CS3244 : Machine Learning

Semester 2 2023/24

Lecture 4 : Linear Models and Support Vector Machine

Xavier Bresson

https://twitter.com/xbresson

Department of Computer Science

National University of Singapore (NUS)

Outline

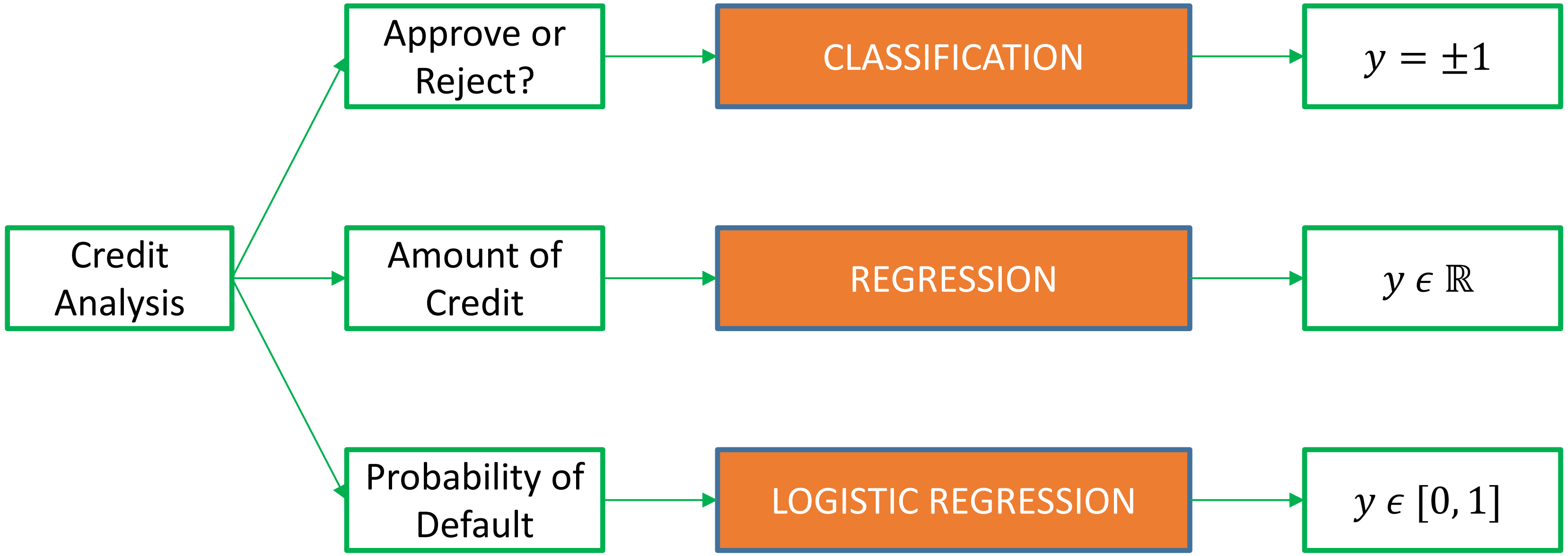
- Admin
- Three applications of linear models
- Classification
- Regression
- Normal equations
- Logistic regression
- Gradient descent
- Support Vector Machine
- Soft-margin SVM
- Kernel SVM
- Conclusion

Outline

- Admin
- Three applications of linear models
- Classification
- Regression
- Normal equations
- Logistic regression
- Gradient descent
- Support Vector Machine
- Soft-margin SVM
- Kernel SVM
- Conclusion

Three learning problems

- What are the simplest models to solve these learning tasks?

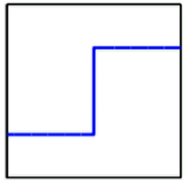


Three learning problems

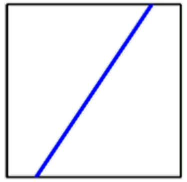
- What are the simplest models to solve these learning tasks?
  - Linear models – baseline to any machine learning project.

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x}$$

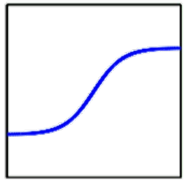
The "mother" linear function



$\mathrm{sign}(\boldsymbol{\theta}^\top \mathbf{x})$ $y = \pm 1$ Linear classification



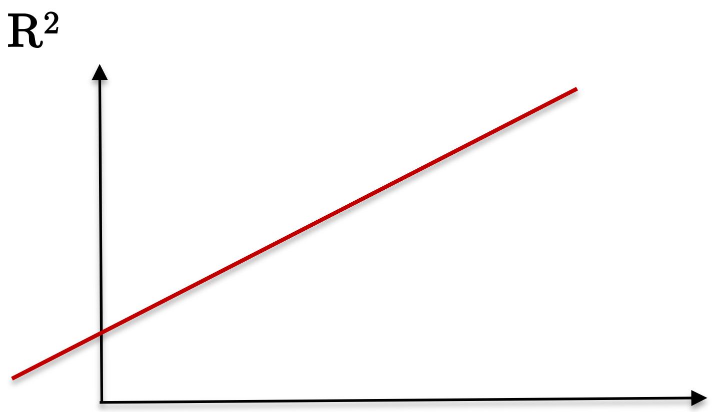
$\boldsymbol{\theta}^\top \mathbf{x}$ $y \in \mathbb{R}$ Linear regression

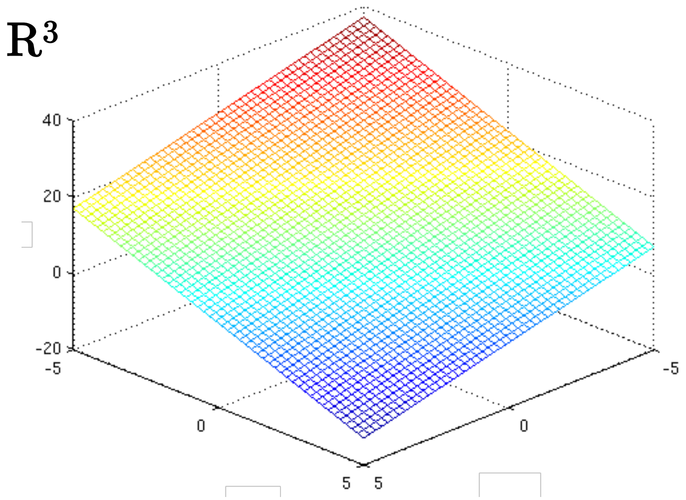


$\mathrm{sigmoid}(\boldsymbol{\theta}^\top \mathbf{x})$ $y \in [0, 1]$ Logistic regression

Linear models

- Hypothesis space H, i.e. the space of all solutions, of linear models :
    - Straight lines (2D space), planes (3D space) and hyper-planes (>3D spaces).
- Parameters of linear models are the slopes and the bias.





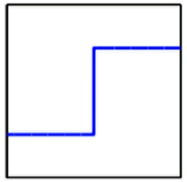
Outline

- Admin
- Three applications of linear models
- Classification
- Regression
- Normal equations
- Logistic regression
- Gradient descent
- Support Vector Machine
- Soft-margin SVM
- Kernel SVM
- Conclusion

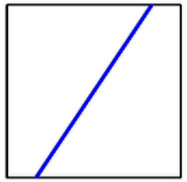
Three learning problems

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x}$$

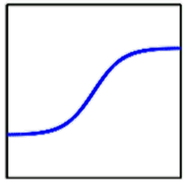
The "mother" linear function



$\text{sign}(\boldsymbol{\theta}^\top \mathbf{x})$ $y = \pm 1$ Linear classification



$\boldsymbol{\theta}^\top \mathbf{x}$ $y \in \mathbb{R}$ Linear regression



$\text{sigmoid}(\boldsymbol{\theta}^\top \mathbf{x})$ $y \in [0, 1]$ Logistic regression

Linear model for credit approval

- Goal : Design a linear model to approve or not a credit to an individual based on personal information or features.

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x} = \sum_{i=1}^{d} \theta_i x_i \begin{cases} > 0 \Rightarrow \text{Approve} \\ < 0 \Rightarrow \text{Reject} \end{cases}$$

Model parameters
Weight/importance of feature x_i

Decision by thresholding
(binary classification task)

Criterion	Value	
Age	32 years	x_1
Salary	40 K	x_2
Debt	26 K	
…	…	
Years in Job	1 year	
Years at Current Residence	3 years	x_d

Slopes of hyper-plane

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x} = \theta_d x_d + \cdots + \theta_2 x_2 + \theta_1 x_1 + \theta_0$$

bias (offset)

Example

- Extend credit using applicant's age
  - If age is higher than 25 years, accept credit.
  - Otherwise, reject.

- Linear model : $f_{\boldsymbol{\theta}}(\mathbf{x}) = x_1 - 25 > 0 \Rightarrow \text{Approve}$

  $< 0 \Rightarrow \text{Reject}$

  With $f_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x} = \theta_1 x_1 + \theta_0$, $\boldsymbol{\theta} = (\theta_1, \theta_0) = (1,-25)$ and $\mathbf{x} = (x_1, 1)$

- Example :

Criterion	Value
Age	32 years

$f_{\boldsymbol{\theta}}(\mathbf{x}) = 32 - 25 = 7 > 0 \Rightarrow \text{Approve}$

Binary classification

- **Binarize a linear function :**

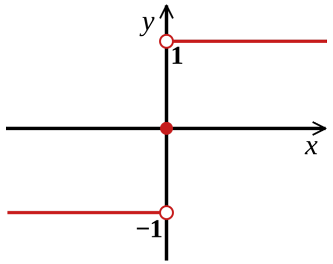
For a data input $\mathbf{x} = (x_1, x_2, \dots, x_n)$ (attributes of a customer)

Approve credit if: $\boldsymbol{\theta}^\top \mathbf{x} = \sum_{i=1}^{n} \theta_i x_i > 0$

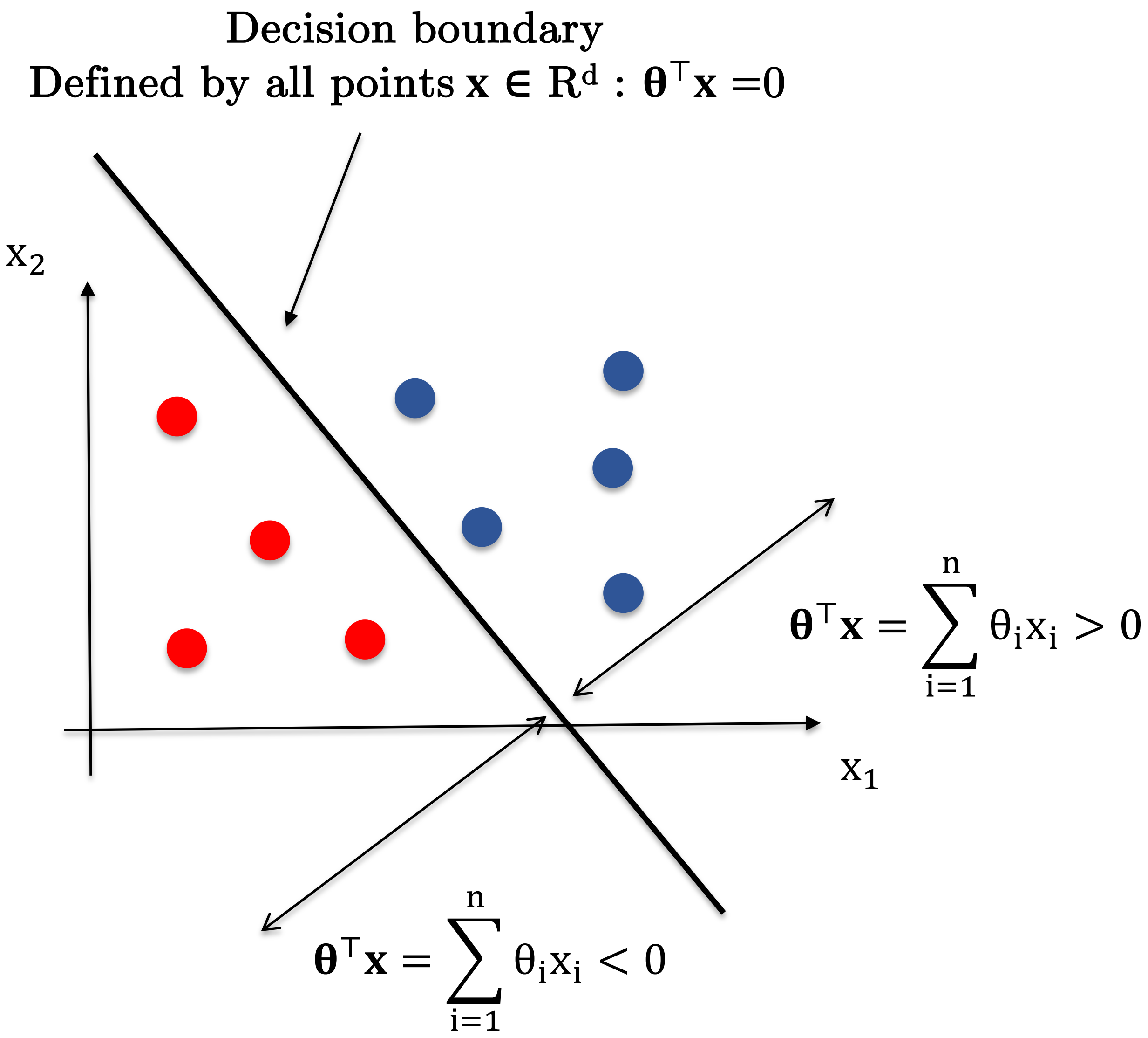
Deny credit otherwise: $\boldsymbol{\theta}^\top \mathbf{x} = \sum_{i=1}^{n} \theta_i x_i < 0$

The linear formula f can be written as:

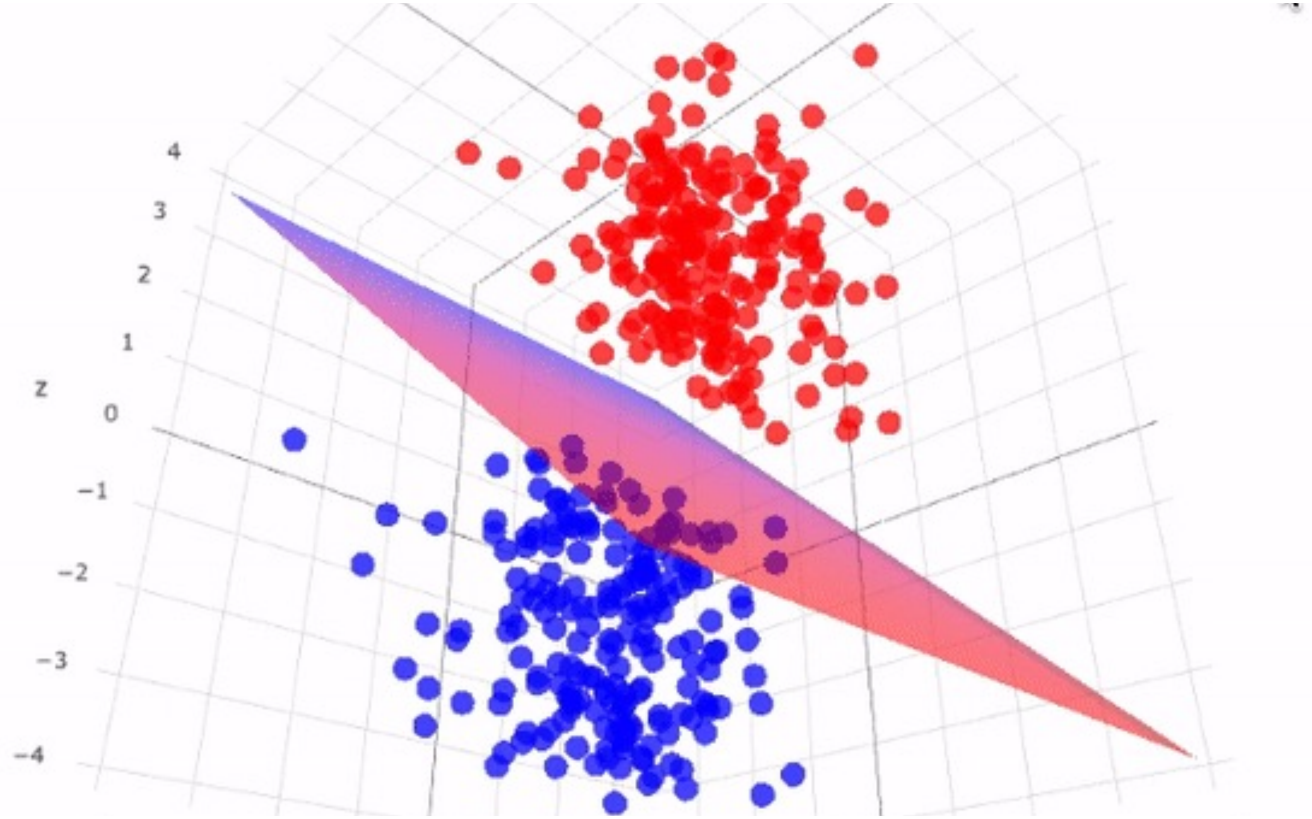
$$f_\theta(\mathbf{x}) = \text{sign}\,(\boldsymbol{\theta}^\top \mathbf{x}) = \text{sign}\,\left(\sum_{i=1}^{n} \theta_i x_i \right) = \pm 1$$



Decision boundary in 2D



Decision boundary in 3D



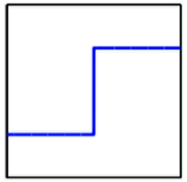
Outline

- Admin
- Three applications of linear models
- Classification
- Regression
- Normal equations
- Logistic regression
- Gradient descent
- Support Vector Machine
- Soft-margin SVM
- Kernel SVM
- Conclusion

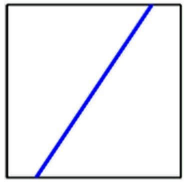
Three learning problems

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x}$$

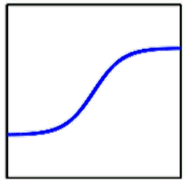
The "mother" linear function



$\text{sign}(\boldsymbol{\theta}^\top \mathbf{x})$ $\quad y = \pm 1 \quad$ Linear classification



$\boldsymbol{\theta}^\top \mathbf{x}$ $\quad y \in \mathbb{R} \quad$ Linear regression



$\text{sigmoid}(\boldsymbol{\theta}^\top \mathbf{x})$ $\quad y \in [0, 1] \quad$ Logistic regression

Linear model for credit amount

- Goal : Design a linear model to estimate the credit amount given to an individual based on personal information or features.

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x} = \sum_{i=1}^{d} \theta_i x_i \Rightarrow \text{Amount (scalar)}$$

Decision by weighted sum of data features

Criterion	Value	
Age	32 years	x_1
Salary	40 K	x_2
Debt	26 K	
…	…	
Years in Job	1 year	
Years at Current Residence	3 years	x_d

Slopes of hyper-plane

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x} = \theta_d x_d + \cdots + \theta_2 x_2 + \theta_1 x_1 + \theta_0$$

bias (offset)

Loss function

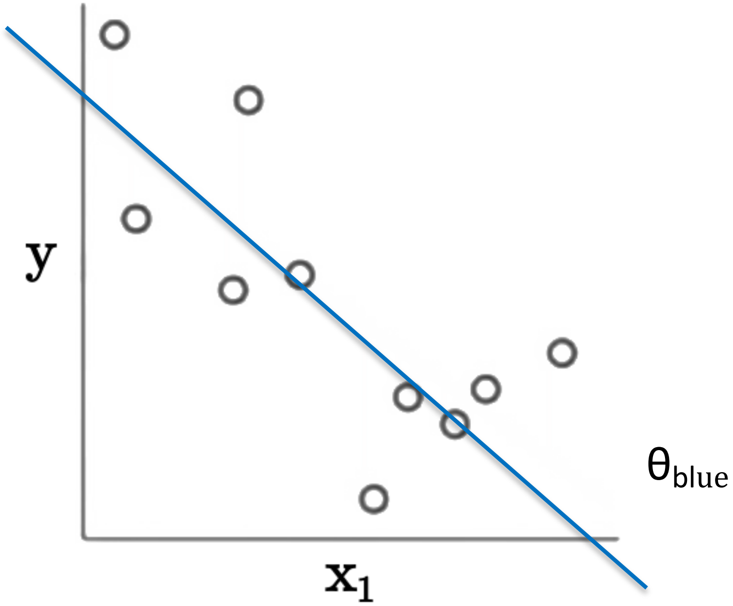
- Quality of a predictive function for a training set is evaluated with a loss function.
  - Training set : Collected/historical data $(\mathbf{X}, \mathbf{y}) = \left(\mathbf{x}^{(1)}, y^{(1)}\right), \left(\mathbf{x}^{(2)}, y^{(2)}\right), \ldots, \left(\mathbf{x}^{(n)}, y^{(n)}\right)$ where $\mathbf{X}$ is a n x d data matrix and $\mathbf{y}$ is a m-dim label vector
    - For classification, $y^{(j)} = \pm 1$ and for regression $y^{(j)} \in \mathbb{R}$.
  - Loss function computes the fit of the predictive function $f_\theta(\mathbf{x})$ to the label $\mathbf{y}$:

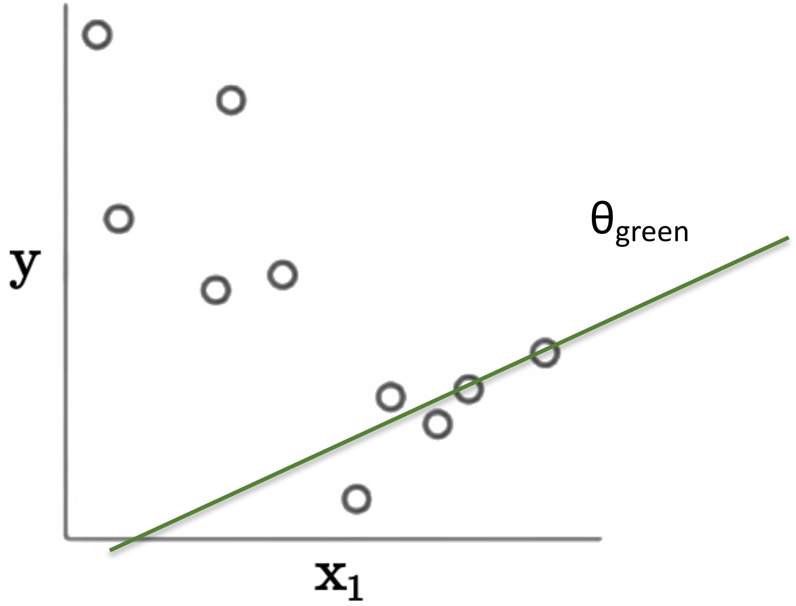
$$L(\theta) = \frac{1}{n} \sum_{j=1}^{n} \underbrace{\big(\overbrace{f_\theta\left(\mathbf{x}^{(j)}\right)}^{\text{Prediction}} - \overbrace{y^{(j)}}^{\text{Target}} \big)^2}_{\text{How good is the prediction}}$$

Mean square error (MSE)

Illustration

- The training set is composed of n data points, each data feature is 1-dimensional, i.e. $\mathbf{x} = x_1$ and the label is y.
- Find the linear model θ that fits all the data points as best as possible.



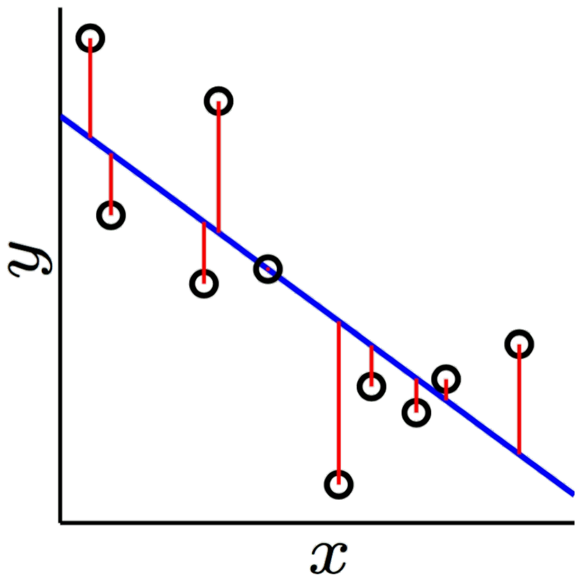


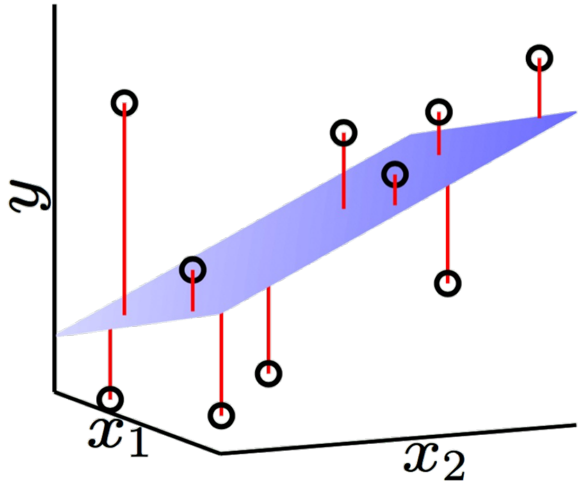
Hypothesis function : $f_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1$

Loss value: $L(\theta_{green}) \geq L(\theta_{blue})$

Illustration

- The best possible linear model $f_\theta(\mathbf{x})$ will have the minimum loss value of $L(\theta)$.





Hypothesis function : $f_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1$ $\qquad$ $f_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$

$$\min_\theta \ L(\theta) = \frac{1}{n} \sum_{j=1}^{n} \left(f_\theta\left(\mathbf{x}^{(j)}\right) - y^{(j)} \right)^2$$

Outline

- Admin
- Three applications of linear models
- Classification
- Regression
- Normal equations
- Logistic regression
- Gradient descent
- Support Vector Machine
- Soft-margin SVM
- Kernel SVM
- Conclusion

Parameter estimation

- How to compute the best parameters θ that minimizes the MSE loss function?

$$\min_\theta \; L(\theta) = \frac{1}{n} \sum_{j=1}^{n} \left(f_\theta\left(\mathbf{x}^{(j)}\right) - y^{(j)} \right)^2$$

- Two approaches
  - Normal equations
  - Gradient descent

Normal equations

- **Write the MSE loss L with matrix-vector representation :**

$$L = \frac{1}{n}\sum_{j=1}^{n}(\boldsymbol{\theta}^\top \mathbf{x}^{(j)} - y^{(j)})^2$$
$$= \frac{1}{n}\|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|^2$$

where $\underset{n\times d}{\mathbf{X}} = \begin{bmatrix} -\mathbf{x}^{(1)^\top} - \\ -\mathbf{x}^{(2)^\top} - \\ -\mathbf{x}^{(3)^\top} - \\ \vdots \\ -\mathbf{x}^{(n)^\top} - \end{bmatrix}$, $\underset{n\times 1}{\mathbf{y}} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \vdots \\ y^{(n)} \end{bmatrix}$ and $\underset{d\times 1}{\boldsymbol{\theta}} = \begin{bmatrix} \theta^{(1)} \\ \theta^{(2)} \\ \theta^{(3)} \\ \vdots \\ \theta^{(d)} \end{bmatrix}$

Normal equations

- Example of $\mathbf{X}\boldsymbol{\theta}$ and $\mathbf{y}$:

$$\mathbf{X}\boldsymbol{\theta} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} \end{bmatrix} \times \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} \theta_1 x_1^{(1)} + \theta_2 x_2^{(1)} + \theta_3 x_3^{(1)} \\ \theta_1 x_1^{(1)} + \theta_2 x_2^{(2)} + \theta_3 x_3^{(3)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}^\top \mathbf{x}^{(1)} \\ \boldsymbol{\theta}^\top \mathbf{x}^{(2)} \end{bmatrix}$$

$$\mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \end{bmatrix}$$

Normal equations

- Expand L :

$$L = \frac{1}{n} \|\mathbf{X\theta} - \mathbf{y}\|^2$$

$$= \frac{1}{n} (\mathbf{X\theta} - \mathbf{y})^\top (\mathbf{X\theta} - \mathbf{y}) = \frac{1}{n} ((\mathbf{X\theta})^\top - \boldsymbol{y}^\top)(\mathbf{X\theta} - \mathbf{y})$$

$$= \frac{1}{n} (\, (\mathbf{X\theta})^\top \mathbf{X\theta} - (\mathbf{X\theta})^\top \mathbf{y} - \mathbf{y}^\top \mathbf{X\theta} + \mathbf{y}^\top \mathbf{y} \,)$$

$$= \frac{1}{n} (\, \boldsymbol{\theta}^\top \mathbf{X}^\top \mathbf{X\theta} - 2\boldsymbol{\theta}^\top \mathbf{X}^\top \mathbf{y} + \mathbf{y}^\top \mathbf{y} \,), \quad \text{with } \boldsymbol{\theta}^\top \mathbf{X}^\top \mathbf{y} = (\mathbf{X\theta})^\top \mathbf{y} = \mathbf{y}^\top \mathbf{X\theta}$$

- Compute gradient (with calculus) and get solution :

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = \frac{1}{n} (\, 2.\mathbf{X}^\top \mathbf{X\theta} - 2.\mathbf{X}^\top \mathbf{y} \,) = \mathbf{0}$$

$$\Rightarrow \mathbf{X}^\top \mathbf{X\theta} = \mathbf{X}^\top \mathbf{y} \;\Rightarrow\; \boldsymbol{\theta} = \Big(\underbrace{(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top}_{} \Big) \mathbf{y}$$

Pseudo-inverse of **X**
Computationally expensive to
compute if d is large.

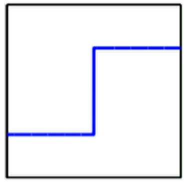
Outline

- Admin
- Three applications of linear models
- Classification
- Regression
- Normal equations
- Logistic regression
- Gradient descent
- Support Vector Machine
- Soft-margin SVM
- Kernel SVM
- Conclusion

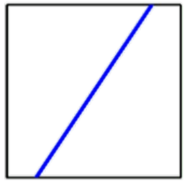
Three learning problems

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x}$$

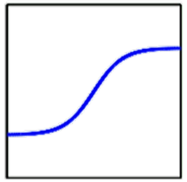
The "mother" linear function



$\mathrm{sign}(\boldsymbol{\theta}^\top \mathbf{x})$ $\quad y = \pm 1$ $\quad$ Linear classification



$\boldsymbol{\theta}^\top \mathbf{x}$ $\quad y \in \mathbb{R}$ $\quad$ Linear regression



$\mathrm{sigmoid}(\boldsymbol{\theta}^\top \mathbf{x})$ $\quad y \in [0, 1]$ $\quad$ Logistic regression

Logistic regression

- Linear classification predicts "hard" classes, e.g. ± 1.
- Linear regression predicts scalar target, e.g. $\in \mathbb{R}$.
- Logistic regression predicts "soft" classes, i.e. $f_\theta(\mathbf{x})$ is interpreted as a probability of class $\in [0,1]$.
- Example
  - Prediction of a heart attack
  - Input $\mathbf{x}$: Cholesterol level, age, diabetic, etc.
  - Output $f_\theta(\mathbf{x})$: Likelihood of a heart attack $\in [0,1]$
  - $\theta^\top \mathbf{x} \in \mathbb{R}$ is the risk score.

Logistic regression

- Training data : $\mathbf{X} = \left(\mathbf{x}^{(1)}, \mathrm{y}^{(1)}\right), \dots, \left(\mathbf{x}^{(n)}, \mathrm{y}^{(n)}\right)$ with
  - $\mathbf{x}^{(j)}$: Person's health information, i.e. data features
  - $\mathrm{y}^{(j)}$: Person has a heart attack or not
- Labels $\mathbf{y}$ are binary, i.e. $\mathrm{y}^{(j)} = \pm 1$.
  - We know with certainty whether a person got a heart attack, $\mathrm{y}^{(j)} = +1$, or not, $\mathrm{y}^{(j)} = -1$.
- Prediction function $\mathrm{f}_\theta(\mathbf{x})$ is the probability in [0,1] that the person had a heart attack.
  - Hence, the probability of a person with no heart attack is 1 - $\mathrm{f}_\theta(\mathbf{x})$.

$$\mathrm{P}_{\boldsymbol{\theta}}(\mathrm{y}|\mathbf{x}) = \begin{cases} \mathrm{f}_\theta(\mathbf{x}) & \text{for } \mathrm{y} = +1 \\ 1 - \mathrm{f}_\theta(\mathbf{x}) & \text{for } \mathrm{y} = -1 \end{cases}$$

Probability of having class y given the input data x
Conditioned probability a.k.a. likelihood probability

Logistic regression

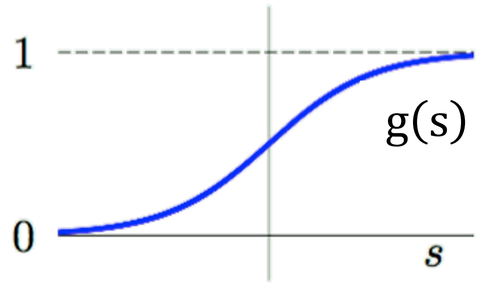
- Prediction $f_\theta(\mathbf{x})$ can be designed as :

$$f_\theta(\mathbf{x}) = \text{sigmoid}(\boldsymbol{\theta}^\top \mathbf{x}) \in [0,1]$$

- Choice of logistic function :
  - Function sigmoid(.)=g(.) is a smooth step function or soft thresholding function.

$$g(s) = \frac{\exp(s)}{1+\exp(s)} = \frac{1}{1+\exp(-s)}$$

$$g(-s) = 1 - g(s)$$



$g : \mathbb{R} \to [0,1]$

Logistic regression

- Loss function for logistic regression
  - Estimate how good is the predictive function $f_\theta(\mathbf{x})$ = sigmoid($\boldsymbol{\theta}^\top \mathbf{x}$) to infer correctly the class y of $\mathbf{x}$:

$$\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \frac{1}{n} \sum_{j=1}^{n} \log(1 + \exp(-y^{(j)} \boldsymbol{\theta}^\top \mathbf{x}^{(j)}))$$

Logistic regression function
a.k.a cross-entropy loss

Number of
training data

Explained in the next slides

Logistic regression

- Understanding the logistic regression loss for one data point $(\mathbf{x},y)$:

Sigmoid

$$P_{\boldsymbol{\theta}}(y|\mathbf{x}) = \begin{cases} g(\boldsymbol{\theta}^\top\mathbf{x}) = f_\theta(\mathbf{x}) & \text{for } y=+1 \\ g(-\boldsymbol{\theta}^\top\mathbf{x}) = 1 - g(\boldsymbol{\theta}^\top\mathbf{x}) = 1 - f_\theta(\mathbf{x}) & \text{for } y=-1 \end{cases} = g(y.\boldsymbol{\theta}^\top\mathbf{x})$$

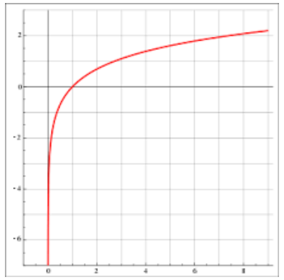
$$g(-s) = 1 - g(s)$$

$$P_{\boldsymbol{\theta}}(y|\mathbf{x}) = g(y.\boldsymbol{\theta}^\top\mathbf{x}) = \frac{1}{1+\exp(-y.\boldsymbol{\theta}^\top\mathbf{x})} \quad \text{with} \quad g(s) = \frac{1}{1+\exp(-s)}$$

$$\log P_{\boldsymbol{\theta}}(y|\mathbf{x}) = -\log(1+\exp(-y.\boldsymbol{\theta}^\top\mathbf{x}))$$

$$\max_\theta P_{\boldsymbol{\theta}}(y|\mathbf{x}) \Leftrightarrow \max_\theta \log P_{\boldsymbol{\theta}}(y|\mathbf{x}) = \max_\theta -\log(1+\exp(-y.\boldsymbol{\theta}^\top\mathbf{x}))$$

$$\Leftrightarrow \min_{\boldsymbol{\theta}} \log(1+\exp(-y.\boldsymbol{\theta}^\top\mathbf{x}))$$



Monotonous
increasing function

Logistic regression

- Understanding the logistic regression loss for all data points $\mathbf{X} = (\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(n)}, y^{(n)})$:
  - Suppose that the data points are i.i.d. samples (independent and identically distributed).

$$P(y^{(1)}, \ldots, y^{(n)} | \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}) = \prod_{j=1}^{n} P(y^{(j)} | \mathbf{x}^{(j)})$$

$$\log P(y^{(1)}, \ldots, y^{(n)} | \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}) = \log \prod_{j=1}^{n} P(y^{(j)} | \mathbf{x}^{(j)}) = \sum_{j=1}^{n} \log P(y^{(j)} | \mathbf{x}^{(j)})$$

$$\log \prod_{j=1}^{n} = \sum_{j=1}^{n} \log$$

$$\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \frac{1}{n} \sum_{j=1}^{n} \log(1 + \exp(-y^{(j)} \boldsymbol{\theta}^\top \mathbf{x}^{(j)}))$$

Outline

- Admin
- Three applications of linear models
- Classification
- Regression
- Normal equations
- Logistic regression
- Gradient descent
- Support Vector Machine
- Soft-margin SVM
- Kernel SVM
- Conclusion

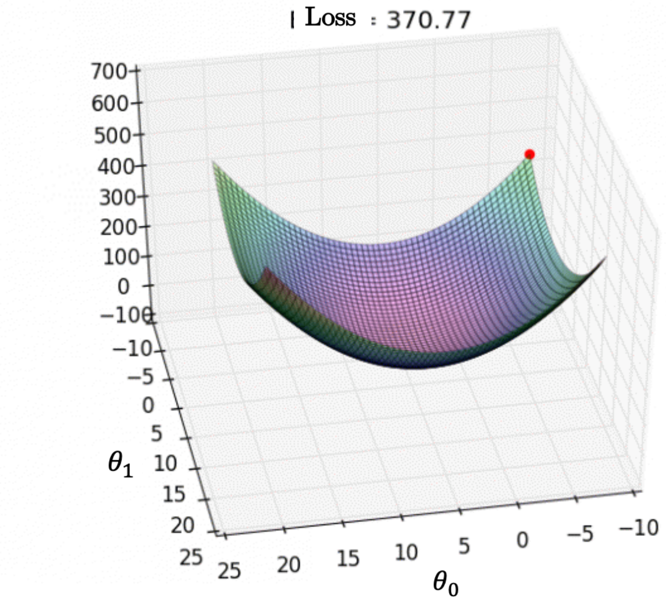
Gradient optimization

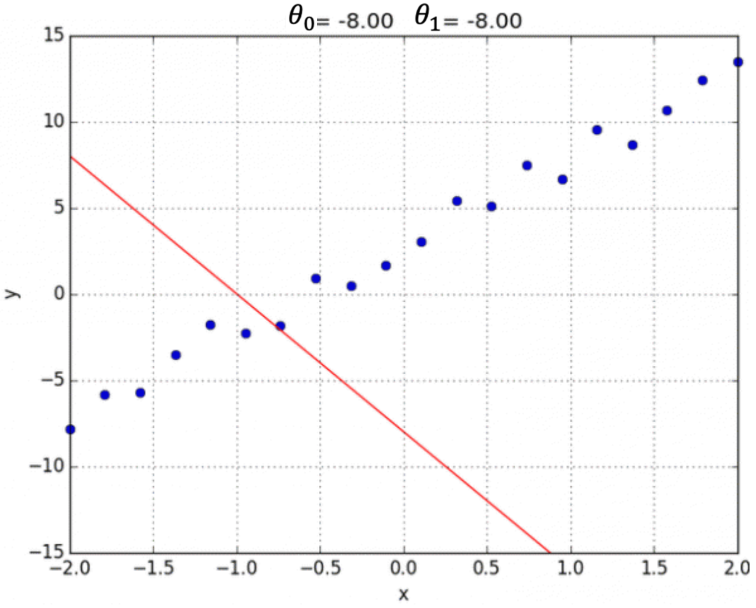
- Gradient optimization algorithm (simplest optimization algorithm)
  - Work well in high-dimensional spaces (because convergence is independent of data dimensionality)
- Two versions : Gradient descent/ascent
  - Start randomly
  - Move in the direction of the steepest descent/ascent.
  - Stop when reaching the minimum/maximum.

Gradient descent

- Predict amount of credit y with data feature x (single feature) :
  - Prediction function : $f_\theta(\mathbf{x}) = \theta_0 + \theta_1 x$
  - Parameters : θ_0, θ_1
  - Loss function : $L(\theta = (\theta_0, \theta_1)) = \frac{1}{n}\sum_{j=1}^{n} (f_\theta(x^{(j)}) - y^{(j)})^2$
  - Loss minimized by gradient descent (introduced next slides):

$$\min_\theta \; L(\theta = (\theta_0, \theta_1))$$

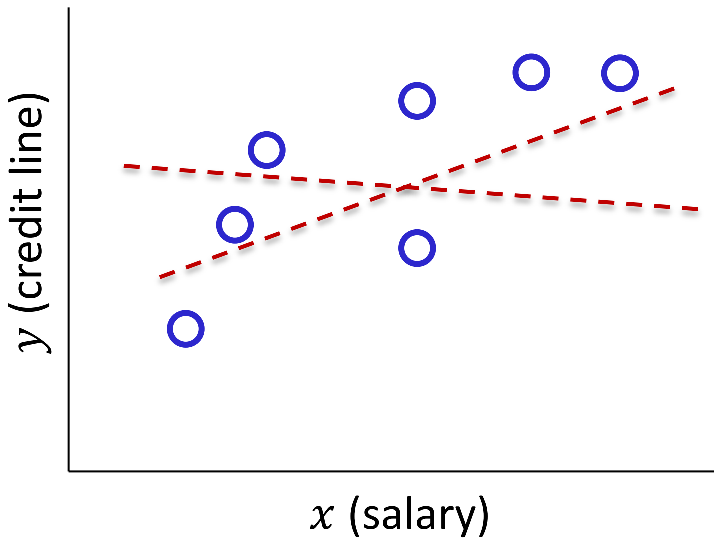




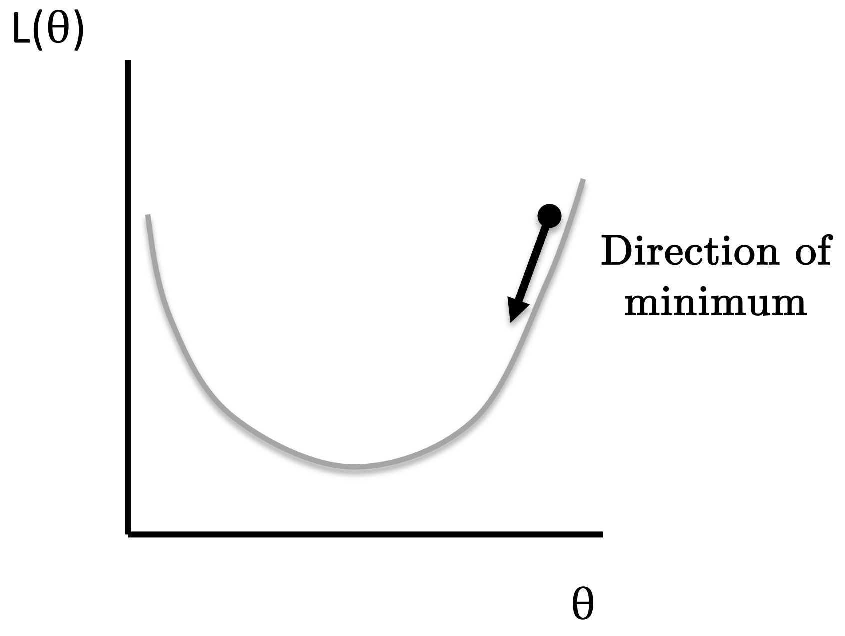
$f_\theta(\mathbf{x}) = \theta_0 + \theta_1 x$

Gradient descent

- Univariate linear regression
  - (Single) parameter θ : slope of the line
- Suppose loss function is a convex function of θ, e.g. $L(\theta)=(\theta x - y)^2$

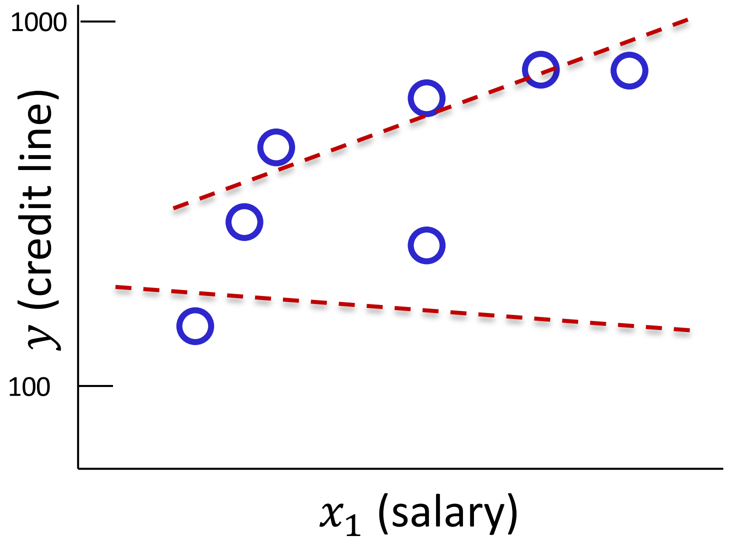


$$f_\theta(\mathbf{x}) = \theta\, x$$

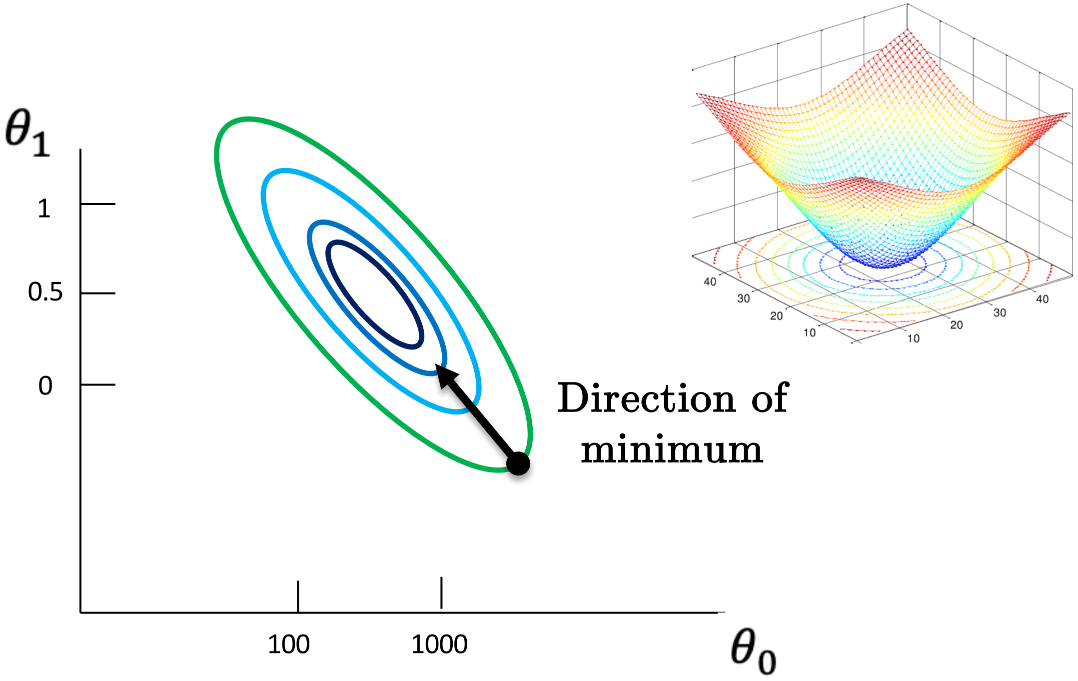


Gradient descent

- Multivariate linear regression
  - Two parameters θ_0, θ_1 : bias and slope of the line
- Suppose loss function is a convex function of $\theta = (\theta_0,\theta_1)$, e.g. $L(\theta_0,\theta_1)=(\theta_0 + \theta_1 x - y)^2$



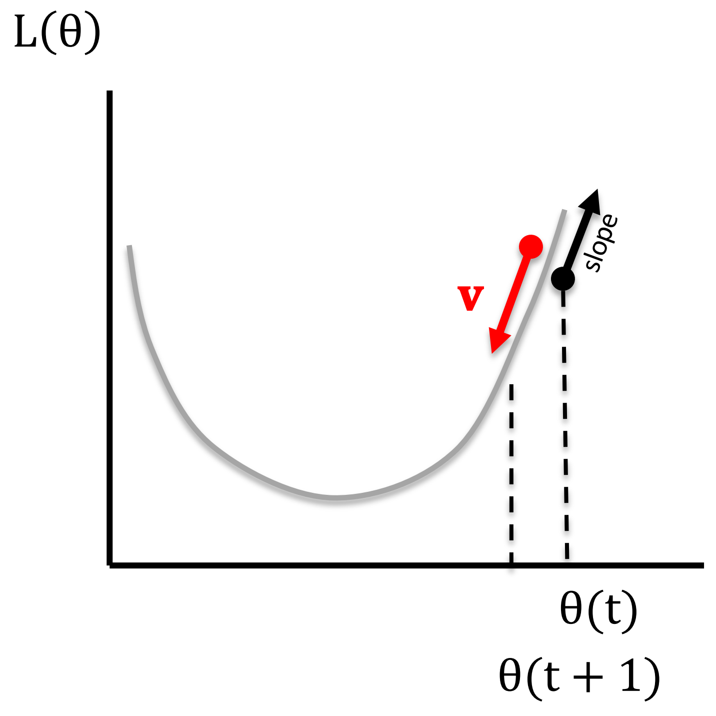
$f_\theta(\mathbf{x}) = \theta_0 + \theta_1 x$



$L(\theta_0, \theta_1)$

Gradient descent

- $\boldsymbol{\theta}(t)$: Vector of all parameters at iteration t
- Start at $\boldsymbol{\theta}(t{=}0)$ = random
- Iterative method
  - Update the value of parameters :
    $\boldsymbol{\theta}(t{+}1){=}\boldsymbol{\theta}(t){+}\mathbf{v}$
  - What is the best direction $\mathbf{v}$?
    - Direction $\mathbf{v}$ must minimize as much as possible the loss function L.
    - $\mathbf{v}$ = Opposite direction of the steepest slope (justification given next slides)



Gradient descent

- Direction $\mathbf{v}$ of the steepest descent :

First-order Taylor approximation

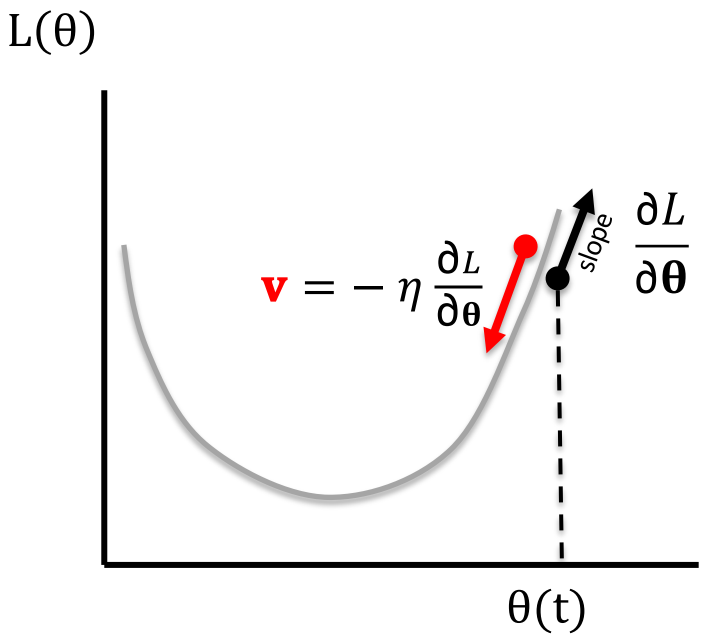
$$L(\boldsymbol{\theta}+\Delta\boldsymbol{\theta}) \approx L(\boldsymbol{\theta}) + \left\langle \frac{\partial L}{\partial \boldsymbol{\theta}}, \Delta\boldsymbol{\theta} \right\rangle$$

$$\Delta L = L(\boldsymbol{\theta}+\Delta\boldsymbol{\theta}) - L(\boldsymbol{\theta}) \approx \left\langle \frac{\partial L}{\partial \boldsymbol{\theta}}, \Delta\boldsymbol{\theta} \right\rangle$$

$$\min_{\Delta\boldsymbol{\theta}} \Delta L \;\Rightarrow\; \Delta\boldsymbol{\theta} = -\eta \frac{\partial L}{\partial \boldsymbol{\theta}},\; \eta > 0 \;\Rightarrow\; \Delta L = -\eta \left| \frac{\partial L}{\partial \boldsymbol{\theta}} \right|^2 < 0$$

$\Delta\boldsymbol{\theta}$ aligns with $\frac{\partial L}{\partial \boldsymbol{\theta}}$ in the opposite direction to minimize.

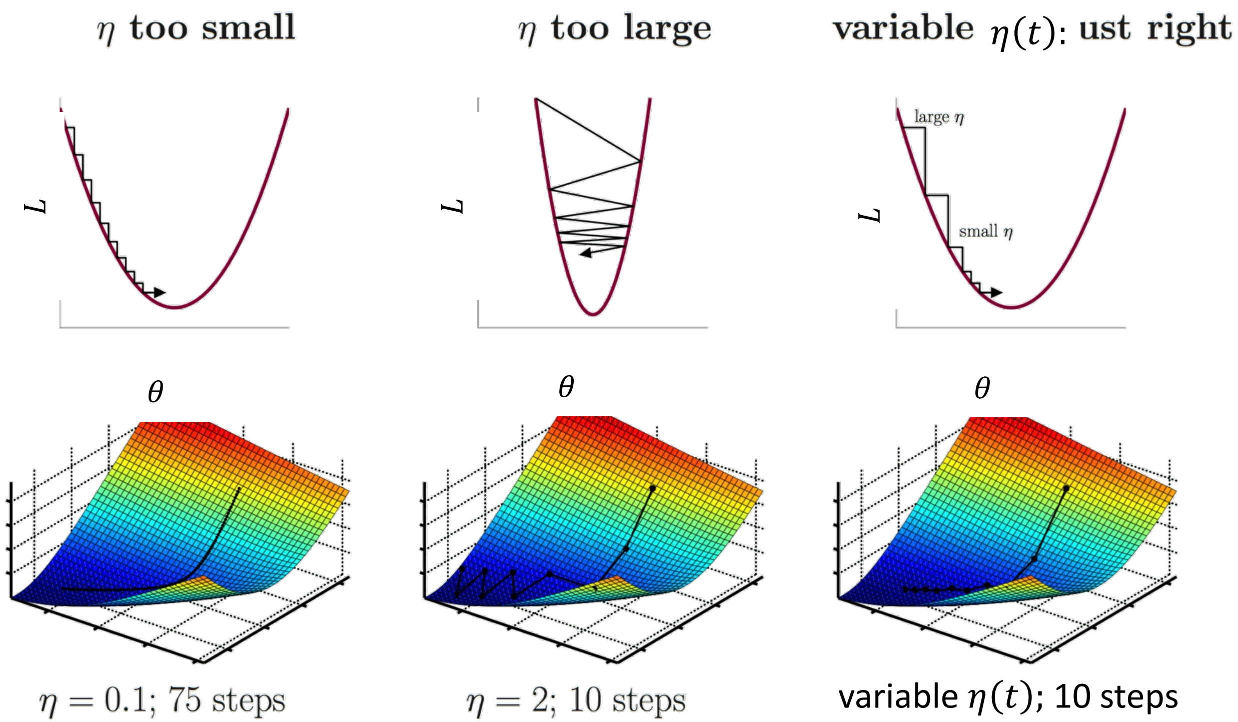
$$\underbrace{\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \Delta\boldsymbol{\theta}(t) = \boldsymbol{\theta}(t) - \eta \frac{\partial L}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}(t))}_{\text{Update equation for gradient descent}}$$



$$\mathbf{v}(t) = \Delta\boldsymbol{\theta}(\mathbf{t}) = -\eta \frac{\partial L}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}(t))$$

Gradient descent

- Step size η controls how fast the gradient descent algorithm descends to the minimum.
- Finding the right value of the step size is heuristic :



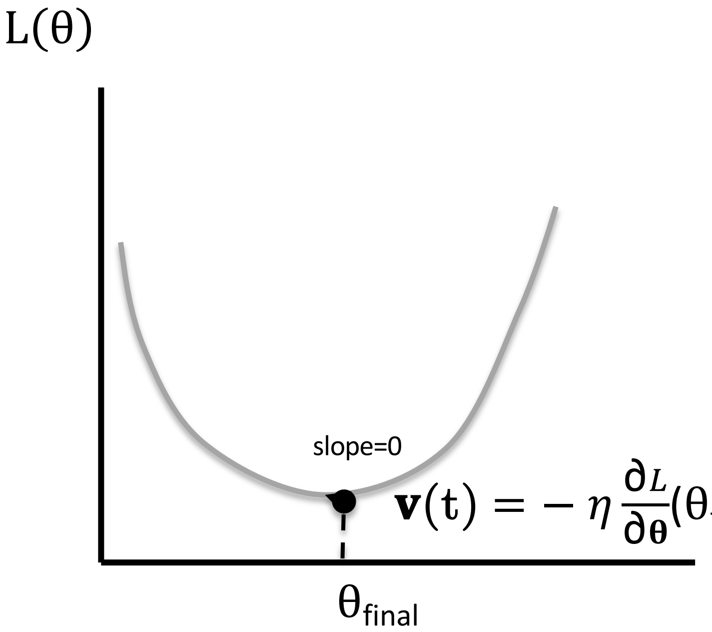
Too slow

Diverges

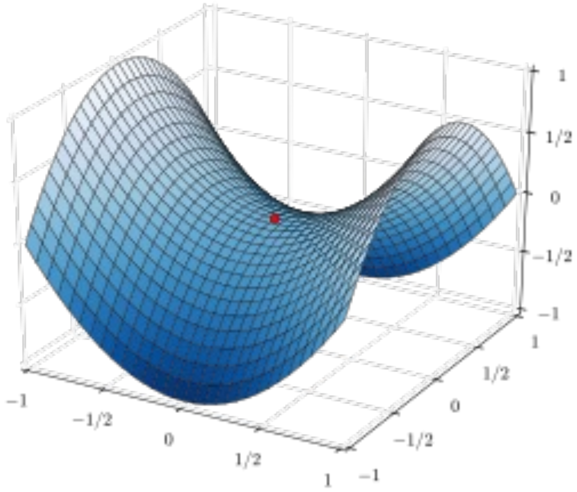
Right speed

Gradient descent

- Termination condition / when to stop the iterative scheme ?
  - Natural choice : when gradient < (arbitrary) threshold
- However, lots of saddle points / flat regions in high-dimensional spaces.

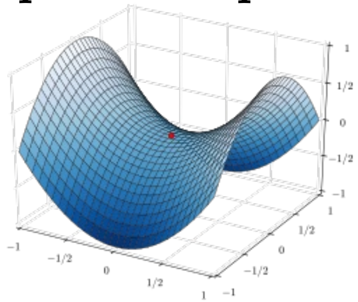


$$\mathbf{v}(t) = -\eta \frac{\partial L}{\partial \boldsymbol{\theta}}(\theta_{\text{final}}) = \mathbf{0}$$



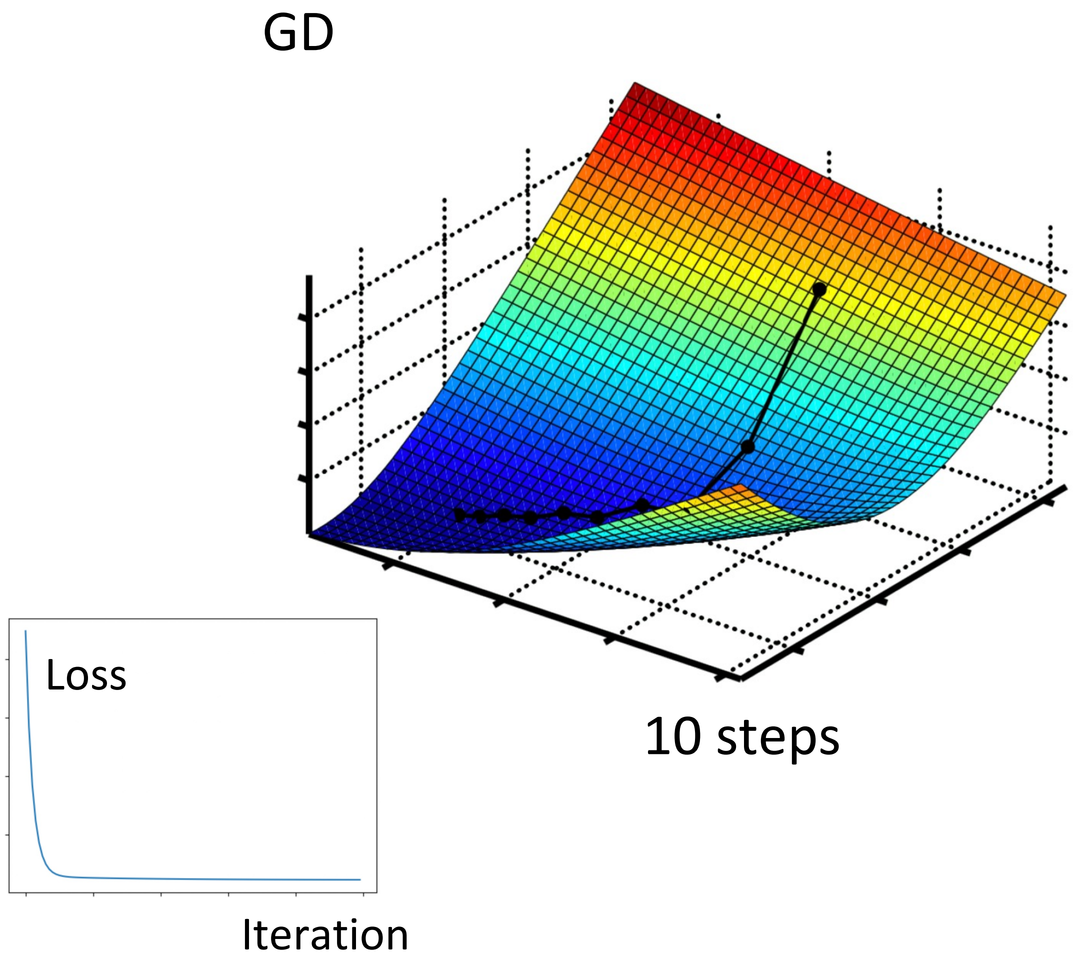
Stochastic gradient descent

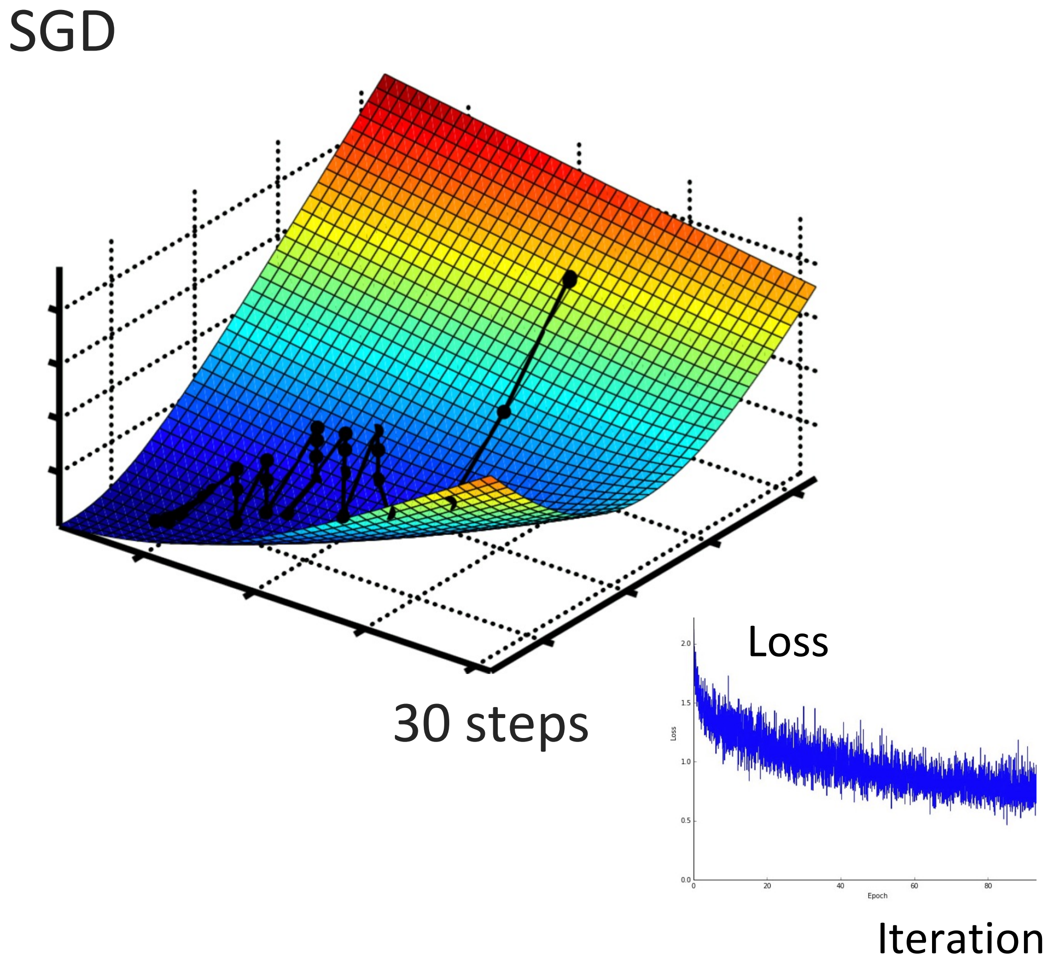
- A fast variation of gradient descent that considers only a small set of data points to update the value of the parameters is stochastic gradient descent (SGD):
  - Pick a (random) small subset of m training data (e.g. a single/512 data points), i.e. $(\mathbf{x}^{(k)}, y^{(k)})$
  - Compute the loss value for this subset, i.e. $L = \frac{1}{m}\sum_{k=1}^{m} L(f_\theta(\mathbf{x}^{(k)}), y^{(k)})$
  - Compute the gradient for this subset, i.e. $\mathbf{v} = \frac{1}{m}\sum_{k=1}^{m} -\nabla L\,(f_\theta(\mathbf{x}^{(k)}), y^{(k)})$
  - Update parameters, i.e. $\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \eta\,\mathbf{v}$
- Advantages
  - Faster update of parameters : Gradient computed with m=512 rather than all data points.
  - Stochastic optimization : Helps escape saddle points in high-dimensional space
  - Simple to implement



Stochastic gradient descent

- GD vs. SGD with by a mini-batch of m=10 samples





Summary

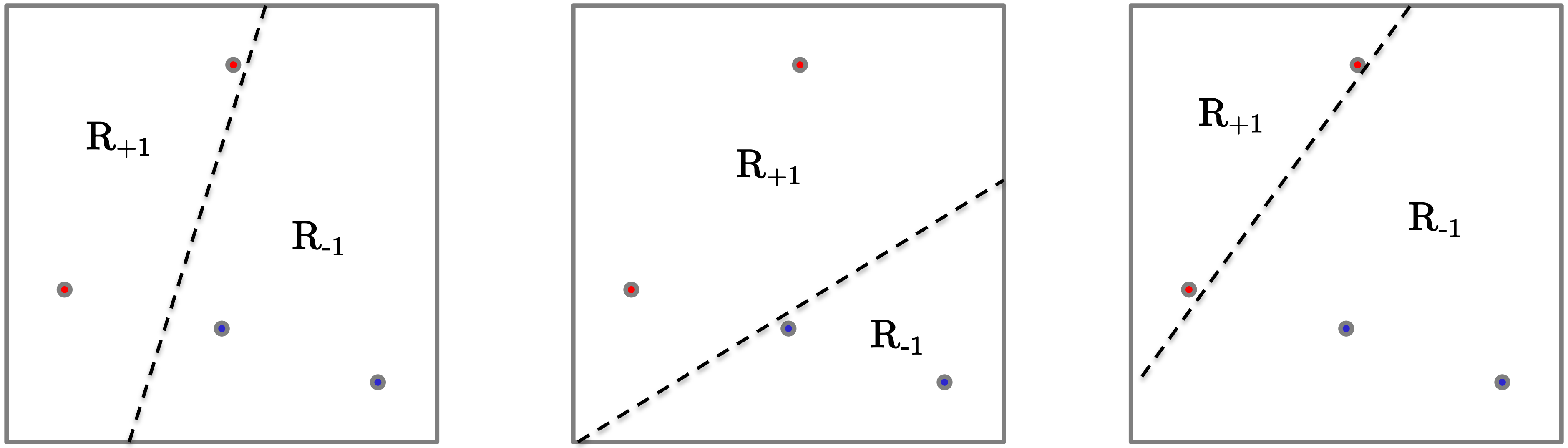
- Two methods for solving the linear regression task :
  - Gradient Descent
    - Works well for very large n, #training data, with SGD (300B tokens with GPT3)
    - Works well for very large d, #data features (d=1M with 1,000×1,000 images)
    - Works well for very large $|\boldsymbol{\theta}|$, #parameters features (175B with GPT3)
    - Very slow and requires to select time step η
  - Normal Equations
    - Very fast for d = $O(10^3)$ data points, do not require to choose η
    - Need to compute $(\mathbf{X}^T\mathbf{X})^{-1}$, $O(d^3)$ operation but faster approximations exist
- Gradient Descent is a universal optimization technique as long as the considered loss is continuous and differentiable (as gradient is required).
  - GD does not work for discrete losses like win/lose at the game of Go.

Outline

- Admin
- Three applications of linear models
- Classification
- Regression
- Normal equations
- Logistic regression
- Gradient descent
- Support Vector Machine
- Soft-margin SVM
- Kernel SVM
- Conclusion

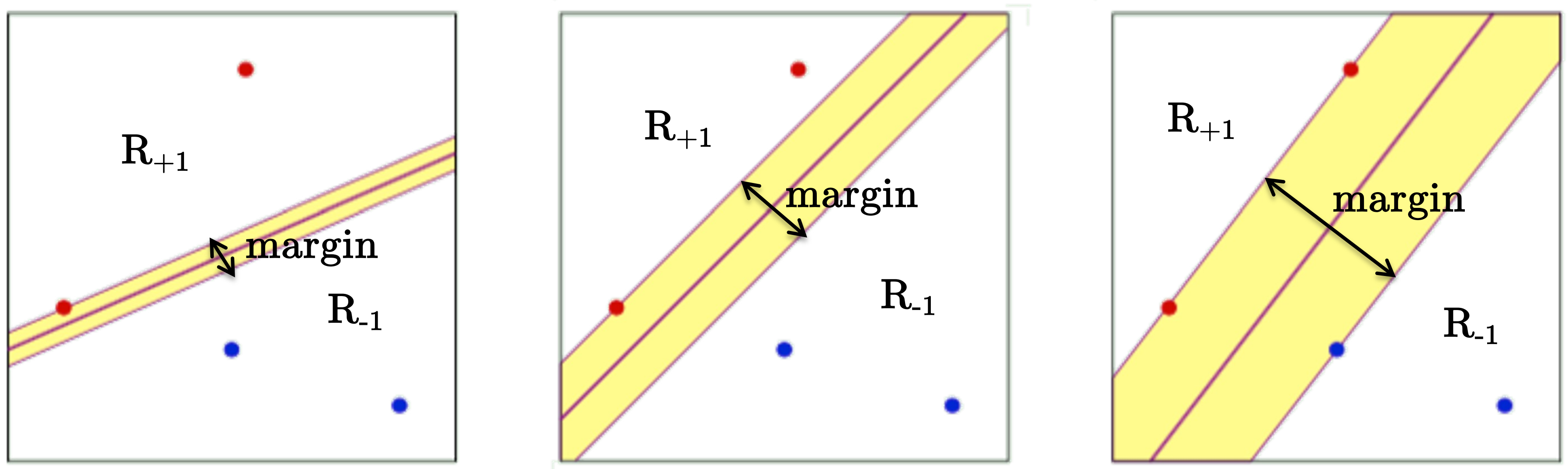
Support Vector Machine

- Consider the binary classification task and linear functions as hypothesis space.
- The goal is to find a linear separator that partitions the feature space into two regions.
- Generally, there exist several possible linear separators.
- How to select an "optimal" linear separator, and how to define "optimal"?



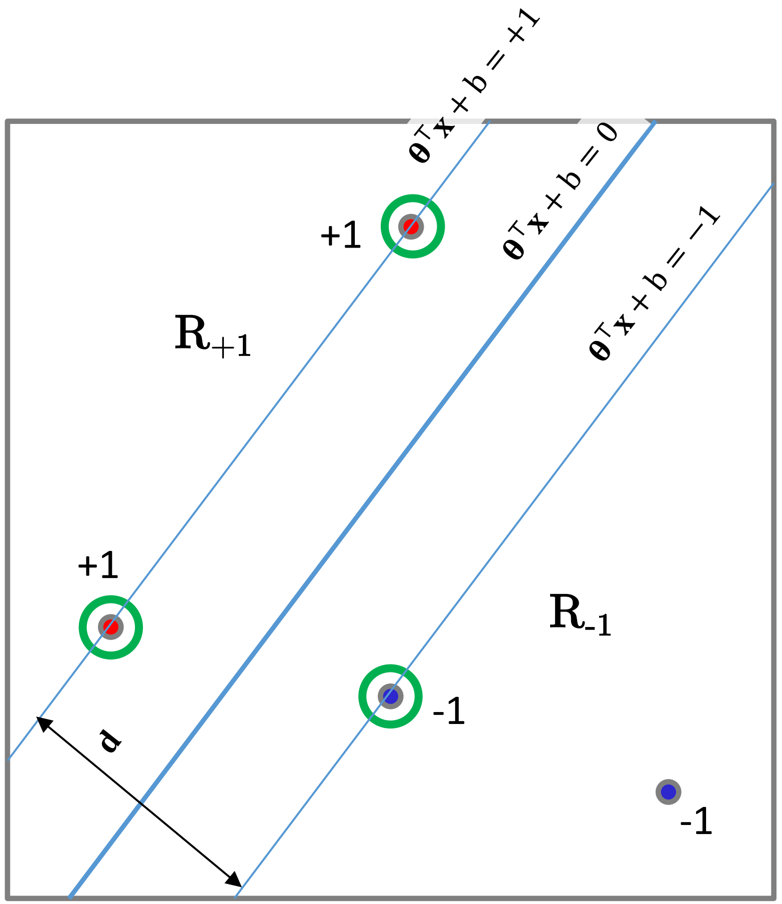
Support Vector Machine

- SVM technique (1964) aims at maximizing the margin between the two classes.
- Maximizing the margin provides better results.



Support Vector Machine

- Hyper-plane equation : { $\mathbf{x}$ such that $f_\theta(\mathbf{x})=\boldsymbol{\theta}^\top\mathbf{x} + b = cte$ }
- Hyper-planes $f_\theta(\mathbf{x})=\pm 1$ are margin planes.
- Hyper-plane $f_\theta(\mathbf{x})=0$ is the class separator.
- Training data points +1 lie above the hyperplane and -1 data points below.
- Parameters $\boldsymbol{\theta}$ controls the orientation/slope of the plane, and b is the offset/bias.
- The distance $\boldsymbol{d}$ between the two margin planes is computed to be as large as possible.
- What are the parameters $\boldsymbol{\theta}$ that maximizes the distance $\boldsymbol{d}$ between the two margins? (response next slide)



Support Vector Machine

- What are the parameters $\boldsymbol{\theta}$ that maximizes the margin?
- Optimal value $\boldsymbol{\theta}$ is solution of a constrained quadratic optimization problem :

$$\text{Given } f_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x} + b$$

$$\text{with } f_{\boldsymbol{\theta}}(\mathbf{x}_{+1}) = \boldsymbol{\theta}^\top \mathbf{x}_{+1} + b = +1$$
$$f_{\boldsymbol{\theta}}(\mathbf{x}_{-1}) = \boldsymbol{\theta}^\top \mathbf{x}_{-1} + b = -1$$

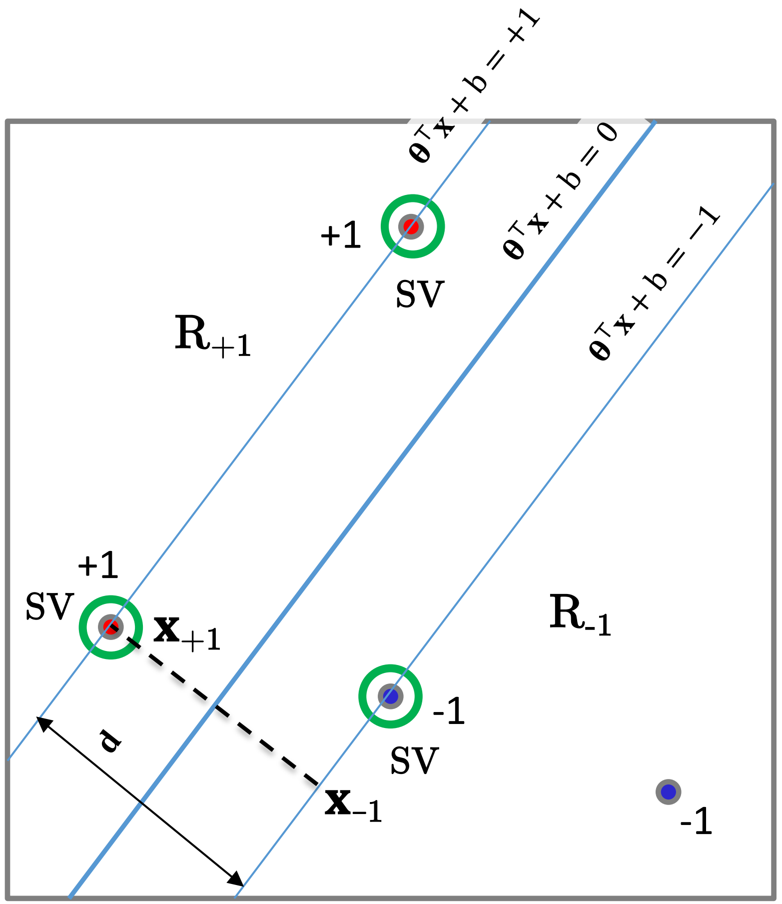
$$\text{We have } f_{\boldsymbol{\theta}}(\mathbf{x}_{+1}) - f_{\boldsymbol{\theta}}(\mathbf{x}_{-1}) = \boldsymbol{\theta}^\top (\mathbf{x}_{+1} - \mathbf{x}_{-1}) = +2$$

If we define the margin vector as $\mathbf{d} = \mathbf{x}_{+1} - \mathbf{x}_{-1}$ then

$$\boldsymbol{\theta}^\top (\mathbf{x}_{+1} - \mathbf{x}_{-1}) = +2 \Rightarrow \|\boldsymbol{\theta}\| . \|\boldsymbol{d}\| = +2 \Rightarrow d = \frac{2}{\|\boldsymbol{\theta}\|}$$

Length of the margin vector

$$\max_d d = \frac{2}{\|\boldsymbol{\theta}\|} \Leftrightarrow \min_{\boldsymbol{\theta}} \|\boldsymbol{\theta}\|^2 \text{ such that } \begin{cases} f_{\boldsymbol{\theta}}(\mathbf{x}) \geq +1 \text{ for } \mathbf{x} \in R_{+1} \\ f_{\boldsymbol{\theta}}(\mathbf{x}) \leq -1 \text{ for } \mathbf{x} \in R_{-1} \end{cases}$$



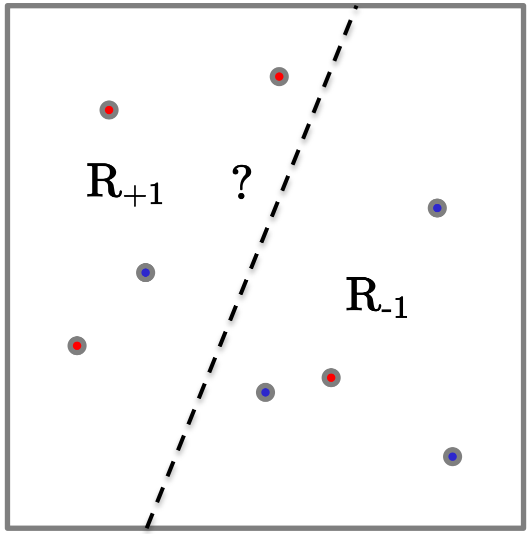
Only a subset of the data points have value $f(\mathbf{x}) = \pm 1$. These are called support vectors (SV).

Outline

- Admin
- Three applications of linear models
- Classification
- Regression
- Normal equations
- Logistic regression
- Gradient descent
- Support Vector Machine
- Soft-margin SVM
- Kernel SVM
- Conclusion

Soft-margin SVM

- Real-world data is noisy.
- In case of data non-linearly separable, i.e. with outliers, there is no mathematical solution to standard/hard-margin SVM.
  - In other words, there exists no linear separator that can split the two classes perfectly, i.e. without errors.



Soft-margin SVM

- SVM can be improved to deal with outliers.
  - Soft-margin SVM technique, 1995
  - Idea is to introduce a slack variable $e^{(j)}$ for each data point that represents the prediction error.
  - These errors $e^{(j)}$ will be minimized while simultaneously maximizing the margin :

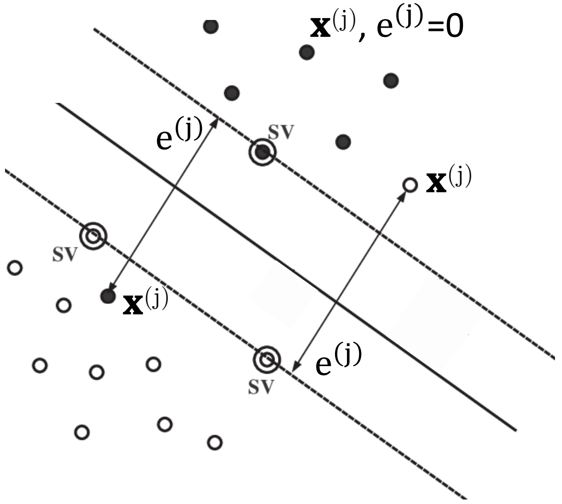
$$\min_{\boldsymbol{\theta}} \ \|\boldsymbol{\theta}\|^2 \text{ such that } \begin{cases} f_{\boldsymbol{\theta}}(\mathbf{x}) \geq +1 \text{ for } \mathbf{x} \in R_{+1} \\ f_{\boldsymbol{\theta}}(\mathbf{x}) \leq -1 \text{ for } \mathbf{x} \in R_{-1} \end{cases}$$

Standard SVM

$$\Downarrow$$

$$\min_{\boldsymbol{\theta},\mathbf{e}} \|\boldsymbol{\theta}\|^2 + C.\sum_{j=1}^{n} e^{(j)} \text{ such that } \begin{cases} f_{\boldsymbol{\theta}}(\mathbf{x}^{(j)}) \geq +1 - e^{(j)} \text{ for } \mathbf{x} \in R_{+1} \\ f_{\boldsymbol{\theta}}(\mathbf{x}^{(j)}) \leq -1 + e^{(j)} \text{ for } \mathbf{x} \in R_{-1} \\ e^{(j)} \geq 0, \ C \geq 0 \end{cases}$$

Soft-margin SVM

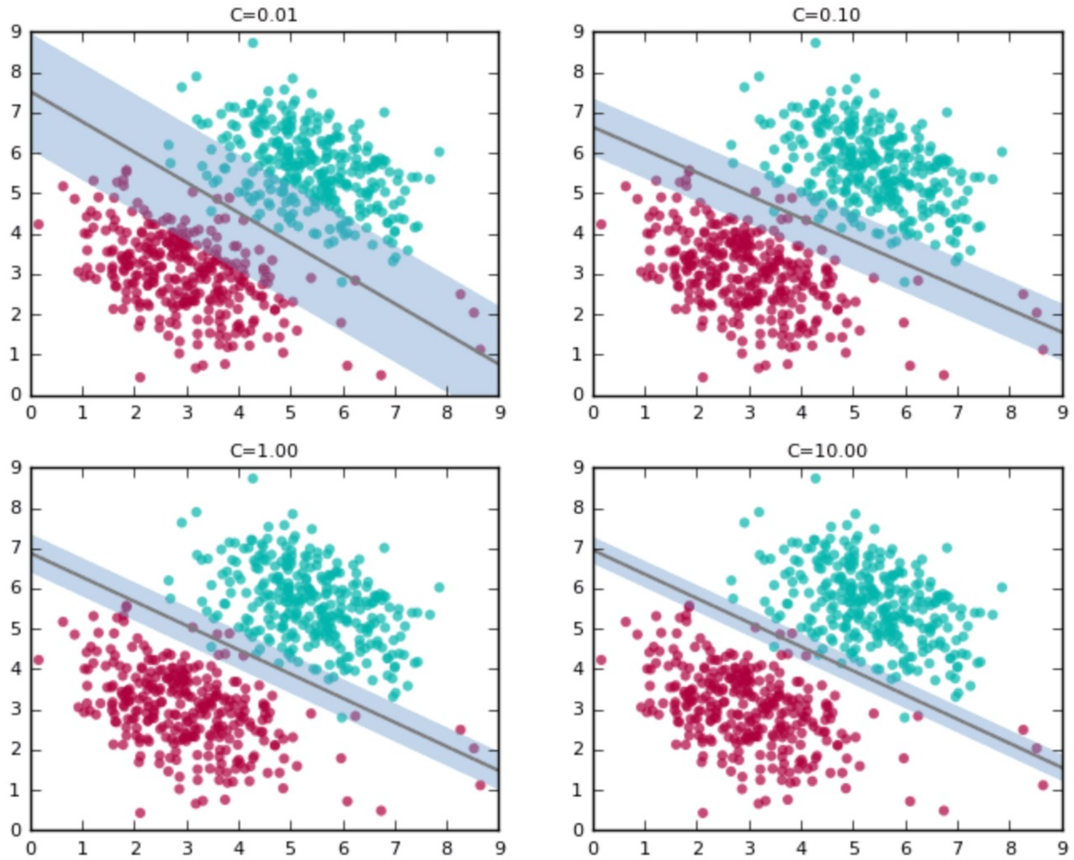


Soft-margin SVM

- Effect of varying C, the regularization parameter.

Goal is to find the largest margin. When C is small, more misclassification errors are allowed.

Note that the margin is large then.



When C is large, less misclassification error are allowed, possibly none.
Note that the margin is small then.

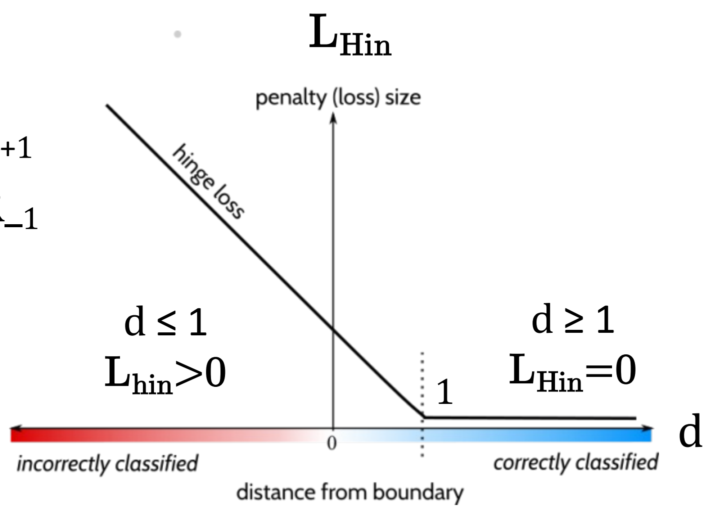
Soft-margin SVM

- Soft-margin SVM penalizes
  - Misclassifications,
  - Correct classifications that fall inside the margin.
- Loss of soft-margin : Hinge loss (popular loss function)

$$\min_{\boldsymbol{\theta},\mathbf{e}} \|\boldsymbol{\theta}\|^2 + C.\sum_{j=1}^{n} e^{(j)} \text{ such that } \begin{cases} f_{\boldsymbol{\theta}}(\mathbf{x}^{(j)}) \geq +1 - e^{(j)} \text{ for } \mathbf{x}^{(j)} \in R_{+1} \\ f_{\boldsymbol{\theta}}(\mathbf{x}^{(j)}) \leq -1 + e^{(j)} \text{ for } \mathbf{x}^{(j)} \in R_{-1} \\ e^{(j)} \geq 0, \ C \geq 0 \end{cases}$$

[proof not included -- it will not be assessed] $\Updownarrow$

$$\min_{\boldsymbol{\theta}} \|\boldsymbol{\theta}\|^2 + C.\sum_{j=1}^{n} \underbrace{\max\left(0, 1 - y^{(j)}.f_{\boldsymbol{\theta}}(\mathbf{x}^{(j)})\right)}_{\substack{L_{Hin}\left(d^{(j)} = y^{(j)}.f_{\boldsymbol{\theta}}(\mathbf{x}^{(j)})\right) \\ \text{(Hinge loss)}}}$$



$d \geq 1$ (correctly classified) if

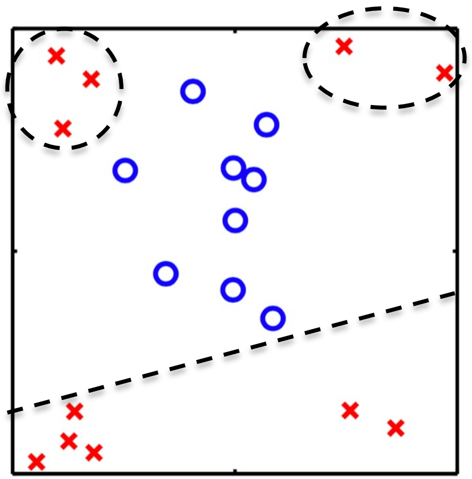
$$\begin{cases} f_{\boldsymbol{\theta}}(\mathbf{x}^{(j)}) \geq +1 \text{ and } y^{(j)} = +1 \\ f_{\boldsymbol{\theta}}(\mathbf{x}^{(j)}) \leq -1 \text{ and } y^{(j)} = -1 \end{cases}$$

[Optional] Proof given in http://image.diku.dk/imagecanon/material/cortes_vapnik95.pdf

Outline

- Admin
- Three applications of linear models
- Classification
- Regression
- Normal equations
- Logistic regression
- Gradient descent
- Support Vector Machine
- Soft-margin SVM
- Kernel SVM
- Conclusion

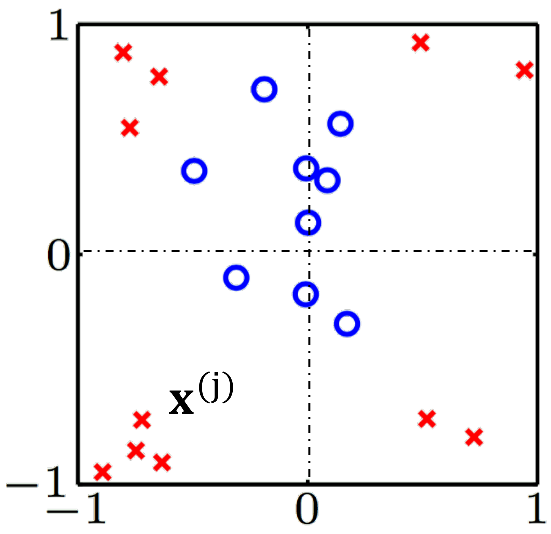
Kernel SVM

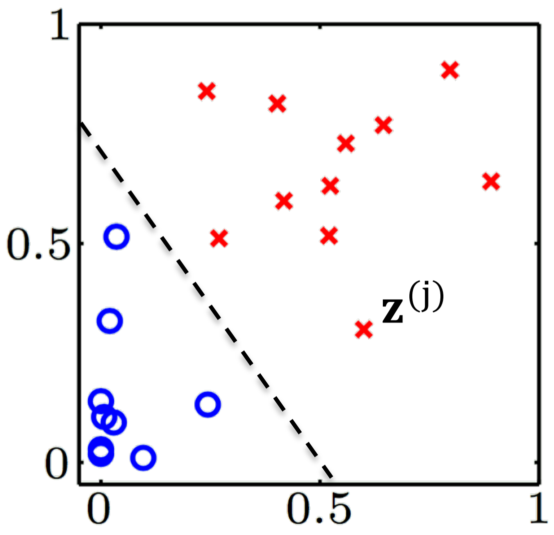


- Linear models are limited to linearly separable data points.
- How to classify complex/non-linear datasets with linear separators?
- Idea is to map the data from their original space R^d, where classes can only be separated with non-linear functions, to a new higher-dimensional space R^b, $b \gg d$, where classes can be distinguished with linear functions :

$$\mathbf{x} = (x_1, x_2, \ldots, x_n) \xrightarrow{\Phi} \mathbf{z} = (z_1, z_2, \ldots, z_n)$$

R^d

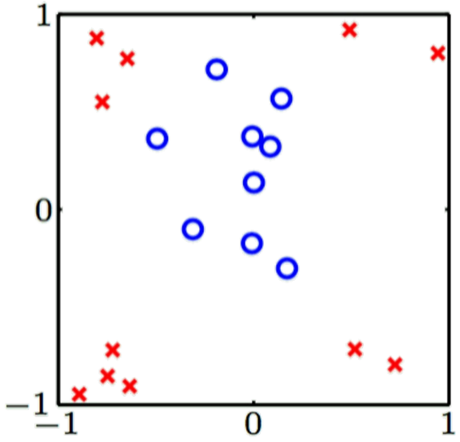




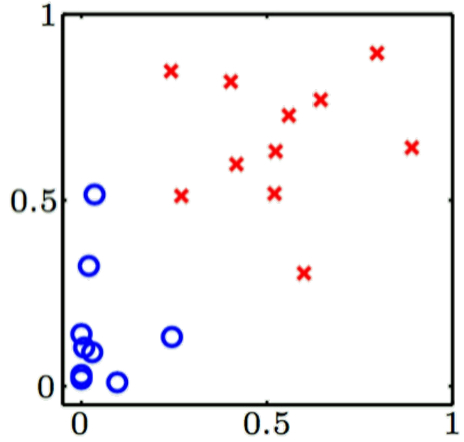
R^b, $b \gg d$

Kernel SVM

1. Original data
$\mathbf{x}^{(j)} \in \mathrm{R}^d$

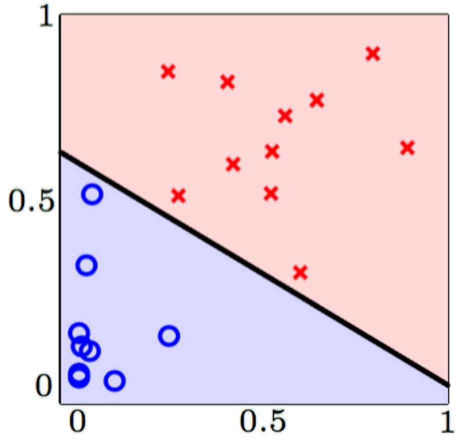


$\xrightarrow{\Phi}$



2. Transform the data
$\mathbf{z}^{(j)} = \Phi\left(\mathbf{x}^{(j)}\right) \in \mathrm{R}^b$, b ≫ d

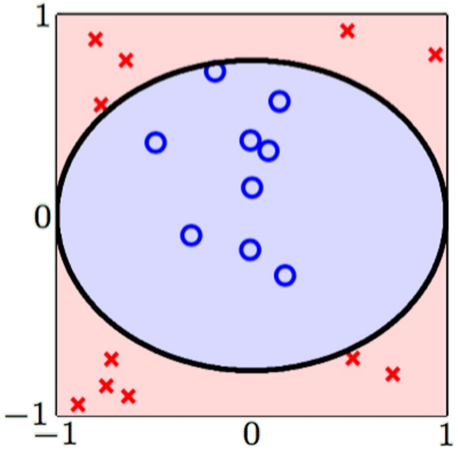
$\downarrow$



3. Separate the data in
R^b space

$$f_{\tilde{\boldsymbol{\theta}}}(\mathbf{z}) = \mathrm{sign}\left(\tilde{\boldsymbol{\theta}}^\top \mathbf{z}\right)$$
$$= \mathrm{sign}(\tilde{\boldsymbol{\theta}}^\top \Phi(\mathbf{x}))$$

$\xleftarrow{\Phi^{-1}}$



4. Classify in R^d space
$\mathbf{x}^{(j)} = \Phi^{-1}\left(\mathbf{z}^{(j)}\right) \in \mathrm{R}^d$

Kernel SVM

- Kernel trick
  - Processing data points $\Phi(\mathbf{x})$ in the higher-dimensional space R^b is time and memory consuming. Kernel trick avoids computing $\Phi(\mathbf{x}) \in R^b$.
  - Instead, we will compute kernel value $K(\mathbf{x},\mathbf{y})$ in R^d with $d \ll b$.
  - Different kernels exist such as polynomial/Gaussian kernels : $K(\mathbf{x},\mathbf{y})=(1+\mathbf{x}^T\mathbf{y})^p$, $K(\mathbf{x},\mathbf{y})=\exp(-\mathbf{x}^T\mathbf{y}/\sigma)$
  - Kernel SVM solutions are computed by solving a quadratic optimization problem :

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T\mathbf{x} \overset{\Phi}{\rightarrow} f_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T\Phi(\mathbf{x})$$

$$\boldsymbol{\theta} = \sum_{j=1}^{n} \alpha^{(j)} y^{(j)} \mathbf{x}^{(j)} \overset{\Phi}{\rightarrow} \boldsymbol{\theta} = \sum_{j=1}^{n} \alpha^{(j)} y^{(j)} \Phi(\mathbf{x}^{(j)})$$

Primal variable

Dual variable

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T\Phi(\mathbf{x}) = \sum_{j=1}^{n} \alpha^{(j)} y^{(j)} \Phi(\mathbf{x}^{(j)})^T\Phi(\mathbf{x})$$

with $\min_{0\leq\boldsymbol{\alpha}\leq C} \boldsymbol{\alpha}^T\mathbf{Q}\boldsymbol{\alpha} - \boldsymbol{\alpha}^T\mathbf{1}$ such that $\boldsymbol{\alpha}^T\mathbf{y} = 0$, $\mathbf{Q} = \mathbf{YKY}$, $\mathbf{Y}=\mathrm{diag}(\mathbf{y})$, $\mathbf{K}(\mathbf{x},\mathbf{y})= \Phi(\mathbf{x})^T\Phi(\mathbf{y})$

[proof not included -- it will not be assessed]

[Optional] Proof given in http://image.diku.dk/imagecanon/material/cortes_vapnik95.pdf

Outline

Convention

- In machine learning, linear models are linear functions w.r.t. data features x, not the parameters $\boldsymbol{\theta}$.
- Remark 1: Mathematically, a linear function is a function that satisfies the additivity property, i.e. $f(x+y) = f(x)+f(y)$. For example, $f_{\boldsymbol{\theta}}(x)=\theta_1 x_1+\theta_2 x_2$ is a linear function w.r.t. $x =(x_1,x_2)$.
- However, the affine function $f_{\boldsymbol{\theta}}(x)=\theta_1 x_1+\theta_2 x_2+\theta_0$ is not a linear function, but we make an abuse of notation and we still refer to the affine function as a "linear" model, but it is *not* true mathematically.
- Remark 2: Another abuse of notation is to disguise the affine function as a linear function as follows: $f_{\boldsymbol{\theta}}(x)=\theta_1 x_1+\theta_0= \boldsymbol{\theta}^T x$ with $\boldsymbol{\theta}=(\theta_1 , \theta_0)$ and $x=(x_1,1)$. So it seems like a linear function w.r.t. x but it is not because it does not satisfy the additivity property, i.e. $f(x+y) = f(x)+f(y)$.
- The same abuse can be used to pretend to represent quadratic function as "linear" function, i.e. $f_{\boldsymbol{\theta}}(x)= \theta_1 x+\theta_2 x^2+\theta_0= \boldsymbol{\theta}^T x$, where $\boldsymbol{\theta}=(\theta_2 ,\theta_1 ,\theta_0)$ and $x=(x^2, x, 1)$. Again, the representation seems linear w.r.t. x but mathematically it is not.

Conclusion

- Linear models can be used for classification and regression tasks.
- Very well-established techniques, which fit optimally CPU/GPU acceleration hardware.
  - BLAS/LAPACK for CPU and CUDA for GPU
- But very limited expressivity, only perform well for linearly separable data points.
- Kernel SVM enhances their expressivity with non-linear separators but requires to hand-craft a kernel operator.
  - Deep learning has significantly surpassed SVM techniques.

References

- Prof Min-Yen Kan, CS3244 NUS, Machine Learning, 2022
  - https://knmnyn.github.io/cs3244-2210
- Prof Xavier Bresson, CS6208 NUS, Advanced Topics in Artificial Intelligence, 2023

Questions?