CSE 5523: HW2+3

The Ohio State University

Outline

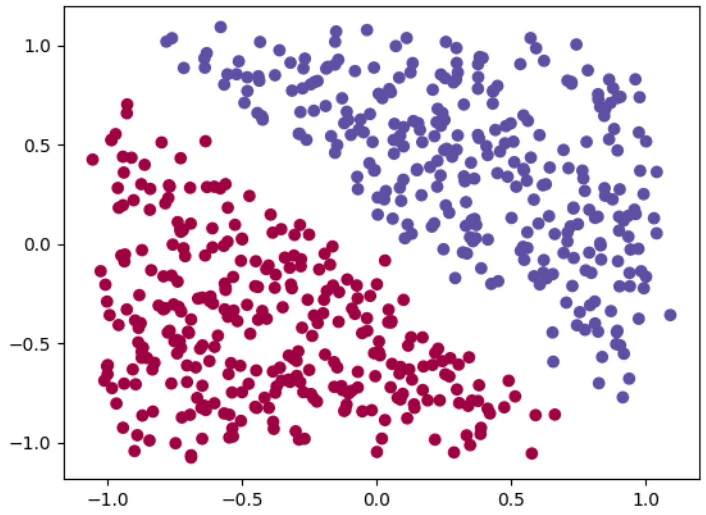
- You are to implement:
  - Linear logistic regression
  - Pocket algorithm (improved perceptron)
  - Linear soft-margin SVM
  - Linear Naïve Bayes
  - Linear Gaussian discriminative analysis
  - Nonlinear Naïve Bayes
  - Nonlinear Gaussian discriminative analysis

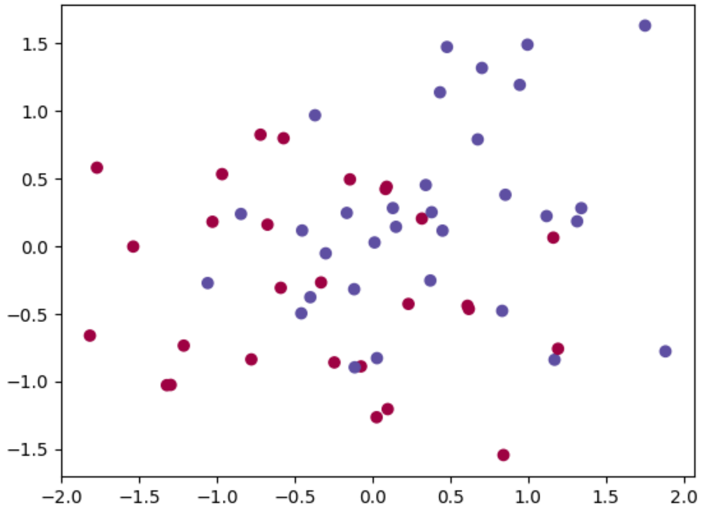
Data

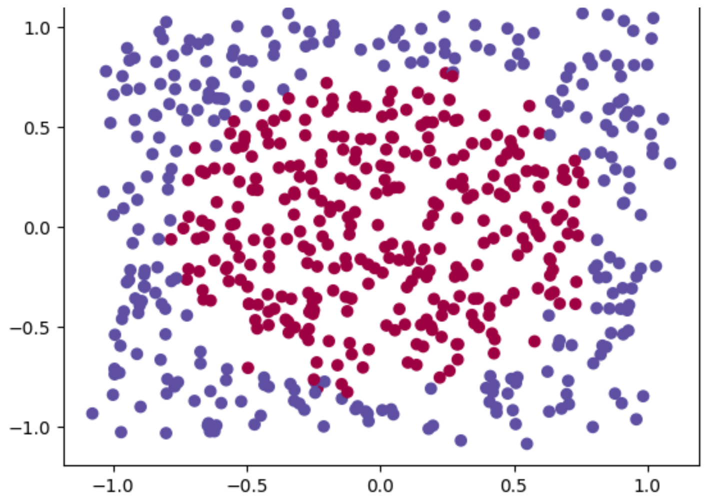
- Four data source
  - 2D linear
  - 2D noisy linear
  - 2D quadratic (circle)
  - MNIST (<5 vs. >=5)

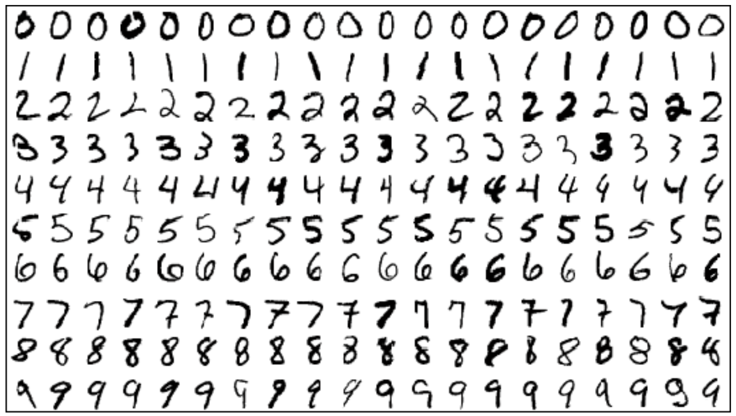
- $\boldsymbol{X} \in \mathbb{R}^{D \times N}$:
  - A column as an instance

- $\boldsymbol{Y} \in \{+1, -1\}^{N \times 1}$









Data

- The data $\boldsymbol{X}$ are not appended with “1” yet.
- For feature transform for a 2D data instance $\boldsymbol{x} \in \mathbb{R}^2$, we do
  - $\boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} x[1] \\ x[2] \\ x[1]^2 \\ x[2]^2 \\ x[1] \times x[2] \end{bmatrix}$
  - Again, you need to append “1” to the data $\boldsymbol{\phi}(\boldsymbol{x})$ if you want to solve $\widetilde{\boldsymbol{w}}$ directly
  - In the homework, we have done $\boldsymbol{\phi}(\boldsymbol{x})$ for you!

Accuracy

- Data = $\{ (\boldsymbol{x}_i,\ y_i \in \{+1, -1\}) \}_{i=1}^{N}$
- Accuracy = $\frac{1}{N} \sum_{i=1}^{N} \mathbf{1}[\hat{y}_i == y_i]$, where $\hat{y}_i$ is the prediction based on $\boldsymbol{x}_i$

Logistic regression

Logistic regression

- **Training data:** $D_{tr} = \{\, (\boldsymbol{x}_i \in \mathbb{R}^D,\ \mathrm{y}_i \in \{+1,-1\})\,\}_{i=1}^{N}$
- **Model:** $\mathrm{sign}(\boldsymbol{w}\boldsymbol{x} + b) = \mathrm{sign}(\widetilde{\boldsymbol{w}}^T \widetilde{\boldsymbol{x}})$
- **Objective:** $E(\widetilde{\boldsymbol{w}}) = \frac{1}{N}\sum_{i=1}^{N} \log\left(1 + e^{-y_i \widetilde{\boldsymbol{w}}^T \widetilde{\boldsymbol{x}}_i}\right)$
  - Please add a $\frac{1}{N}$ for normalization

Gradient descent (GD) for logistic regression

- Initialize $\tilde{\boldsymbol{w}}$
- For $t = 1{:}T$
  - $\tilde{\boldsymbol{w}} \leftarrow \tilde{\boldsymbol{w}} - \eta \nabla_{\tilde{\boldsymbol{w}}} E(\tilde{\boldsymbol{w}})$; $E(\tilde{\boldsymbol{w}})$: the loss function you want to minimize!
  - No need to stop earlier

- Note:
  - If $E(\tilde{\boldsymbol{w}}) = \frac{1}{N}\sum_n e_n(\tilde{\boldsymbol{w}})$; for example, $e_n(\tilde{\boldsymbol{w}})$ is the loss on the *i*-th example
  - $\nabla_{\tilde{\boldsymbol{w}}} E(\tilde{\boldsymbol{w}}) = \frac{1}{N}\sum_n \nabla_{\tilde{\boldsymbol{w}}} e_n(\tilde{\boldsymbol{w}})$

Gradient descent (GD) for logistic regression

- $e_n(\tilde{\boldsymbol{w}}) = \log\left(1 + e^{-y_n \tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{x}}_n}\right)$ for $y_i \in \{+1, -1\}$

$$\nabla_{\tilde{\boldsymbol{w}}} e_n(\tilde{\boldsymbol{w}}) = \frac{1}{1 + e^{-y_n \tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{x}}_n}} \times \nabla_{\tilde{\boldsymbol{w}}} \left(1 + e^{-y_n \tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{x}}_n}\right)$$

$$= \frac{1}{1 + e^{-y_n \tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{x}}_n}} \times (-y_n \tilde{\boldsymbol{x}}_n) \times e^{-y_n \tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{x}}_n}$$

$$= \rho\left(y_n \tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{x}}\right) \times (-y_n \tilde{\boldsymbol{x}}_n) \times e^{-y_n \tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{x}}_n}$$

$$= \rho\left(y_n \tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{x}}\right) \times (-y_n \tilde{\boldsymbol{x}}_n) \times \frac{1 - \rho\left(y_n \tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{x}}\right)}{\rho\left(y_n \tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{x}}\right)}$$

$$= -y_n \left(1 - \rho\left(y_n \tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{x}}\right)\right) (\tilde{\boldsymbol{x}}_n)$$

Gradient descent (GD) for logistic regression

- $e_n(\widetilde{\boldsymbol{w}}) = -\left[y_n \log p(+1|\boldsymbol{x}_n; \boldsymbol{\theta}) + (1 - y_n) \log\left(1 - p(+1|\boldsymbol{x}_n; \boldsymbol{\theta})\right)\right]$ for $y_i \in \{+1, 0\}$

$$\nabla_{\widetilde{\boldsymbol{w}}} e_n(\widetilde{\boldsymbol{w}}) = -\left[y_n \nabla_{\widetilde{\boldsymbol{w}}} \log p(+1|\boldsymbol{x}_n; \boldsymbol{\theta}) + (1 - y_n) \nabla_{\widetilde{\boldsymbol{w}}} \log\left(1 - p(+1|\boldsymbol{x}_n; \boldsymbol{\theta})\right)\right]$$
$$= -\left[y_n \nabla_{\widetilde{\boldsymbol{w}}} \log \rho\left(\widetilde{\boldsymbol{w}}^T \widetilde{\boldsymbol{x}}\right) + (1 - y_n) \nabla_{\widetilde{\boldsymbol{w}}} \log\left(1 - \rho\left(\widetilde{\boldsymbol{w}}^T \widetilde{\boldsymbol{x}}\right)\right)\right]$$
$$= -\left[y_n \times \left(1 - \rho\left(\boldsymbol{w}^T \boldsymbol{x}_n\right)\right) \boldsymbol{x}_n - (1 - y_n) \times \rho\left(\boldsymbol{w}^T \boldsymbol{x}_n\right) \boldsymbol{x}_n\right]$$
$$= -\left(y_n - \rho\left(\boldsymbol{w}^T \boldsymbol{x}_n\right)\right) \boldsymbol{x}_n$$

Pocket algorithm

Pocket algorithm

- **Training data:** $D_{tr} = \left\{ \left(\boldsymbol{x}_i \in \mathbb{R}^D,\ \mathrm{y}_i \in \{+1, -1\} \right) \right\}_{i=1}^{N}$

- **Model:** $\mathrm{sign}(\boldsymbol{w}\boldsymbol{x} + b) = \mathrm{sign}\left(\widetilde{\boldsymbol{w}}^T \widetilde{\boldsymbol{x}}\right)$

Pocket algorithm

- Initialize $\tilde{\boldsymbol{w}}$ and $\tilde{\boldsymbol{w}}^{\text{best}}$
- For $t = 1{:}\,T$
    - Loop for all training examples $\tilde{\boldsymbol{x}}_{\boldsymbol{n}}$ (random order!)
        - Predict $\hat{y}_n = \text{sign}\left(\tilde{\boldsymbol{w}}^T \tilde{\boldsymbol{x}}_{\boldsymbol{n}}\right)$
        - If $\hat{y}_n \neq y_n$
            - Update: $\tilde{\boldsymbol{w}} \leftarrow \tilde{\boldsymbol{w}} + \eta(y_n \tilde{\boldsymbol{x}}_{\boldsymbol{n}})$
    - Evaluate $\tilde{\boldsymbol{w}}$ on the “training data” and calculate the training accuracy
        - If training accuracy by $\tilde{\boldsymbol{w}}$ is “higher” than the training accuracy by $\tilde{\boldsymbol{w}}^{\text{best}}$
        - $\tilde{\boldsymbol{w}}^{\text{best}} \leftarrow \tilde{\boldsymbol{w}}$
- Output $\tilde{\boldsymbol{w}}^{\text{best}}$

Soft-margin SVM algorithm

Soft-margin SVM

- **Training data:** $D_{tr} = \{ (\boldsymbol{x}_i \in \mathbb{R}^D,\ \mathrm{y}_i \in \{+1, -1\}) \}_{i=1}^{N}$
- **Model:** $\mathrm{sign}(\boldsymbol{w}\boldsymbol{x} + b)$
- **Objective:** $E(\boldsymbol{w}, b) = \frac{1}{N} \sum_n \max\{1 - \mathrm{y}_n(\boldsymbol{w}^T \boldsymbol{x}_n + b), 0\} + \frac{1}{2} \lambda \boldsymbol{w}^T \boldsymbol{w}$
  - Please add a $\frac{1}{N}$ for normalization
  - λ: regularization coefficients (i.e., reg_coeff)

Gradient descent (GD) for soft-margin SVM

- Initialize $\tilde{\boldsymbol{w}}$
- For $t = 1{:}T$
  - $\tilde{\boldsymbol{w}} \leftarrow \tilde{\boldsymbol{w}} - \eta \nabla_{\tilde{\boldsymbol{w}}} E(\tilde{\boldsymbol{w}})$; $E(\tilde{\boldsymbol{w}})$: the loss function you want to minimize!
  - No need to stop earlier

- Note:
  - If $E(\boldsymbol{w}, b) = \frac{1}{N} \sum_n e_n(\boldsymbol{w}, b) + \frac{1}{2} \lambda \boldsymbol{w}^T \boldsymbol{w}$
  - $\nabla_{\boldsymbol{w}} E(\boldsymbol{w}, b) = \frac{1}{N} \sum_n \nabla_{\boldsymbol{w}} e_n(\boldsymbol{w}, b) + \lambda \boldsymbol{w}$
  - $\nabla_b E(\boldsymbol{w}, b) = \frac{1}{N} \sum_n \nabla_b e_n(\boldsymbol{w}, b)$

Gradient descent (GD) for soft-margin SVM

- $\nabla_{\boldsymbol{w}} e_n(\boldsymbol{w}, b) = \begin{cases} -y_n \boldsymbol{x}_n, & \text{if } \mathrm{y}_n\left(\boldsymbol{w}^T \boldsymbol{x}_n + b\right) < 1 \\ 0, & \text{othewise} \end{cases}$
- $\nabla_b e_n(\boldsymbol{w}, b) = \begin{cases} -y_n, & \text{if } \mathrm{y}_n\left(\boldsymbol{w}^T \boldsymbol{x}_n + b\right) < 1 \\ 0, & \text{othewise} \end{cases}$

Naïve Bayes

Naïve Bayes

- **Training data:** $D_{tr} = \{ (\boldsymbol{x}_i \in \mathbb{R}^D,\ \mathrm{y}_i \in \{+1, -1\}) \}_{i=1}^{N}$

- **Goal:** construct $p(Y = c|\boldsymbol{x})$ for $\hat{y} = \max\limits_{c \in \{+1,-1\}} p(Y = c|\boldsymbol{x})$

- **Bayes' rules:** $p(Y = c|\boldsymbol{x}) \propto p(\boldsymbol{x}|Y = c)p(Y = c)$
  - $p(Y = c)$: Bernoulli
  - $\mathrm{p}(\boldsymbol{x}|Y = c) = \mathrm{p}(x[1], x[2], x[3], \dots, x[D] \mid Y = c)$

    $$= p(x[1] \mid Y = c)p(x[2] \mid Y = c) \dots \dots p(x[D] \mid Y = c) = \prod_{d=1}^{D} p(x[d] \mid Y = c)$$
  - $p(x[d] \mid Y = c)$: one-dimensional Gaussian

Nonlinear Naïve Bayes

- $p(x[d] \mid Y = +1)$ and $p(x[d] \mid Y = +1)$ have their own standard deviations $\sigma_{d,+1}$ and $\sigma_{d,-1}$
- See slides 11 or 12 for how to compute them
- You can represent $[\sigma_{1,+1}, \sigma_{2,+1}, \dots \dots, \sigma_{N,+1}]^T$ as a vector

Linear Naïve Bayes

- $p(x[d] \mid Y = +1)$ and $p(x[d] \mid Y = +1)$ share the same standard deviation σ_d
- Built upon the previous slide, given $\sigma_{d,+1}$, $\sigma_{d,-1}$ and let N_{+1}, N_{-1} be the number of training examples per class, $\sigma_d^2 = \frac{N_{+1} \times \sigma_{d,+1}^2 + N_{-1} \times \sigma_{d,-1}^2}{N}$
- You can represent $[\sigma_1, \sigma_2, \dots \dots, \sigma_N]^T$ as a vector

Prediction (please do "log" to prevent overflow)

$$\max_{c\in\{+1,-1\}} p(Y=c|\boldsymbol{x}) = \max_{c\in\{+1,-1\}} p(\boldsymbol{x}|Y=c)p(Y=c)$$

$$= \max_{c\in\{+1,-1\}} p(Y=c)\prod_{d=1}^{D} p(x[d] \mid Y=c)$$

$$= \max_{c\in\{+1,-1\}} \log p(Y=c) + \sum_{d=1}^{D} \log p(x[d] \mid Y=c)$$

Gaussian discriminant analysis

GDA

- **Training data:** $D_{tr} = \left\{ \left(\boldsymbol{x}_i \in \mathbb{R}^D,\ \mathrm{y}_i \in \{+1, -1\} \right) \right\}_{i=1}^{N}$
- **Goal:** construct $p(Y = c|\boldsymbol{x})$ for $\hat{y} = \max_{c \in \{+1,-1\}} p(Y = c|\boldsymbol{x})$
- **Bayes' rules:** $p(Y = c|\boldsymbol{x}) \propto p(\boldsymbol{x}|Y = c)p(Y = c)$
  - $p(Y = c)$: Bernoulli
  - $\mathrm{p}(\boldsymbol{x}|Y = c)$: multi-dimensional Gaussian

Nonlinear GDA

- $p(\boldsymbol{x}|\, Y = +1)$ and $p(\boldsymbol{x}|\, Y = +1)$ have their own covariance matrices $\boldsymbol{\Sigma}_{+1}, \boldsymbol{\Sigma}_{-1}$
- See slides 10, 11 for how to compute them
- See also your homework # 2

Linear GDA

- $p(\boldsymbol{x}|\ Y = +1)$ and $p(\boldsymbol{x}|\ Y = +1)$ share the same covariance matrix $\boldsymbol{\Sigma}$

- Built upon the previous slide, given $\boldsymbol{\Sigma}_{+1}, \boldsymbol{\Sigma}_{-1}$ and let N_{+1}, N_{-1} be the number of training examples per class, $\boldsymbol{\Sigma} = \frac{N_{+1} \times \boldsymbol{\Sigma}_{+1} + N_{-1} \times \boldsymbol{\Sigma}_{-1}}{N}$

- See your homework # 2 for how to compute it

Prediction (please do “log” to prevent overflow)

$$\max_{c\in\{+1,-1\}} p(Y=c|\boldsymbol{x}) = \max_{c\in\{+1,-1\}} p(\boldsymbol{x}|Y=c)p(Y=c)$$

$$= \max_{c\in\{+1,-1\}} \log p(\boldsymbol{x}|Y=c) + \log p(Y=c)$$