Machine Learning

Lecture 2
Linear Models

Vincent Adam & Vicenç Gómez

2022-23

Content

1 Supervised learning

2 Perceptron

3 Linear regression

4 Logistic regression

5 Non-linear transforms

6 Exercises

Content

1 Supervised learning

2 Perceptron

3 Linear regression

4 Logistic regression

5 Non-linear transforms

6 Exercises

Supervised learning problem

A supervised learning problem consists of:

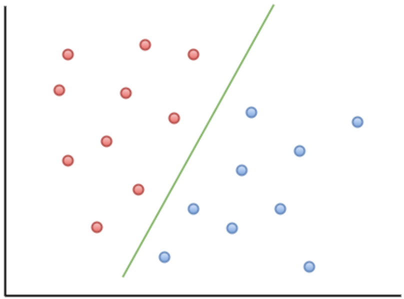
- A domain set $\mathcal{X} = \mathcal{X}^1 \times \cdots \times \mathcal{X}^d$
- An unknown probability distribution $\mathcal{D}$ on $\mathcal{X}$
- A target set $\mathcal{Y}$
- An unknown labelling function $f : \mathcal{X} \to \mathcal{Y}$
- A training set $S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m))$ 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

Linear models



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

Content

1 Supervised learning

2 Perceptron

3 Linear regression

4 Logistic regression

5 Non-linear transforms

6 Exercises

Perceptron

- Algorithm for binary classification: $\mathcal{Y} = \{-1, +1\}$

Perceptron

- Algorithm for binary classification: $\mathcal{Y} = \{-1, +1\}$
- Assume inputs $\mathbf{x} = (x_1, \ldots, x_d)$ on d numerical features

Perceptron

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

$$h(\mathbf{x}) = \begin{cases} 1 & \text{if } \sum_{i=1}^{d} w_i x_i > \theta \\ -1 & \text{otherwise} \end{cases}$$

Perceptron

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

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

Perceptron

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

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

Perceptron

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

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

Perceptron

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

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

where $x_0 = 1$ is a dummy feature
and $w_0 = -\theta$ (a.k.a. bias weight)

Perceptron

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

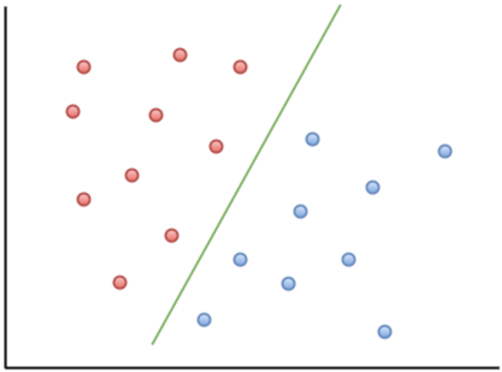
$$h(\mathbf{x}) = \mathrm{sign}\left(\sum_{i=1}^{d} w_i x_i - \theta\right) = \mathrm{sign}\left(\sum_{i=0}^{d} w_i x_i\right) = \mathrm{sign}\left(\mathbf{w}^\top \mathbf{x}\right)$$

  where $x_0 = 1$ is a dummy feature
  and $w_0 = -\theta$ (a.k.a. bias weight)
- Classification loss:

$$\ell(h(\mathbf{x}_i), y_i) = [\![h(\mathbf{x}_i) \neq y_i]\!]$$

$$L_S(h) = \frac{1}{m} \sum_{i=1}^{m} [\![h(\mathbf{x}_i) \neq y_i]\!]$$

Linearly separable data



Perceptron learning algorithm (PLA)

Builds a sequences of weights $\mathbf{w}^0, \mathbf{w}^1, \ldots, \mathbf{w}^t$

Perceptron learning algorithm

1. Initialize weight vector $\mathbf{w}^0 = 0$
2. Find a mistake $(\mathbf{x}_i, y_i)$ such that $h(\mathbf{x}_i) = \mathrm{sign}(\mathbf{w}_0^\top \mathbf{x}_i) \neq y_i$
3. Update weights as $\mathbf{w}^1 \leftarrow \mathbf{w}^0 + y_i \mathbf{x}_i$
4. Repeat from 2. for weight vector $\mathbf{w}^t$, $t = 1, 2, \ldots$

Properties

- If data is linearly separable, PLA guaranteed to converge to a hypothesis h_S (i.e. weight vector $\mathbf{w}_S$) 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 $\mathbf{w}^t$ when total number of mistakes decreases

Content

1 Supervised learning

2 Perceptron

3 Linear regression

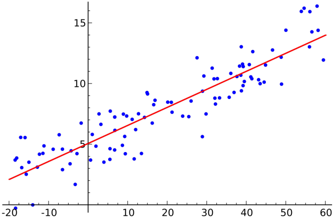
4 Logistic regression

5 Non-linear transforms

6 Exercises

Linear regression

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



Linear regression

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

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

- $L_S(\mathbf{w})$ is continuous, differentiable, and convex

Linear regression

$$\left\{ L_S(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{w}^\top \mathbf{x}_i - y_i)^2 = \frac{1}{m}\left\| \begin{pmatrix} \mathbf{w}^\top \mathbf{x}_1 - y_1 \\ \vdots \\ \mathbf{w}^\top \mathbf{x}_m - y_m \end{pmatrix} \right\|^2 \right.$$

Linear regression

$$
\begin{cases}
L_S(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{w}^\top \mathbf{x}_i - y_i)^2 = \frac{1}{m}\left\| \begin{pmatrix} \mathbf{w}^\top \mathbf{x}_1 - y_1 \\ \vdots \\ \mathbf{w}^\top \mathbf{x}_m - y_m \end{pmatrix} \right\|^2 \\
\qquad = \frac{1}{m}\left\| \begin{pmatrix} — & \mathbf{x}_1^\top & — \\ — & \mathbf{x}_2^\top & — \\ & \vdots & \\ — & \mathbf{x}_m^\top & — \end{pmatrix} \begin{pmatrix} w_0 \\ \vdots \\ w_d \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \right\|^2
\end{cases}
$$

Linear regression

$$
\begin{cases}
L_S(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{w}^\top \mathbf{x}_i - y_i)^2 = \frac{1}{m}\left\| \begin{pmatrix} \mathbf{w}^\top \mathbf{x}_1 - y_1 \\ \vdots \\ \mathbf{w}^\top \mathbf{x}_m - y_m \end{pmatrix} \right\|^2 \\
\qquad = \frac{1}{m}\left\| \begin{pmatrix} — & \mathbf{x}_1^\top & — \\ — & \mathbf{x}_2^\top & — \\ & \vdots & \\ — & \mathbf{x}_m^\top & — \end{pmatrix} \begin{pmatrix} w_0 \\ \vdots \\ w_d \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \right\|^2 \\
\qquad = \frac{1}{m}\left\| \mathbf{X}w - \mathbf{y} \right\|^2
\end{cases}
$$

Linear regression

$$
\begin{cases}
L_S(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{w}^\top \mathbf{x}_i - y_i)^2 = \frac{1}{m}\left\| \begin{pmatrix} \mathbf{w}^\top \mathbf{x}_1 - y_1 \\ \vdots \\ \mathbf{w}^\top \mathbf{x}_m - y_m \end{pmatrix} \right\|^2 \\
= \frac{1}{m}\left\| \begin{pmatrix} — & \mathbf{x}_1^\top & — \\ — & \mathbf{x}_2^\top & — \\ & \vdots & \\ — & \mathbf{x}_m^\top & — \end{pmatrix} \begin{pmatrix} w_0 \\ \vdots \\ w_d \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \right\|^2 \\
= \frac{1}{m}\|\mathbf{X}w - \mathbf{y}\|^2 = \frac{1}{m}\left(\mathbf{w}^\top \mathbf{X}^\top \mathbf{X}\mathbf{w} - 2\mathbf{w}^\top \mathbf{X}^\top \mathbf{y} + \mathbf{y}^\top \mathbf{y}\right)
\end{cases}
$$

Linear regression

$$\begin{cases} L_S(\mathbf{w}) = \dfrac{1}{m}\displaystyle\sum_{i=1}^{m}(\mathbf{w}^\top\mathbf{x}_i - y_i)^2 = \dfrac{1}{m}\left\| \begin{pmatrix} \mathbf{w}^\top\mathbf{x}_1 - y_1 \\ \vdots \\ \mathbf{w}^\top\mathbf{x}_m - y_m \end{pmatrix} \right\|^2 \\ = \dfrac{1}{m}\left\| \begin{pmatrix} — & \mathbf{x}_1^\top & — \\ — & \mathbf{x}_2^\top & — \\ & \vdots & \\ — & \mathbf{x}_m^\top & — \end{pmatrix} \begin{pmatrix} w_0 \\ \vdots \\ w_d \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \right\|^2 \\ = \dfrac{1}{m}\left\|\mathbf{X}w - \mathbf{y}\right\|^2 = \dfrac{1}{m}\left(\mathbf{w}^\top\mathbf{X}^\top\mathbf{X}\mathbf{w} - 2\mathbf{w}^\top\mathbf{X}^\top\mathbf{y} + \mathbf{y}^\top\mathbf{y}\right) \end{cases}$$

Linear regression

$$
\begin{cases}
L_S(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{w}^\top \mathbf{x}_i - y_i)^2 = \frac{1}{m}\left\| \begin{pmatrix} \mathbf{w}^\top \mathbf{x}_1 - y_1 \\ \vdots \\ \mathbf{w}^\top \mathbf{x}_m - y_m \end{pmatrix} \right\|^2 \\
= \frac{1}{m}\left\| \begin{pmatrix} — & \mathbf{x}_1^\top & — \\ — & \mathbf{x}_2^\top & — \\ & \vdots & \\ — & \mathbf{x}_m^\top & — \end{pmatrix} \begin{pmatrix} w_0 \\ \vdots \\ w_d \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \right\|^2 \\
= \frac{1}{m}\|\mathbf{X}w - \mathbf{y}\|^2 = \frac{1}{m}\left(\mathbf{w}^\top \mathbf{X}^\top \mathbf{X}\mathbf{w} - 2\mathbf{w}^\top \mathbf{X}^\top \mathbf{y} + \mathbf{y}^\top \mathbf{y}\right)
\end{cases}
$$

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

Linear regression

- Minimize $L_S(\mathbf{w}) \quad \Leftrightarrow \quad$ set gradient $\nabla_{\mathbf{w}} L_S(\mathbf{w})$ to 0

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

Linear regression

- Minimize $L_S(\mathbf{w}) \quad \Leftrightarrow \quad$ set gradient $\nabla_{\mathbf{w}} L_S(\mathbf{w})$ to 0

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

- Gradient of loss term $(\mathbf{w}^\top \mathbf{x}_i - y_i)^2$ w.r.t. scalar weight w_k:

$$\frac{\partial (\mathbf{w}^\top \mathbf{x}_i - y_i)^2}{\partial w_k} = 2(\mathbf{w}^\top \mathbf{x}_i - y_i) x_{i,k}$$

Linear regression

- Minimize $L_S(\mathbf{w}) \quad \Leftrightarrow \quad$ set gradient $\nabla_{\mathbf{w}} L_S(\mathbf{w})$ to 0

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

- Gradient of loss term $(\mathbf{w}^\top \mathbf{x}_i - y_i)^2$ w.r.t. scalar weight w_k:

$$\frac{\partial (\mathbf{w}^\top \mathbf{x}_i - y_i)^2}{\partial w_k} = 2(\mathbf{w}^\top \mathbf{x}_i - y_i) x_{i,k}$$

- Gradient

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

Linear regression

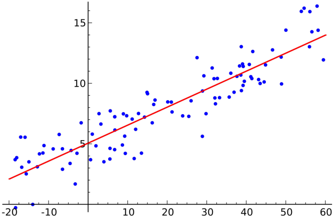
- $\nabla_{\mathbf{w}} L_S(\mathbf{w}) = 0 \quad \Leftrightarrow \quad \mathbf{X}^\top \mathbf{X} \mathbf{w} = \mathbf{X}^\top \mathbf{y}$
- $\mathbf{X}^\top \mathbf{X}$ is a $(d+1) \times (d+1)$ matrix

Linear regression

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

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

Hat matrix



- Approximate labels on inputs $\mathbf{x}_1, \ldots, \mathbf{x}_m$:

$$\hat{\mathbf{y}} = \mathbf{X}w_{\text{lin}} = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y} = \mathbf{H}\mathbf{y}$$

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

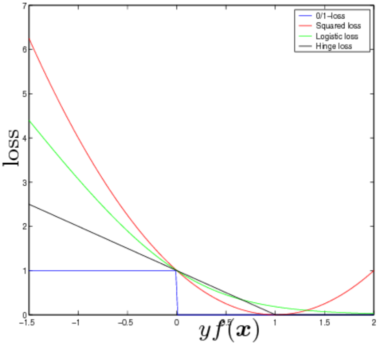
Is linear regression really "learning"?

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

Linear classification vs. linear regression

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

Relationship between loss functions



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

A second look at the Perceptron

What is the loss implicitly optimized by the PLA?

$$\mathbf{w}^{t+1} \leftarrow \begin{cases} \mathbf{w}^t + y_i \mathbf{x}_i & \text{if } \; y_i \mathbf{w}^t \mathbf{x}_i < 0 \\ \mathbf{w}^t & \text{otherwise} \end{cases}$$

$$\leftarrow \mathbf{w}^t - \nabla_{\mathbf{w}} \max(0, -y_i \mathbf{w}^t \mathbf{x}_i)$$

A second look at the Perceptron

What is the loss implicitly optimized by the PLA?

$$\mathbf{w}^{t+1} \leftarrow \begin{cases} \mathbf{w}^t + y_i \mathbf{x}_i & \text{if } \; y_i \mathbf{w}^t \mathbf{x}_i < 0 \\ \mathbf{w}^t & \text{otherwise} \end{cases}$$

$$\leftarrow \mathbf{w}^t - \nabla_{\mathbf{w}} \max(0, -y_i \mathbf{w}^t \mathbf{x}_i)$$

PLA follows the gradient of the local hinge loss

$$\ell_i(\mathbf{w}) = max(0, -y_i \mathbf{w} \mathbf{x}_i)$$

Content

1 Supervised learning

2 Perceptron

3 Linear regression

4 Logistic regression

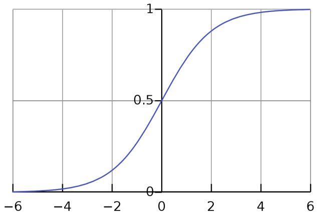
5 Non-linear transforms

6 Exercises

Logistic regression

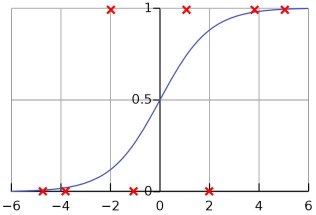
- Soft classification: estimate probability of belonging to class
- Hypothesis $h(\mathbf{x}) = \theta(\sum_{i=0}^{d} w_i x_i) = \theta(\mathbf{w}^\top \mathbf{x})$
- Logistic function

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



Logistic regression

- Assume that there are two classes $\{-1, +1\}$
- Ideally, data would be on the form $((\mathbf{x}_1, 0.8), \ldots, (\mathbf{x}_m, 0.1))$, i.e. the probability of belonging to class $+1$
- However, data is usually on the form $((\mathbf{x}_1, +1), \ldots, (\mathbf{x}_m, -1))$
- We can view labels as hard probabilities, i.e. 0 or 1



Logistic loss

- Find **w** that minimizes

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

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

Logistic loss

- Likelihood of a biased coin flip

$$p(y|\mathbf{x}, \mathbf{w}) = \theta(y\mathbf{w}^\top \mathbf{x})$$

Logistic loss

- Likelihood of a biased coin flip

$$p(y|\mathbf{x},\mathbf{w}) = \theta(y\mathbf{w}^\top\mathbf{x})$$

- Logistic loss as logarithm of likelihood

$$\begin{aligned}\log\ p(y|\mathbf{x},\mathbf{w}) &= \log\theta(y\mathbf{w}^\top\mathbf{x}) \\ &= \log\left(\frac{1}{1+e^{-y\mathbf{w}^\top\mathbf{x}}}\right) \\ &= -\log(1+e^{-y\mathbf{w}^\top\mathbf{x}})\end{aligned}$$

Logistic loss

- Likelihood of a biased coin flip

$$p(y|\mathbf{x}, \mathbf{w}) = \theta(y\mathbf{w}^\top \mathbf{x})$$

- Logistic loss as logarithm of likelihood

$$\begin{aligned} \log\ p(y|\mathbf{x}, \mathbf{w}) &= \log \theta(y\mathbf{w}^\top \mathbf{x}) \\ &= \log\left(\frac{1}{1 + e^{-y\mathbf{w}^\top \mathbf{x}}}\right) \\ &= -\log(1 + e^{-y\mathbf{w}^\top \mathbf{x}}) \end{aligned}$$

- Probabilistic interpretation of losses $\rightarrow$ Bayesian formalism

Logistic loss

- Minimize $L_S(\mathbf{w})$ $\Leftrightarrow$ Find w such that $\nabla_{\mathbf{w}} L_S(\mathbf{w}) = 0$
- Gradient

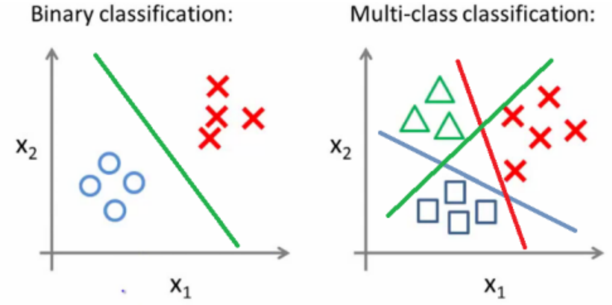
$$\nabla_{\mathbf{w}} L_S(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^{m} \theta(-y_i \mathbf{w}^\top \mathbf{x}_i)(-y_i \mathbf{x}_i)$$

- Unfortunately, no analytic solution to $\nabla_{\mathbf{w}} L_S(\mathbf{w}) = 0$ 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

1 Supervised learning

2 Perceptron

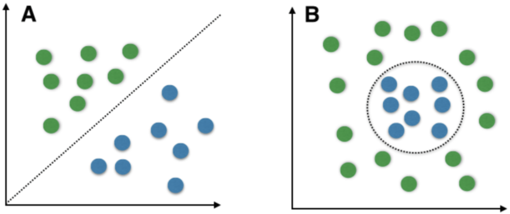
3 Linear regression

4 Logistic regression

5 Non-linear transforms

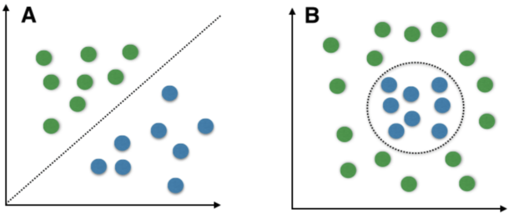
6 Exercises

Non-linear data



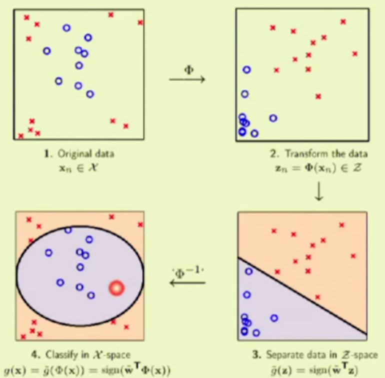
- Often data is not linearly separable at all
- Idea: derive circular perceptron, circular regression, etc.
- It would be better to take advantage of linear models!

Non-linear transforms



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

Non-linear transforms



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

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

1 Supervised learning

2 Perceptron

3 Linear regression

4 Logistic regression

5 Non-linear transforms

6 Exercises

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 $\binom{10}{2} = \frac{10 \cdot 9}{2} = 45$ classifiers in total
- Each classifier uses only data points of 2 classes, i.e. $2N/10$

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 $8N^3/1000$

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 $8N^3/1000$
- Hence the total running time is $45 \cdot 8N^3/1000 = \frac{9}{25}N^3$

Hat matrix

Show that the hat matrix $\mathbf{H} = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top$ has the following properties (where $\mathbf{I}$ is the identity matrix):

1. $\mathbf{H}$ is symmetric, i.e. $\mathbf{H}^\top = \mathbf{H}$
2. $\mathbf{H}^2 = \mathbf{H}$
3. $(\mathbf{I} - \mathbf{H})^2 = (\mathbf{I} - \mathbf{H})$

Hat matrix

Note that $(\mathbf{AB})^\top = \mathbf{B}^\top \mathbf{A}^\top$ and that $\left(\mathbf{A}^{-1}\right)^\top = \left(\mathbf{A}^\top\right)^{-1}$

$$\left\{ \mathbf{H}^\top = \left(\mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\right)^\top \right.$$

Hat matrix

Note that $(\mathbf{AB})^\top = \mathbf{B}^\top \mathbf{A}^\top$ and that $\left(\mathbf{A}^{-1}\right)^\top = \left(\mathbf{A}^\top\right)^{-1}$

$$\begin{cases} \mathbf{H}^\top &= \left(\mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\right)^\top \\ &= \mathbf{X}\left((\mathbf{X}^\top\mathbf{X})^{-1}\right)^\top \mathbf{X}^\top \\ & \\ & \end{cases}$$

Hat matrix

Note that $(\mathbf{AB})^\top = \mathbf{B}^\top\mathbf{A}^\top$ and that $\left(\mathbf{A}^{-1}\right)^\top = \left(\mathbf{A}^\top\right)^{-1}$

$$
\begin{cases}
\mathbf{H}^\top &= \left(\mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\right)^\top \\
&= \mathbf{X}\left((\mathbf{X}^\top\mathbf{X})^{-1}\right)^\top\mathbf{X}^\top \\
&= \mathbf{X}\left((\mathbf{X}^\top\mathbf{X})^\top\right)^{-1}\mathbf{X}^\top \\
\end{cases}
$$

Hat matrix

Note that $(\mathbf{AB})^\top = \mathbf{B}^\top\mathbf{A}^\top$ and that $\left(\mathbf{A}^{-1}\right)^\top = \left(\mathbf{A}^\top\right)^{-1}$

$$\begin{cases} \mathbf{H}^\top = \left(\mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\right)^\top \\ \quad\;\; = \mathbf{X}\left((\mathbf{X}^\top\mathbf{X})^{-1}\right)^\top\mathbf{X}^\top \\ \quad\;\; = \mathbf{X}\left((\mathbf{X}^\top\mathbf{X})^\top\right)^{-1}\mathbf{X}^\top \\ \quad\;\; = \mathbf{X}\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top = \mathbf{H} \end{cases}$$

Hat matrix

Note that $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$ and $\mathbf{AI} = \mathbf{IA} = \mathbf{A}$

$$\begin{cases} \mathbf{H}^2 = \mathbf{H} \cdot \mathbf{H} = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top \\ \\ \\ \end{cases}$$

Hat matrix

Note that $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$ and $\mathbf{A}\mathbf{I} = \mathbf{I}\mathbf{A} = \mathbf{A}$

$$\begin{cases} \mathbf{H}^2 = \mathbf{H} \cdot \mathbf{H} = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top \\ \quad\;\; = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}(\mathbf{X}^\top\mathbf{X})(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top \\ \\ \end{cases}$$

Hat matrix

Note that $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$ and $\mathbf{AI} = \mathbf{IA} = \mathbf{A}$

$$\begin{cases} \mathbf{H}^2 = \mathbf{H} \cdot \mathbf{H} = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top \\ \quad\;\; = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}(\mathbf{X}^\top\mathbf{X})(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top \\ \quad\;\; = \mathbf{XI}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top \end{cases}$$

Hat matrix

Note that $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$ and $\mathbf{A}\mathbf{I} = \mathbf{I}\mathbf{A} = \mathbf{A}$

$$\begin{cases} \mathbf{H}^2 = \mathbf{H} \cdot \mathbf{H} = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top \\ \quad\; = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}(\mathbf{X}^\top\mathbf{X})(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top \\ \quad\; = \mathbf{X}\mathbf{I}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top \\ \quad\; = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top = \mathbf{H} \end{cases}$$

Hat matrix

$$\begin{cases} (\mathbf{I} - \mathbf{H})^2 = \mathbf{I}^2 - 2\mathbf{I}\mathbf{H} + \mathbf{H}^2 \\ \end{cases}$$

Hat matrix

$$\begin{cases} (\mathbf{I}-\mathbf{H})^2 = \mathbf{I}^2 - 2\mathbf{I}\mathbf{H} + \mathbf{H}^2 \\ \qquad\qquad\; = \mathbf{I} - 2\mathbf{H} + \mathbf{H} = \mathbf{I} - \mathbf{H} \end{cases}$$

Perceptron learning algorithm (PLA)

Perceptron learning algorithm

1. Initialize weight vector $\mathbf{w}^0 = 0$
2. Find a mistake $(\mathbf{x}_i, y_i)$ such that $h(\mathbf{x}_i) \neq y_i$
3. Update weights as $w_1 \leftarrow \mathbf{w}^0 + y_i\mathbf{x}_i$
4. Repeat from 2. for weight vector $\mathbf{w}^t$, $t = 1, 2, \ldots$

Letting $R^2 = \max_i \|\mathbf{x}_i\|^2$, show that after t iterations, $\|\mathbf{w}^t\|^2 \leq tR^2$

Perceptron learning algorithm (PLA)

An update happens when prediction are wrong

$$\text{sign}(\mathbf{w}^\top \mathbf{x}_i) \neq y_i \equiv y_i \mathbf{w}^\top \mathbf{x}_i \leq 0$$

Perceptron learning algorithm (PLA)

An update happens when prediction are wrong

$$\mathrm{sign}(\mathbf{w}^\top \mathbf{x}_i) \neq y_i \equiv y_i \mathbf{w}^\top \mathbf{x}_i \leq 0$$

Consider a single step of PLA:

$$\begin{cases} \|\mathbf{w}^{t+1}\|^2 = \|\mathbf{w}^t + y_i \mathbf{x}_i\|^2 \\ \\ \\ \end{cases}$$

Perceptron learning algorithm (PLA)

An update happens when prediction are wrong

$$\text{sign}(\mathbf{w}^\top \mathbf{x}_i) \neq y_i \equiv y_i \mathbf{w}^\top \mathbf{x}_i \leq 0$$

Consider a single step of PLA:

$$\begin{cases} \|\mathbf{w}^{t+1}\|^2 = \|\mathbf{w}^t + y_i \mathbf{x}_i\|^2 \\ \qquad\qquad = \|\mathbf{w}^t\|^2 + 2 y_i \mathbf{w}^{t\top} \mathbf{x}_i + y_i^2 \|\mathbf{x}_i\|^2 \end{cases}$$

Perceptron learning algorithm (PLA)

An update happens when prediction are wrong

$$\mathrm{sign}(\mathbf{w}^\top \mathbf{x}_i) \neq y_i \equiv y_i \mathbf{w}^\top \mathbf{x}_i \leq 0$$

Consider a single step of PLA:

$$\begin{cases} \begin{aligned} \|\mathbf{w}^{t+1}\|^2 &= \|\mathbf{w}^t + y_i \mathbf{x}_i\|^2 \\ &= \|\mathbf{w}^t\|^2 + 2y_i \mathbf{w}^{t\top} \mathbf{x}_i + y_i^2 \|\mathbf{x}_i\|^2 \\ &\leq \|\mathbf{w}^t\|^2 + 0 + \|\mathbf{x}_i\|^2 \end{aligned} \end{cases}$$

Perceptron learning algorithm (PLA)

An update happens when prediction are wrong

$$\text{sign}(\mathbf{w}^\top \mathbf{x}_i) \neq y_i \equiv y_i \mathbf{w}^\top \mathbf{x}_i \leq 0$$

Consider a single step of PLA:

$$\begin{cases} \|\mathbf{w}^{t+1}\|^2 = \|\mathbf{w}^t + y_i \mathbf{x}_i\|^2 \\ \qquad\qquad = \|\mathbf{w}^t\|^2 + 2 y_i \mathbf{w}^{t\top} \mathbf{x}_i + y_i^2 \|\mathbf{x}_i\|^2 \\ \qquad\qquad \leq \|\mathbf{w}^t\|^2 + 0 + \|\mathbf{x}_i\|^2 \\ \qquad\qquad \leq \|\mathbf{w}^t\|^2 + R^2 \end{cases}$$

Perceptron learning algorithm (PLA)

An update happens when prediction are wrong

$$\mathrm{sign}(\mathbf{w}^\top \mathbf{x}_i) \neq y_i \equiv y_i \mathbf{w}^\top \mathbf{x}_i \leq 0$$

Consider a single step of PLA:

$$\begin{cases} \|\mathbf{w}^{t+1}\|^2 = \|\mathbf{w}^t + y_i \mathbf{x}_i\|^2 \\ \qquad\qquad = \|\mathbf{w}^t\|^2 + 2y_i \mathbf{w}^{t\top} \mathbf{x}_i + y_i^2 \|\mathbf{x}_i\|^2 \\ \qquad\qquad \leq \|\mathbf{w}^t\|^2 + 0 + \|\mathbf{x}_i\|^2 \\ \qquad\qquad \leq \|\mathbf{w}^t\|^2 + R^2 \end{cases}$$

Perceptron learning algorithm (PLA)

An update happens when prediction are wrong

$$\text{sign}(\mathbf{w}^\top \mathbf{x}_i) \neq y_i \equiv y_i \mathbf{w}^\top \mathbf{x}_i \leq 0$$

Consider a single step of PLA:

$$\begin{cases} \|\mathbf{w}^{t+1}\|^2 = \|\mathbf{w}^t + y_i \mathbf{x}_i\|^2 \\ \qquad\qquad = \|\mathbf{w}^t\|^2 + 2y_i \mathbf{w}^{t\top} \mathbf{x}_i + y_i^2 \|\mathbf{x}_i\|^2 \\ \qquad\qquad \leq \|\mathbf{w}^t\|^2 + 0 + \|\mathbf{x}_i\|^2 \\ \qquad\qquad \leq \|\mathbf{w}^t\|^2 + R^2 \end{cases}$$

Hence after t iterations, $\|\mathbf{w}^t\|^2 \leq \|\mathbf{w}^0\|^2 + tR^2 = 0 + tR^2 = tR^2$

Perceptron learning algorithm (PLA)

Data linearly separable $\Rightarrow$ exists w_* such that $y_i \mathbf{w}_*^\top \mathbf{x}_i > 0, \ \forall i \in [m]$

Perceptron learning algorithm

1. Initialize weight vector $\mathbf{w}^0 = 0$
2. Find a mistake $(\mathbf{x}_i, y_i)$ such that $h(\mathbf{x}_i) \neq y_i$
3. Update weights as $\mathbf{w}^1 \leftarrow \mathbf{w}^0 + y_i \mathbf{x}_i$
4. Repeat from 2. for weight vector $\mathbf{w}^t$, $t = 1, 2, \ldots$

Letting $\rho = \min_i y_i \frac{\mathbf{w}_*^\top}{\|\mathbf{w}_*\|} \mathbf{x}_i > 0$, show that after t iterations, $\frac{\mathbf{w}_*^\top \mathbf{w}^t}{\|\mathbf{w}_*\|} \geq t\rho$

Perceptron learning algorithm (PLA)

Consider a single step of PLA:

$$\begin{cases} \dfrac{\mathbf{w}_*^\top \mathbf{w}^{t+1}}{\|\mathbf{w}_*\|} = \dfrac{\mathbf{w}_*^\top w^t}{\|\mathbf{w}_*\|} + y_i \dfrac{\mathbf{w}_*^\top}{\|\mathbf{w}_*\|}\mathbf{x}_i \\ \end{cases}$$

Perceptron learning algorithm (PLA)

Consider a single step of PLA:

$$\begin{cases} \dfrac{\mathbf{w}_*^\top \mathbf{w}^{t+1}}{\|\mathbf{w}_*\|} = \dfrac{\mathbf{w}_*^\top w^t}{\|\mathbf{w}_*\|} + y_i \dfrac{\mathbf{w}_*^\top}{\|\mathbf{w}_*\|}\mathbf{x}_i \\ \qquad\qquad \geq \dfrac{\mathbf{w}_*^\top w^t}{\|\mathbf{w}_*\|} + \rho \end{cases}$$

Perceptron learning algorithm (PLA)

Consider a single step of PLA:

$$\begin{cases} \dfrac{\mathbf{w}_*^\top \mathbf{w}^{t+1}}{\|\mathbf{w}_*\|} = \dfrac{\mathbf{w}_*^\top w^t}{\|\mathbf{w}_*\|} + y_i \dfrac{\mathbf{w}_*^\top}{\|\mathbf{w}_*\|}\mathbf{x}_i \\ \qquad\qquad \geq \dfrac{\mathbf{w}_*^\top w^t}{\|\mathbf{w}_*\|} + \rho \end{cases}$$

Perceptron learning algorithm (PLA)

Consider a single step of PLA:

$$\begin{cases} \dfrac{\mathbf{w}_*^\top \mathbf{w}^{t+1}}{\|\mathbf{w}_*\|} = \dfrac{\mathbf{w}_*^\top w^t}{\|\mathbf{w}_*\|} + y_i \dfrac{\mathbf{w}_*^\top}{\|\mathbf{w}_*\|}\mathbf{x}_i \\ \qquad\qquad \geq \dfrac{\mathbf{w}_*^\top w^t}{\|\mathbf{w}_*\|} + \rho \end{cases}$$

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

Perceptron learning algorithm (PLA)

- Putting the two results together, we get

$$\frac{\mathbf{w}_*^\top w^t}{\|\mathbf{w}_*\|\|\mathbf{w}^t\|} \geq \frac{t\rho}{\sqrt{t}R} = \sqrt{t}\frac{\rho}{R}$$

Perceptron learning algorithm (PLA)

- Putting the two results together, we get

$$\frac{\mathbf{w}_*^\top w^t}{\|\mathbf{w}_*\|\|\mathbf{w}^t\|} \geq \frac{t\rho}{\sqrt{t}R} = \sqrt{t}\frac{\rho}{R}$$

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

$$1 \geq \frac{\mathbf{w}_*^\top \mathbf{w}^t}{\|\mathbf{w}_*\|\|\mathbf{w}^t\|} \geq \sqrt{t}\frac{\rho}{R}$$

Perceptron learning algorithm (PLA)

- Putting the two results together, we get

$$\frac{\mathbf{w}_*^\top w^t}{\|\mathbf{w}_*\|\|\mathbf{w}^t\|} \geq \frac{t\rho}{\sqrt{t}R} = \sqrt{t}\frac{\rho}{R}$$

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

$$1 \geq \frac{\mathbf{w}_*^\top \mathbf{w}^t}{\|\mathbf{w}_*\|\|\mathbf{w}^t\|} \geq \sqrt{t}\frac{\rho}{R}$$

- 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

$$\frac{\mathbf{w}_*^\top w^t}{\|\mathbf{w}_*\|\|\mathbf{w}^t\|} \geq \frac{t\rho}{\sqrt{t}R} = \sqrt{t}\frac{\rho}{R}$$

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

$$1 \geq \frac{\mathbf{w}_*^\top \mathbf{w}^t}{\|\mathbf{w}_*\|\|\mathbf{w}^t\|} \geq \sqrt{t}\frac{\rho}{R}$$

- 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!