Lectures 8

CS436/536: Introduction to Machine Learning

Zhaohan Xi
Binghamton University

zxi1@binghamton.edu

Recap

Write the equation for the cross-entropy error measure for logistic regression

when the hypothesis function is defined by a line $h(x) = \boldsymbol{w}^T\boldsymbol{x}$

and the error is measured on a dataset $\mathcal{D} = \{(x_1, y_1), \dots, (x_N, y_N)\}$

$$E_{in}(\boldsymbol{w}) =$$

Quiz 3, Problem 1

(2 point)

Preliminaries

Recall the chain rule for derivatives

$\frac{d}{dx}(\ln[f(x)]) =$ **Quiz 3, Problem 2** (1 points)

$\frac{d}{dx}\left(e^{f(x)}\right) =$ **Quiz 3, Problem 3** (1 points)

Recap Logistic Regression

$f(z) = \ln\left(1 + e^{-czb}\right)$, where c, b are constants

$f'(z) =$

Quiz 3, Problem 4
(3 points)

Recap Gradient Descent Weight Update Rule

$\boldsymbol{w}(t+1) =$

Quiz 3, Problem 5
(3 points)

Recap

Write the equation for the cross-entropy error measure for logistic regression

when the hypothesis function is defined by a line $h(x) = \boldsymbol{w}^T \boldsymbol{x}$

and the error is measured on a dataset $\mathcal{D} = \{(x_1, y_1), \dots, (x_N, y_N)\}$

$$E_{in}(\boldsymbol{w}) = \boxed{\frac{1}{N} \sum_{n=1}^{N} \ln\left(1 + e^{-y_n \boldsymbol{w}^T \boldsymbol{x}_n}\right)}$$

Preliminaries

Recall the chain rule for derivatives

$$\frac{d}{dx}(\ln[f(x)]) = \boxed{\frac{f'(x)}{f(x)}}$$

$$\frac{d}{dx}\left(e^{f(x)}\right) = \boxed{f'(x)e^{f(x)}}$$

Recap Logistic Regression

$f(z) = \ln\left(1 + e^{-czb}\right)$, where c, b are constants

$$f'(z) = \boxed{\begin{aligned} &\frac{d}{dz}\ln\left(1 + e^{-czb}\right) \\ = \quad &\frac{\frac{d}{dz}e^{-czb}}{1+e^{-czb}} \\ = \quad &\frac{-cb\ e^{-czb}}{1+e^{-czb}} \\ = \quad &-\frac{cb}{1+e^{czb}} \end{aligned}}$$

Recap Gradient Descent Weight Update Rule

$$\boldsymbol{w}(t+1) = \boxed{\boldsymbol{w}(t) + y_* \boldsymbol{x}_* \frac{\eta}{1 + e^{y_* \boldsymbol{w}^T \boldsymbol{x}_*}}}$$

Logistic Regression for Classification

$$g(x) \in [0, 1] = \widehat{Pr}(y = +1 \mid x)$$

Use a threshold to decide:

E.g. if $g(x) \geq 0.5$, output $+1$

otherwise, output -1

Classifier Evaluation Metrics: Confusion Matrix

- **Confusion Matrix:**

Actual class\Predicted class	C_1	$\neg C_1$	
C_1	**True Positives (TP)**	**False Negatives (FN)**	**P**
$\neg C_1$	**False Positives (FP)**	**True Negatives (TN)**	**N**

- In a confusion matrix with m classes, $CM_{i,j}$ indicates # of tuples in class i that were labeled by the classifier as class j
  - May have extra rows/columns to provide totals
- **Example of Confusion Matrix:**

Actual class\Predicted class	Bat first = yes	Bat first = no	Total
Bat first = yes	**6954**	**46**	7000
Bat first = no	**412**	**2588**	3000
Total	7366	2634	10000

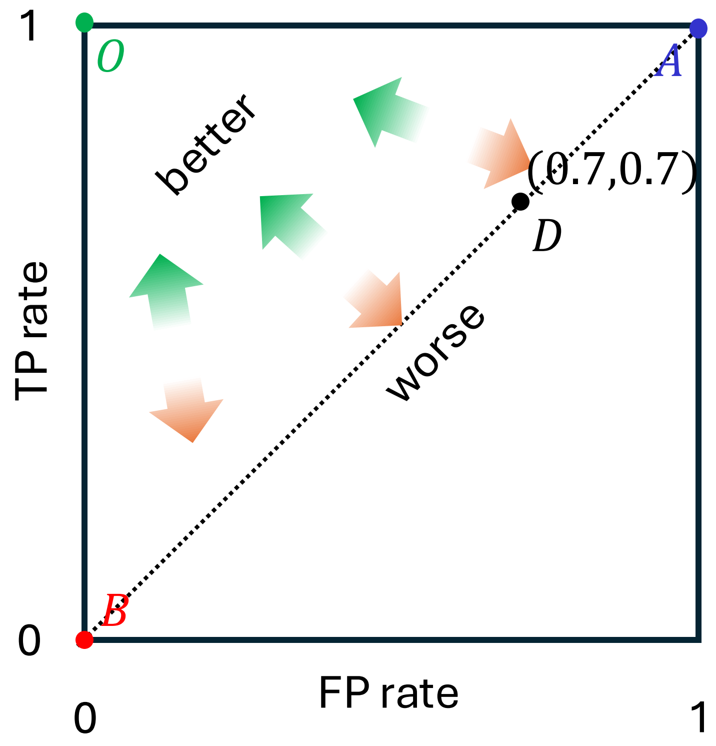
Tradeoffs between Specificity (TN/N) and Sensitivity (TP/P)

- Binary classifier A always outputs +1
- Binary classifier B always outputs -1
- Binary classifier O always predicts correctly

What are their:
TP rate and FP rate?
Sensitivity and Specificity?

- Binary classifier D guesses +1 w/ prob. 0.7

❑Vertical axis: True Positive rate (TP/P)
❑Horizontal axis: False Positive rate (FP/N)



Probabilistic Classifiers Use a Threshold

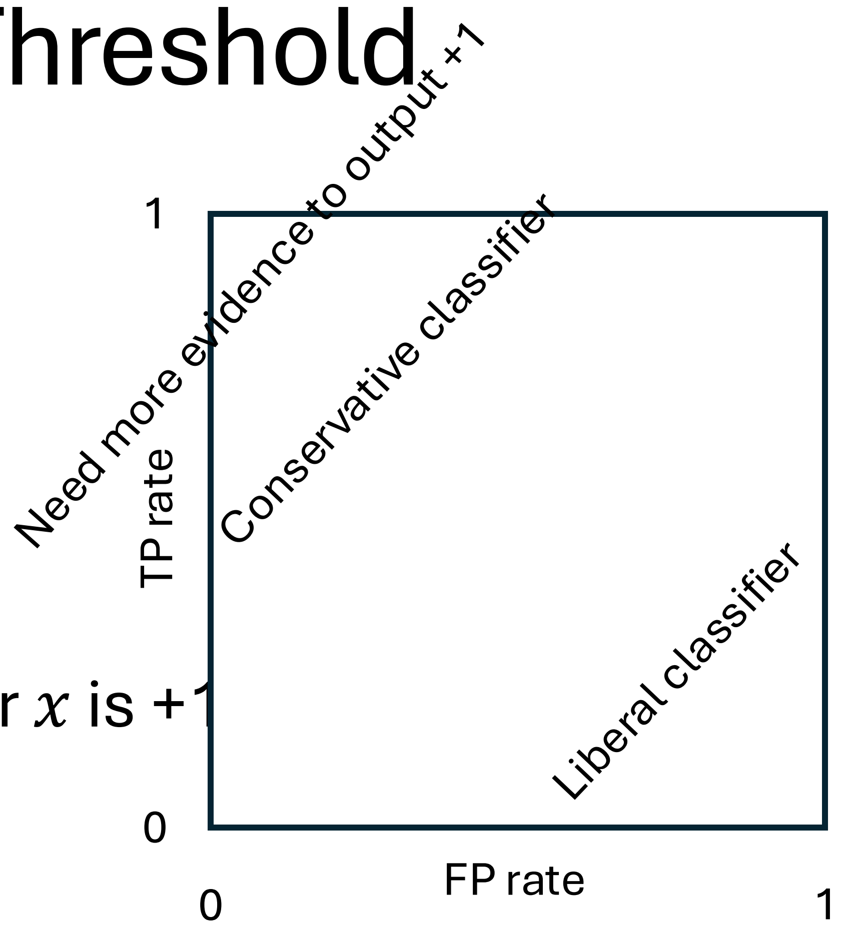
- Naïve Bayes
- Logistic Regression
- …

- Input: data example x
- Unknown label: y could be either +1 or -1
- Output: Predict probability $g(x)$ that the label for x is +1

- How to decide?

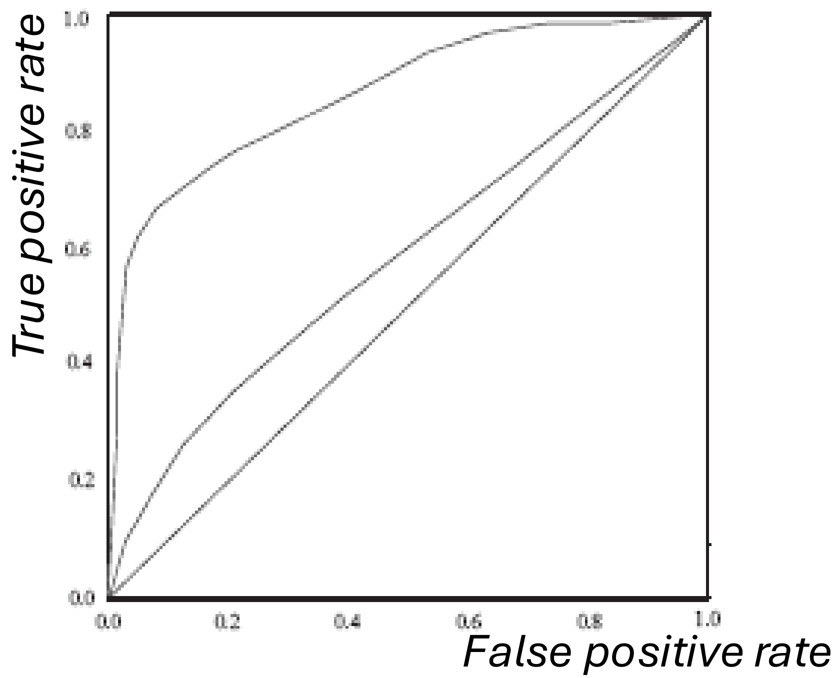
Use a threshold σ: If $g(x) \geq \sigma$, classify x as +1
Otherwise, classify x as -1

- How to set the threshold? How to determine if g is a good classifier?



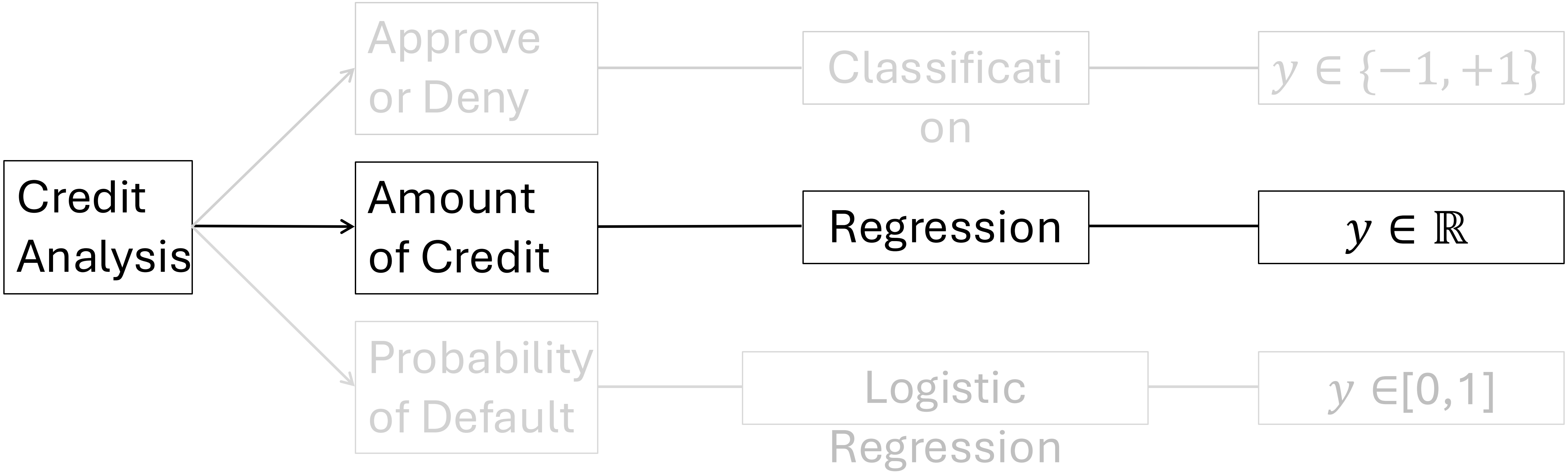
Model Selection: ROC Curves

- **ROC** (Receiver Operating Characteristics) curves: for visual comparison of classification models
- Originated from signal detection theory
- Shows the trade-off between the true positive rate and the false positive rate
- The area under the ROC curve (**AUC**: Area Under Curve) is a measure of the accuracy of the model
- Rank the test tuples in decreasing order: the one that is most likely to belong to the positive class appears at the top of the list
- The closer to the diagonal line (i.e., the closer the area is to 0.5), the less accurate is the model



- ❑ Vertical axis represents the true positive rate (TP/P)
- ❑ Horizontal axis rep. the false positive rate (FP/N)
- ❑ The plot also shows a diagonal line
- ❑ A model with perfect accuracy will have an area of 1.0

Linear Models for Three Learning Problems



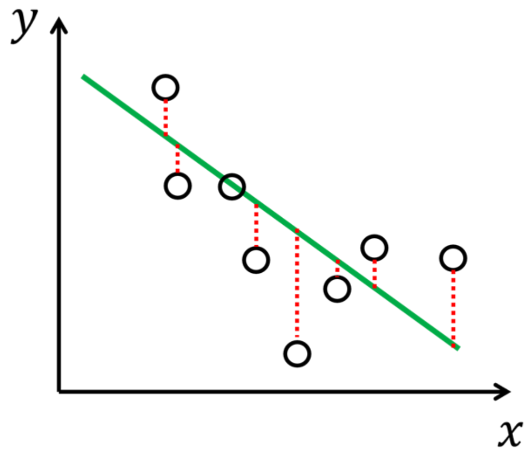
Linear Regression

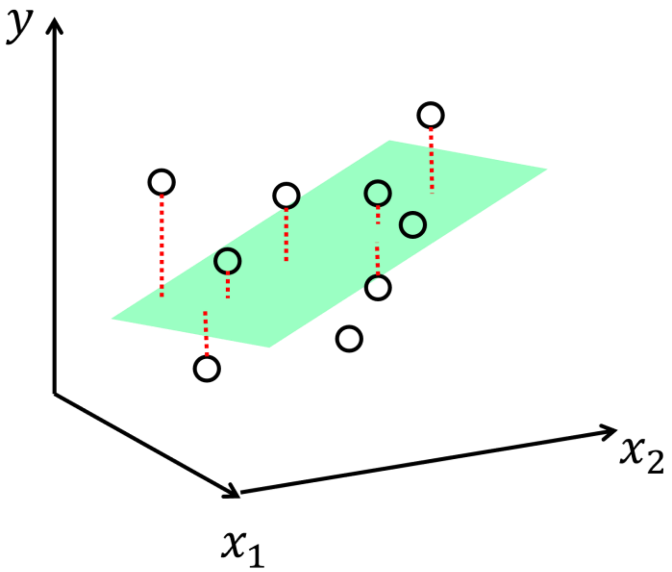
age	33 years
salary	50,000
debt	27,500
years employed	1
years at residence	2
…	…

Classification: Approve/Deny for credit?

Regression: Amount of credit? $y \in \mathbb{R}$

Least Squares Linear Regression





$$y = f(\boldsymbol{x}) + \epsilon \sim P(y|\boldsymbol{x})$$

$$\left.\begin{aligned} E_{in}(h) &= \frac{1}{N}\sum_{n=1}^{N}\left(h(\boldsymbol{x}_n) - f(\boldsymbol{x}_n)\right)^2 \\ E_{out}(h) &= \mathbb{E}_{\boldsymbol{x}}\left[\left(h(\boldsymbol{x}) - f(\boldsymbol{x})\right)^2\right] \end{aligned}\right\} h(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}$$

Least Squares Linear Regression

$$E_{in}(h) = \frac{1}{N}\sum_{n=1}^{N}\big(h(\boldsymbol{x}_n) - f(\boldsymbol{x}_n)\big)^2$$
$$= \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{w}^T\boldsymbol{x}_n - y_n)^2$$
$$= \frac{1}{N}\sum_{n=1}^{N}(\hat{y}_n - y_n)^2$$

Want: $\hat{y} \approx y$, where $y \in \mathbb{R}$

Towards a more compact representation

Data point

Weights

Prediction $\hat{y} = \boldsymbol{x}^T \boldsymbol{w}$ $[= \boldsymbol{w}^T \boldsymbol{x}]$

$$\boldsymbol{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \dots \\ x_d \end{bmatrix} \in 1 \times \mathbb{R}^d \text{ (i.e. } x_0 = 1\text{)}$$

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \dots \\ w_d \end{bmatrix} \in \mathbb{R}^{d+1}$$

$$\hat{y} = [x_0 x_1 x_2 \dots x_d] \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \dots \\ w_d \end{bmatrix} \in \mathbb{R}$$

$(d+1) \times 1$

$(d+1) \times 1$

Towards a more compact representation

<u>Dataset</u>

$$\boldsymbol{X} = \begin{bmatrix} -\boldsymbol{x}_1^T - \\ -\boldsymbol{x}_2^T - \\ \dots \\ -\boldsymbol{x}_N^T - \end{bmatrix}$$

$N \times (d+1)$

<u>Target</u>

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_N \end{bmatrix}$$

$N \times 1$

<u>Weights</u>

$$\boldsymbol{w} = \begin{bmatrix} w \\ w_2 \\ \dots \\ w_{d+1} \end{bmatrix}$$

$(d+1) \times 1$

<u>Predictions</u>(in-sample)

$$\widehat{\boldsymbol{y}} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \dots \\ \hat{y}_N \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_1^T\boldsymbol{w} \\ \boldsymbol{x}_2^T\boldsymbol{w} \\ \dots \\ \boldsymbol{x}_N^T\boldsymbol{w} \end{bmatrix} = \boldsymbol{X}\boldsymbol{w}$$

$N \times 1$

Using Matrices for Linear Regression

$$\boldsymbol{X} = \begin{bmatrix} -\boldsymbol{x}_1^T - \\ -\boldsymbol{x}_2^T - \\ \cdots \\ -\boldsymbol{x}_N^T - \end{bmatrix}$$

$N \times (d+1)$

data matrix

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_N \end{bmatrix}$$

$N \times 1$

target vector

$$\widehat{\boldsymbol{y}} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \cdots \\ \hat{y}_N \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_1^T \boldsymbol{w} \\ \boldsymbol{x}_2^T \boldsymbol{w} \\ \cdots \\ \boldsymbol{x}_N^T \boldsymbol{w} \end{bmatrix} = \boldsymbol{X}\boldsymbol{w}$$

$N \times 1$

in-sample predictions

$$\begin{aligned} E_{in}(\boldsymbol{w}) &= \frac{1}{N}\sum_{n=1}^{N}(\hat{y}_n - y_n)^2 \\ &= \frac{1}{N}\|\widehat{\boldsymbol{y}} - \boldsymbol{y}\|_2^2 \\ &= \frac{1}{N}\|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2 \\ &= \frac{1}{N}(\boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{y} + \boldsymbol{y}^T\boldsymbol{y}) \end{aligned}$$

$\|\cdot\|_2$ is the Euclidean norm

for any $k \times 1$ vector $\boldsymbol{z} = \begin{bmatrix} z_1 \\ \cdots \\ z_k \end{bmatrix}$,

$$\|\boldsymbol{z}\|_2 = \sqrt{z_1^2 + \cdots + z_k^2}$$

$$\|\boldsymbol{z}\|_2^2 = \boldsymbol{z}^T\boldsymbol{z}$$
$$(U - V)^T = U^T - V^T$$
$$(UV)^T = V^T U^T$$

Want: $\boldsymbol{w}_{lin} = \arg\min_{\boldsymbol{w}\in\mathbb{R}^{d+1}} E_{in}(\boldsymbol{w})$

Ordinary Least Squares: Minimizing E_{in}

$$E_{in}(\boldsymbol{w}) = \frac{1}{N}(\boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{y} + \boldsymbol{y}^T\boldsymbol{y})$$ [Differentiable]

Input data set $\boldsymbol{X}$, target values on input data set $\boldsymbol{y}$ are fixed

Intermediate Goal: $\boldsymbol{w}_{lin} = \arg\min_{\boldsymbol{w}\in\mathbb{R}^{d+1}} E_{in}(\boldsymbol{w})$

- Set derivative $\nabla E_{in}(\boldsymbol{w}) = 0$

  [using matrix calculus]

Similar to standard calculus
Want:
$z^* = \arg\min_{z} f(z) = az^2 + bz + c$
Set $\frac{df}{dz} = 2az + b = 0$
$\Rightarrow z^* = -\frac{b}{2a}$

Ordinary Least Squares: Minimizing E_{in}

$$E_{in}(\boldsymbol{w}) = \frac{1}{N}(\boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{y} + \boldsymbol{y}^T\boldsymbol{y})$$

Differentiable

Similar to standard calculus
Want:
$z^* = \arg\min_z f(z) = az^2 + bz + c$
Set $\frac{df}{dz} = 2az + b = 0$
$\Rightarrow z^* = -\frac{b}{2a}$

$\nabla E_{in}(\boldsymbol{w})$ is a $(d+1)$ – vector

$[\nabla E_{in}(\boldsymbol{w})]_i = \frac{\partial}{\partial w_i} E_{in}(\boldsymbol{w})$ is the i-th component

$$\nabla_{\boldsymbol{w}} E_{in}(\boldsymbol{w}) = \frac{2}{N}(\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^T\boldsymbol{y})$$
$$= 0$$

Useful gradient identities:
- $\nabla_z(\boldsymbol{z}^T\boldsymbol{A}\boldsymbol{z}) = (\boldsymbol{A} + \boldsymbol{A}^T)\boldsymbol{z}$
- $\nabla_z(\boldsymbol{z}^T\boldsymbol{b}) = \boldsymbol{b}$

Solve for $\boldsymbol{w}$: $\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{y}$

Ordinary Least Squares: Minimizing E_{in}

Ordinary Least Squares Algorithm: Solve for $\boldsymbol{w}$: $\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{y}$

- Input: $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \dots, (\boldsymbol{x}_n, y_n)$
- Construct matrix $\boldsymbol{X} = \begin{bmatrix} -\boldsymbol{x}_1^T - \\ -\boldsymbol{x}_2^T - \\ \dots \\ -\boldsymbol{x}_N^T - \end{bmatrix}, \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_N \end{bmatrix}$
- Compute pseudo-inverse $\boldsymbol{X}^\dagger$ of $\boldsymbol{X}$
    - If $\boldsymbol{X}^T\boldsymbol{X}$ is *invertible*, $\boldsymbol{X}^\dagger = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T$
- Return $\boldsymbol{w}_{lin} = \boldsymbol{X}^\dagger\boldsymbol{y}$

Analytical Solution!
$O(Nd^2 + d^3)$

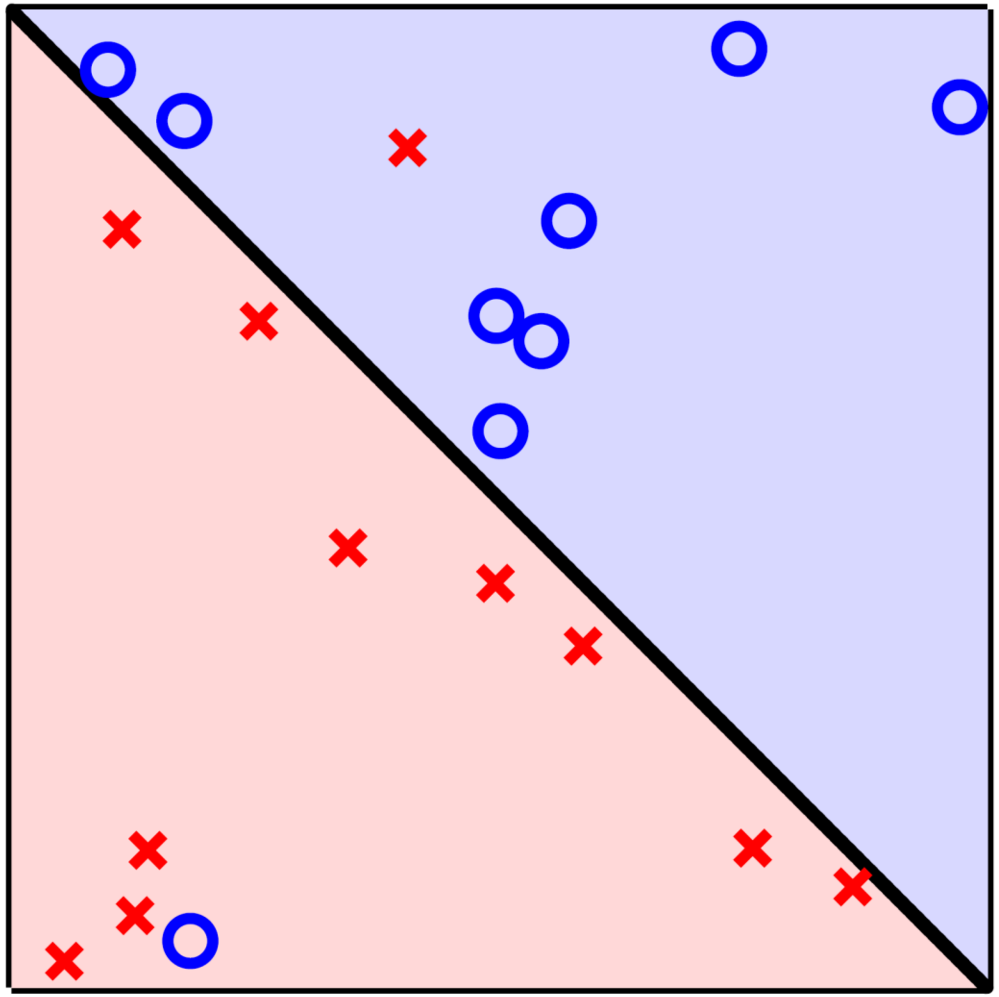
Ordinary Least Squares Linear Regression

- Input: $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \dots, (\boldsymbol{x}_n, y_n)$
- $E_{in}(\boldsymbol{w}) = \frac{1}{N}(\boldsymbol{w}^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{w} - 2\boldsymbol{w}^T \boldsymbol{X}^T \boldsymbol{y} + \boldsymbol{y}^T \boldsymbol{y})$
- Intermediate Goal: $\boldsymbol{w}_{lin} = \arg \min_{\boldsymbol{w} \in \mathbb{R}^{d+1}} E_{in}(\boldsymbol{w}) = \boldsymbol{X}^\dagger \boldsymbol{y} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y}$
- **Output:** $g(\boldsymbol{x}) = \boldsymbol{w}_{lin}^T \boldsymbol{x}$

- In sample predictions: $\widehat{\boldsymbol{y}} = \boldsymbol{X} \boldsymbol{w}_{lin} = \boldsymbol{X}(\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y}$;
  $E_{in}(\boldsymbol{g}) = E_{in}(\boldsymbol{w}_{lin}) = \|\widehat{\boldsymbol{y}} - \boldsymbol{y}\|_2^2$

$$E_{out}(g) = E_{in}(g) + O\left(\frac{d}{N}\right)$$

Linear Classification with Non-Separable Data



- A hard combinatorial optimization problem

$$\min_{\boldsymbol{w} \in \mathbb{R}^{d+1}} \frac{1}{N} \sum_{n=1}^{N} [\![sign(\boldsymbol{w}^T \boldsymbol{x}_n) \neq y_n]\!]$$

The Perceptron Learning Algorithm (PLA) (From Lecture 4)

A simple iterative algorithm

1. $\boldsymbol{w}(0) = \mathbf{0}$ — Start at some weights
2. **for** iteration $t = 1, 2, 3, \ldots$ **do**
3. the weight vector is $\boldsymbol{w}(t)$
4. **from** $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)$ pick *any* misclassified example
5. let (x_*, y_*) be the misclassified example — Observe a misclassification
$$sign(\boldsymbol{w}(t) \cdot \boldsymbol{x}_*) \neq y_*$$
6. update the weights
$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) + y_* \boldsymbol{x}_*$$
The update rule: Correct a misclassification
7. $t \leftarrow t + 1$

"incremental learning" one example at a time

Linear Regression for Linear Classification

- $y \in \{-1, +1\}$ Still a valid regression problem
- Output: $\boldsymbol{w}_{lin}$
- Very likely that $sign(\boldsymbol{w}_{lin}^T \boldsymbol{x}) \approx y$

- Use $\boldsymbol{w}_{lin}$ as starting point for PLA
  - $E_{in}(\mathrm{g})$ no worse than starting point!

Pretty pretty good (in practice)