{\rho}{R}$$

- It follows that

$$\sqrt{t} \leq \frac{R}{\rho} \quad \Leftrightarrow \quad t \leq \frac{R^2}{\rho^2}$$

Perceptron learning algorithm (PLA)

- Putting the two results together, we get

$$\frac{w_*^\top w_t}{\|w_*\|\|w_t\|} \geq \frac{t\rho}{\sqrt{t}R} = \sqrt{t}\frac{\rho}{R}$$

- The scalar product of two unit vectors is upper bounded by 1:

$$1 \geq \frac{w_*^\top w_t}{\|w_*\|\|w_t\|} \geq \sqrt{t}\frac{\rho}{R}$$

- It follows that

$$\sqrt{t} \leq \frac{R}{\rho} \quad \Leftrightarrow \quad t \leq \frac{R^2}{\rho^2}$$

- Hence PLA converges after at most R^2/ρ^2 iterations!