Lecture II: Linear regression and classification. Loss functions

Marina Meilă
mmp@uwaterloo.ca

With Thanks to Pascal Poupart & Gautam Kamath
Cheriton School of Computer Science
University of Waterloo

January 12, 2026

Linear predictors generalities

Loss functions

Least squares linear regression

- Linear regresssion as minimizing L_{LS}
- Linear regresssion as maximizing likelihood

Linear Classifiers

- Linear Discriminant Analysis (LDA)
- OPTIONAL – QDA (Quadratic Discriminant Analysis)
- Logistic Regression
- OPTIONAL – The PERCEPTRON algorithm

Reading HTF Ch.: 2.1–5,2.9, 7.1–4 bias-variance tradeoff, Murphy Ch.: 1., 8.6[1], Bach Ch.:

[1] Neither textbook is close to these notes except in a few places; take them as alternative perspectives or related reading

Linear predictors

- Linear predictors for regression

$$f(x) = \beta^T x \tag{1}$$

  where $Y \in \mathbb{R}$, $X \in \mathbb{R}^d$ and $\beta \in \mathbb{R}^d$ is a **vector of parameters**.
  Hence, the model class is $\mathcal{F} = \{\beta \in \mathbb{R}^d\}$ the set of all linear functions over $\mathbb{R}^d$.
- Linear predictors for classification

$$\hat{y}(x) = \text{sgn}(\beta^T x) \tag{2}$$

  i.e. the decision boundary is linear

Transforming categorical inputs into real values

- if X_j takes two values (e.g "yes", "no"), map it to $\{\pm 1\}$ or $\{0, 1\} \subset \mathbb{R}$.
- discrete multivariate inputs
  - Let X_j take values in $\Omega_j = \{0, \ldots r-1\}$.
  - One defines the $r-1$ binary variables $\tilde{X}_{jk} = \mathbf{1}_{[]} X_j = k,\ k = 1 : r-1$. The variable X_j is replaced with $\tilde{X}_{jk},\ k = 1 : r-1$
  - the parameter β_j with $r-1$ parameters $\beta_{j1} \ldots \beta j, r-1$ representing the coefficients of $\tilde{X}_{j1}, \ldots \tilde{X}_{j,r-1}$.

  This substitution is widely used to parametrize any function of a discrete variable as a linear function

Example: The demographic variable Race takes values in { African, Asian, Caucasian, ...}; the corresponding parameters in the model will be $\hat{\beta}_{\text{Asian}}, \hat{\beta}_{\text{Caucasian}}, \ldots$.

The intercept as a slope

- Sometimes we like f to have an intercept $f(x) = \beta^T x + \beta_0$, with $x, \beta \in \mathbb{R}^d$. Such a function is affine, not linear, and not homogeneous. Here is a trick to get the best of both worlds.
- Add a dummy input $x_0 \equiv 1$ to x. Then its coefficient β_0 is the intercept.

$$\tilde{x} \leftarrow \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_d \end{bmatrix} \in \mathbb{R}^{d+1} \quad \tilde{\beta} \leftarrow \begin{bmatrix} \beta_0 \\ \beta_1 \\ \cdots \\ \beta_d \end{bmatrix} \in \mathbb{R}^{d+1} \quad f(x) = \tilde{\beta}^T \tilde{x} \tag{3}$$

- in classification, β_0 is called **threshold** or **bias term**

The linear predictor as classifier

The linear predictor can be used for [binary] classification with the sign trick from Lecture I.

$$f(x) = \text{sgn}\beta^T x \tag{4}$$

Later in the course we will see a natural way to use real-valued predictors for multi-way classification.
What is the meaning of the β parameter for (4)?
In the following lectures we will see three possible "interpretations" for β, which correspond three different ways to construct a linear classifier for a problem.

How good is a regressor? Measuring the "Error"

- Prediction error for y^i: $e^i = y^i - f(x^i)$
- "Error" of f on $\mathcal{D}$
  - $''Err'' = \frac{1}{n}\sum_{i=1}^{n} e^i$ **X**
  - $''Err'' = \frac{1}{n}\sum_{i=1}^{n} |e^i|$?
  - ... norms!
- Let $e = [e^1\ e^2\ \ldots\ e^n]$.
- e is a vector in $\mathbb{R}^n$. $\frac{1}{n}\sum_{i=1}^{n} |e^i| = \frac{1}{n}\|e\|_1$
- But we can use other norms, e.g. $\frac{1}{n}\|e\|_2$, $\frac{1}{n}\|e\|_\infty$.

How good is a regressor? Measuring the "Error"

- Prediction error for y^i: $e^i = y^i - f(x^i)$
- "Error" of f on $\mathcal{D}$
  - $''Err'' = \frac{1}{n}\sum_{i=1}^{n} e^i$ **X**
  - $''Err'' = \frac{1}{n}\sum_{i=1}^{n} |e^i|$?
  - ... norms!
- Let $e = [e^1\ e^2\ \ldots\ e^n]$.
- e is a vector in $\mathbb{R}^n$. $\frac{1}{n}\sum_{i=1}^{n} |e^i| = \frac{1}{n}\|e\|_1$
- But we can use other norms, e.g. $\frac{1}{n}\|e\|_2$, $\frac{1}{n}\|e\|_\infty$.

- Formally, $''Err''$ as above is called **loss** function.

Loss functions

The **loss function** represents the cost of error in a prediction problem. We denote it by L, where

$$L(y, \hat{y}) = \text{the cost of predicting } \hat{y} \text{ when the actual outcome is } y$$

As usually $\hat{y} = f(x)$ or $\mathrm{sgn} f(x)$, we will typically abuse notation and write $L(y, f(x))$.

Loss functions

The **loss function** represents the cost of error in a prediction problem. We denote it by L, where

$$L(y, \hat{y}) = \text{the cost of predicting } \hat{y} \text{ when the actual outcome is } y$$

As usually $\hat{y} = f(x)$ or $\text{sgn} f(x)$, we will typically abuse notation and write $L(y, f(x))$.

- For **Regression**
  - **Least-Squares** L_2 Loss $L_{LS}(y, f(x)) = \frac{1}{n}\|e\|_2^2$
  - L_1 Loss $L_{LS}(y, f(x)) = \frac{1}{n}\|e\|_1$
  - Statistical losses...
- For **Classification**
  - **Misclassification Error (0-1 Loss)** $L_{01} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{[y^i \neq \hat{y}^i]}$
  - Statistical losses...

Loss functions for classification

For classification, a natural loss function is the **misclassification error** (also called **0-1 loss**)

$$L_{01}(y, f(x)) = \mathbf{1}_{[y \neq f(x)]} = \begin{cases} 1 & \text{if } y \neq f(x) \\ 0 & \text{if } y = f(x) \end{cases} \tag{5}$$

Sometimes different errors have different costs. For instance, classifying a HIV+ patient as negative (**a false negative** error) incurs a much higher cost than classifying a normal patient as HIV+ (**false positive** error). This is expressed by **asymmetric misclassification costs**. For instance, assume that a false positive has cost one and a false negative has cost 100. We can express this in the matrix

$f(x):$	$+$	$-$
true $:+$	0	100
$-$	1	0

In general, when there are p classes, the matrix $L = [L_{kl}]$ defines the loss, with L_{kl} being the cost of misclassifying as l an example whose true class is k.

Training set loss and expected loss

- **Training set loss**
- Objective of prediction = to minimize loss on future data,

$$\text{minimize } L(f) = E_{P(X,Y)}[L(Y, f(X)] \text{ over } f \in \mathcal{F} \tag{6}$$

We call $L(f)$ above **expected loss**.

Example (Misclassification error $L_{01}(f)$)

$L_{01}(f)$ = probability of making an error on future data.

$$L_{01}(f) = P[Yf(X) < 0] = E_{P_{XY}}[\mathbf{1}_{[Yf(X)<0]}] \tag{7}$$

Training set loss and expected loss

- **Training set loss**
- Objective of prediction = to minimize loss on future data,

$$\text{minimize } L(f) = E_{P(X,Y)}[L(Y, f(X)] \text{ over } f \in \mathcal{F} \tag{6}$$

  We call $L(f)$ above **expected loss**.
- $L(f)$ cannot be minimized or even computed directly, because we don't know the data distribution P_{XY}.
- Therefore, in training we use the **traing set** loss.
- . . . we approximate data distribution P_{XY} by the sample $\mathcal{D}$.
- The empirical loss (or **empirical error** or **training error**) is the average loss on $\mathcal{D}$

$$\hat{L}(f) = \frac{1}{n} \sum_{i=1}^{n} 1_{[y^i f(x^i) < 0]} \tag{7}$$

Training set loss and expected loss

- **Training set loss**
- Objective of prediction = to minimize loss on future data,

$$\text{minimize } L(f) = E_{P(X,Y)}[L(Y, f(X)] \text{ over } f \in \mathcal{F} \tag{6}$$

  We call $L(f)$ above **expected loss**.
- Therefore, in training we use the **traing set** loss.
- . . . we approximate data distribution P_{XY} by the sample $\mathcal{D}$.
- The empirical loss (or **empirical error** or **training error**) is the average loss on $\mathcal{D}$

$$\hat{L}(f) = \frac{1}{n} \sum_{i=1}^{n} 1_{[y^i f(x^i) < 0]} \tag{7}$$

- And we approximate $L(f)$ the expected loss by a different data set $\mathcal{D}^{\text{test}}$ from the same P_{XY}.
- The size of $\mathcal{D}^{\text{test}}$ is n', not necessarily equal to n.

- Problem: how to lear a linear predictor from data?
- Now: examples of what one can do
- Later lectures: larger view of the estimation problem for predictors

(Linear) least squares regression

- define **data matrix** or (transpose) **design matrix**

$$\mathbf{X} = \begin{bmatrix} (x^1)^T \\ (x^2)^T \\ \cdots \\ (x^i)^T \\ \cdots \\ (x^n)^T \end{bmatrix} \in \mathbb{R}^{N \times n} \quad \text{and} \quad Y = \begin{bmatrix} y^1 \\ y^2 \\ \cdots \\ y^n \end{bmatrix}, \quad E = \begin{bmatrix} \varepsilon^1 \\ \varepsilon^2 \\ \cdots \\ \varepsilon^d \end{bmatrix} \in \mathbb{R}^d$$

- Then we can write

$$Y = X\beta + E$$

- The solution $\hat{\beta}$ is chosen to minimize the sum of the squared errors $\sum_{i=1}^{d}(\varepsilon^i)^2 = \sum_{i=1}^{d}(y^i - \beta^T x^i)^2 = ||E||^2$

$$\hat{\beta} = \underset{\beta \in \mathbb{R}^d}{\operatorname{argmin}} \sum_{i=1}^{d}(y^i - \beta^T x_i)^2$$

- This optimization problem is called a **least squares** problem. Its solution is

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T Y \tag{8}$$

- Underlying statistical model $y = \beta^T x + \varepsilon, \quad \varepsilon \sim N(0, \sigma^2)$ (and i.i.d. sampling of $(x^{1:N}, y^{1:N})$ of course).
  Then $\hat{\beta}$ from (8) is the Maximum Likelihood (ML) estimator of the parameter β.

The statistical view of Machine Learning: Likelihood

The statistical view of Machine Learning: Likelihood

- What is random? **the noise** $\epsilon^{1:n}$
- Express noise as function of $(x^{1:n}, y^{1:n})$

$$\epsilon^i = y^i - \beta_0 - \beta^T x^i \quad \sim N(0, \sigma^2) \tag{9}$$

- Likelihood

- Let $p_{0,\sigma^2}(\epsilon) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{\epsilon^2}{2\sigma^2}} = N(\epsilon; 0, \sigma^2)$
- Then

$$L(\beta_0, \beta_{1:d}, \sigma^2) = \prod_{i=1}^{n} p_{0,\sigma^2}(\epsilon^i) \tag{10}$$

$$= \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(\epsilon^i)^2}{2\sigma^2}} \tag{11}$$

$$= \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y^i - \beta_0 - \beta^T x^i)^2}{2\sigma^2}} \tag{12}$$

- **log-likelihood**

$$l(\beta_0, \beta_{1:d}, \sigma^2) = \tag{13}$$

$$= \sum_{i=1}^{n} \left\{ -\frac{1}{2} \ln \sigma^2 - \frac{1}{2} \ln(2\pi) - \frac{1}{2} \left(y^i - \beta_0 - \beta^T x^i \right)^2 \frac{1}{2\sigma^2} \right\} \tag{14}$$

$$= -\frac{n}{2} \ln \sigma^2 - \frac{1}{2} \ln(2\pi) + \text{constant} - \frac{1}{2} \sum_{i=1}^{n} \left(y^i - \beta_0 - \beta^T x^i \right)^2 \tag{15}$$

Maximizing the log-likelihood w.r.t β

- For simplicity, let $\beta_0 = 0$; hence $y^i = \beta^T x^i + \epsilon^i$
- log-likelihood

$$l(\beta_{1:d}, \sigma^2) = -\frac{n}{2} \ln \sigma^2 - \frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(y^i - \beta_0 - \beta^T x^i \right)^2 + \text{constant} \tag{16}$$

- For any σ^2,

$$\underset{\beta}{\operatorname{argmax}}\, l(\sigma^2, \beta) = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y^i - \beta_0 - \beta^T x^i \right)^2 \tag{17}$$

  a **Least Squares Problem**

- In matrix form $\min_\beta \|y - \mathbf{X}\beta\|^2$
- Solution

$$\beta^{\mathrm{ML}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T y \tag{18}$$

  with $(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \equiv \mathbf{X}^\dagger$ the pseudoinverse of $\mathbf{X}$
- β^{ML} is linear in y!

Linear Discriminant Analysis (LDA)

Fitting a linear predictor for classification, first approach. (We are in the binary classification case)

- Assume each class is generated by a Normal distribution

$$P_{X|Y}(x|+) = \mathcal{N}(x; \mu_+, \Sigma_+), \quad P_{X|Y}(x|-) = \mathcal{N}(x; \mu_-, \Sigma_-) \quad \text{and } P_Y(1) = p$$

- Given x, what is the probability it came from class $+$?

$$P_{Y|X}(+|x) = \frac{P_Y(1)P_{X|Y}(x|+)}{P_Y(1)P_{X|Y}(x|+) + P_Y(-)P_{X|Y}(x|+-)} \quad \text{and} \quad P_{Y|X}(-|x) = 1 - P_{Y|X}(+|x) \tag{19}$$

  This formula is true whether the distributions $P_{X|Y}$ are normal or not.
- We assign x to the class with maximum posterior probability.

$$\hat{y}(x) = \operatorname*{argmax}_{y \in \{\pm 1} P_{Y|X}(y|x) \tag{20}$$

  This too, holds true whether the distributions $P_{X|Y}$ are normal or not.

LDA – continued

Now we specialize to the case of normal class distribution. We assume in addition that $\Sigma_+ = \Sigma_- = K^{-1}$.

- Decision rule: $\hat{y} = 1$ iff $P_{Y|X}(+|x) > P_{Y|X}(-|x)$
- or equivalently iff

$$
\begin{aligned}
0 \quad \leq \quad & f(x) \;=\; \ln \frac{P_{Y|X}(+|x)}{P_{Y|X}(-|x)} && (21)\\
= \quad & \ln \frac{p}{1-p} - \frac{1}{2}\left[x^T K x - 2\mu_+^T K x + \mu_+^T K \mu_+\right] \\
& -\frac{1}{2}\left[x^T K x - 2\mu_-^T K x + \mu_-^T K \mu_-\right] && (22)\\
= \quad & [K(\mu_+ - \mu_-)]^T x + \ln \frac{p}{1-p} + \frac{\mu_-^T K \mu_- - \mu_+^T K \mu_+}{2} && (23)\\
= \quad & \beta^T x + \beta_0 && (24)
\end{aligned}
$$

- The above is a linear expression in x, hence this classifier is called **(Fisher's) Linear Discriminant**
- Note that if we change the variables to $x \leftarrow \sqrt{K}x$, $\mu_\pm \leftarrow \sqrt{K}\mu_\pm$, and if we shift the origin to $\frac{\mu_+ + \mu_-}{2}$ (24) becomes

$$2\mu_+^T x + \ln \frac{p}{1-p} \qquad (25)$$

This has a geometric interpretation

LDA Algorithm

LDA Algorithm

Train

1. Estimate μ_+ from data $\{(x^i, y^i),\ y^i = +1\}$
2. Estimate μ_- from data $\{(x^i, y^i),\ y^i = -1\}$
3. Estimate Σ jointly for both classes, calculate $K = \Sigma^{-1}$. Exercise Derive the formula for this estimate, in the Max Likelihood setting
4. Estimate $p = |\{(x^i, y^i),\ y^i = +1\}|/n$.

redict Now apply (24) to classify new data x

OPTIONAL – QDA (Quadratic Discriminant Analysis)

- If we do not assume $\Sigma_+ = \Sigma_-$ then (21) is a quadratic function of x Exercise Plot the curve $f(x) = 0$ in (21) for various data sets in two dimensions. What kind of curves do you observe? Can the decision region be bounded?

$$
\begin{aligned}
f(x) \;=& \\
&= \ln \frac{p}{1-p} - \tfrac{1}{2} \ln |\Sigma_+| + \tfrac{1}{2} \ln |\Sigma_-| && (26)\\
&- \frac{1}{2}\left[x^T \Sigma_+^{-1} x - 2\mu_+^T \Sigma_+^{-1} x + \mu_+^T \Sigma_+^{-1} \mu_+\right] && (27)\\
&+ \frac{1}{2}\left[x^T \Sigma_-^{-1} x - 2\mu_-^T \Sigma_-^{-1} x + \mu_-^T \Sigma_-^{-1} \mu_-\right] && (28)\\
&= \left[\ln \frac{p}{1-p} - \tfrac{1}{2} \ln |\Sigma_+| + \tfrac{1}{2} \ln |\Sigma_-| - \tfrac{1}{2}\mu_+^T \Sigma_+^{-1} \mu_+ + \tfrac{1}{2}\mu_-^T \Sigma_-^{-1} \mu_-\right] && (29)\\
&+ \underbrace{\left[\mu_+^T \Sigma_+^{-1} - \mu_-^T \Sigma_-^{-1}\right] x}_{\text{linear}} \underbrace{- \frac{1}{2} x^T \left[\Sigma_+^{-1} - \Sigma_-^{-1}\right] x}_{\text{quadratic}} && (30)
\end{aligned}
$$

Logistic Regression

Fitting a linear predictor for classification, another approach.

Let $\boxed{f(x) = \beta^T x}$ model the **log odds** of class 1

$$f(X) = \frac{P(Y=1|X)}{P(Y=-1|X)} \tag{31}$$

Then

- $\hat{y} = 1$ iff $P(Y=1|X) > P(Y=-1|X)$
  - just like in the previous case! so what's the difference?

Logistic Regression

Fitting a linear predictor for classification, another approach.

Let $\boxed{f(x) = \beta^T x}$ model the **log odds** of class 1

$$f(X) = \frac{P(Y=1|X)}{P(Y=-1|X)} \tag{31}$$

Then

- $\hat{y} = 1$ iff $P(Y=1|X) > P(Y=-1|X)$
  - just like in the previous case! so what's the difference?
  - Answer: We don't assume each class is Gaussian, so we are in a more general situation than LDA
- What is $p(x) = P(Y=1|X=x)$ under our linear model?

$$\ln \frac{p}{1-p} = f, \quad \frac{p}{1-p} = e^f, \quad p = \frac{e^f}{1+e^f} \quad 1-p = \frac{1}{1+e^f} \tag{32}$$

- Note that we can put the last two formulas together as

$$P[\,y\,|\,x\,] = \frac{1}{1+e^{-yf(x)}} = \phi(yf(x)) \tag{33}$$

- where $\phi(z) = \frac{1}{1+e^{-z}}$ is the **logistic** function

Estimating the parameters by Max Likelihood

- The likelihood of a data point is $P(y|x) = \phi(yf(x))$
- The log-likelihood for a single pair (x, y) is $l(\beta; (x, y)) = -\ln(1 + e^{-yf(x)})$
- Log-likelihood of the data set $\mathcal{D}$

$$l(\beta; \mathcal{D}) = \frac{1}{n} \sum_{i=1}^{n} l(\beta; (x^i, y^i)) = \frac{1}{n} \sum_{i=1}^{n} \ln(1 + e^{-y^i f(x^i)}) \tag{34}$$

- The optimal β maximizes $l(\beta; \mathcal{D})$. It is the value of β that sets the gradient $\nabla l(\beta; \mathcal{D}) = 0$.
- Unfortunately, this maximization problem does not have a closed form solution!
  We maximize the (log)likelihood by iterative methods (e.g. gradient ascent) to obtain the β of the classifier.

The gradient of $l(\beta; \mathcal{D})$

- For a single data point $l(\beta; (x, y)) = -\ln(1 + e^{-yf(x)}) = \phi(yf(x))$
- First calculate

$$\frac{\partial f}{\partial \beta} = \frac{\partial}{\partial \beta}(\beta^T x) = x \tag{35}$$

- Then, note $\phi' = \phi(1 - \phi)$,
- Putting them together

$$\frac{\partial l}{\partial f} = \frac{\partial \ln \phi(yf)}{\partial f} = \frac{y\phi(yf)(1 - \phi(yf)}{\phi(yf)} = y(1 - \phi(yf)) \tag{36}$$

- Finally

$$\frac{\partial l}{\partial \beta} = \frac{\partial l}{\partial f}\frac{\partial f}{\partial \beta} = (1 - \phi(yf))yx \tag{37}$$

- The gradient of the log-likelihood of the dataset $\mathcal{D}$ is now

$$\frac{\partial l(\beta; \mathcal{D})}{\partial \beta} = \frac{1}{n}\sum_{i=1}^{n}\left(1 - \underbrace{\phi(e^{y^i f(x^i)})}_{P(y_i|x^i,\beta)}\right) y_i x^i \tag{38}$$

The gradient of $l(\beta; \mathcal{D})$

- For a single data point $l(\beta; (x, y)) = -\ln(1 + e^{-yf(x)}) = \phi(yf(x))$
- First calculate

$$\frac{\partial f}{\partial \beta} = \frac{\partial}{\partial \beta}(\beta^T x) = x \tag{35}$$

- Then, note $\phi' = \phi(1 - \phi)$,
- Putting them together

$$\frac{\partial l}{\partial f} = \frac{\partial \ln \phi(yf)}{\partial f} = \frac{y\phi(yf)(1 - \phi(yf)}{\phi(yf)} = y(1 - \phi(yf)) \tag{36}$$

- Finally

$$\frac{\partial l}{\partial \beta} = \frac{\partial l}{\partial f}\frac{\partial f}{\partial \beta} = (1 - \phi(yf))yx \tag{37}$$

- The gradient of the log-likelihood of the dataset $\mathcal{D}$ is now

$$\frac{\partial l(\beta; \mathcal{D})}{\partial \beta} = \frac{1}{n}\sum_{i=1}^{n}\left(1 - \underbrace{\phi(e^{y^i f(x^i)})}_{P(y_i|x^i,\beta)}\right) y_i x^i \tag{38}$$

- Interpretation: The infinitezimal change of β to increase log-likelihood for a single data point is along the direction of x, with the sign of y

OPTIONAL – The Perceptron algorithm

Fitting a linear predictor for classification, third approach.
Define $f(x) = \beta^T x$ and find β that classifies all the data correctly (when possible).
Perceptron **Algorithm**

Input labeled training set $\mathcal{D}$

Initialize $\beta = 0$, for all i, $x^i \rightarrow \frac{x^i}{||x^i||}$ (normalize the inputs)

Repeat until no more mistakes
for $i = 1 : N$

1. if $\text{sgn}(\beta^T x^i) \neq y^i$ (a mistake)
$\beta \leftarrow \beta + y^i x^i$

(Other variants exist)

The perceptron algorithm and linearly separable data

- $\mathcal{D}$ is **linearly separable** iff there is a β_* so that $\text{sgn}\beta_*^T x^i = y^i$ for all $i = 1, \ldots N$.
  If one such β_* exists, then there are an infinity of them

Theorem

Let $\mathcal{D}$ be a linearly separable data set, and define

$$\gamma = \min_i \frac{|\beta_*^T x^i|}{||\beta_*|| ||x^i||}. \tag{39}$$

Then, the number of mistakes made by the PERCEPTRON *algorithm is at most $1/\gamma^2$.*

- Note that if we scale the examples to have norm 1, then γ is the minimum distance to the hyperplane $\beta_*^T x = 0$ in the data set.
  Exercise Show that if $\mathcal{D}$ is linearly separable, the scaling $x^i \to \frac{x^i}{||x^i||}$ leaves it linearly separable.
- If $\mathcal{D}$ is not linearly separable, the algorithm oscillates indefinitely.