

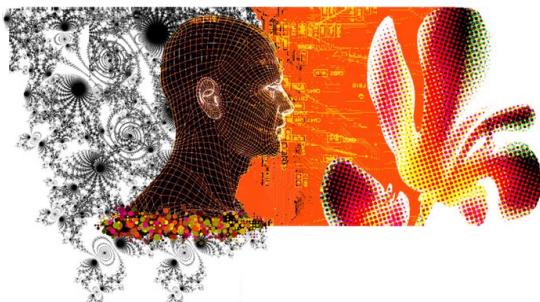
CS3244 : Machine Learning

Semester 1 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)



Material used for preparation

- 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

Outline

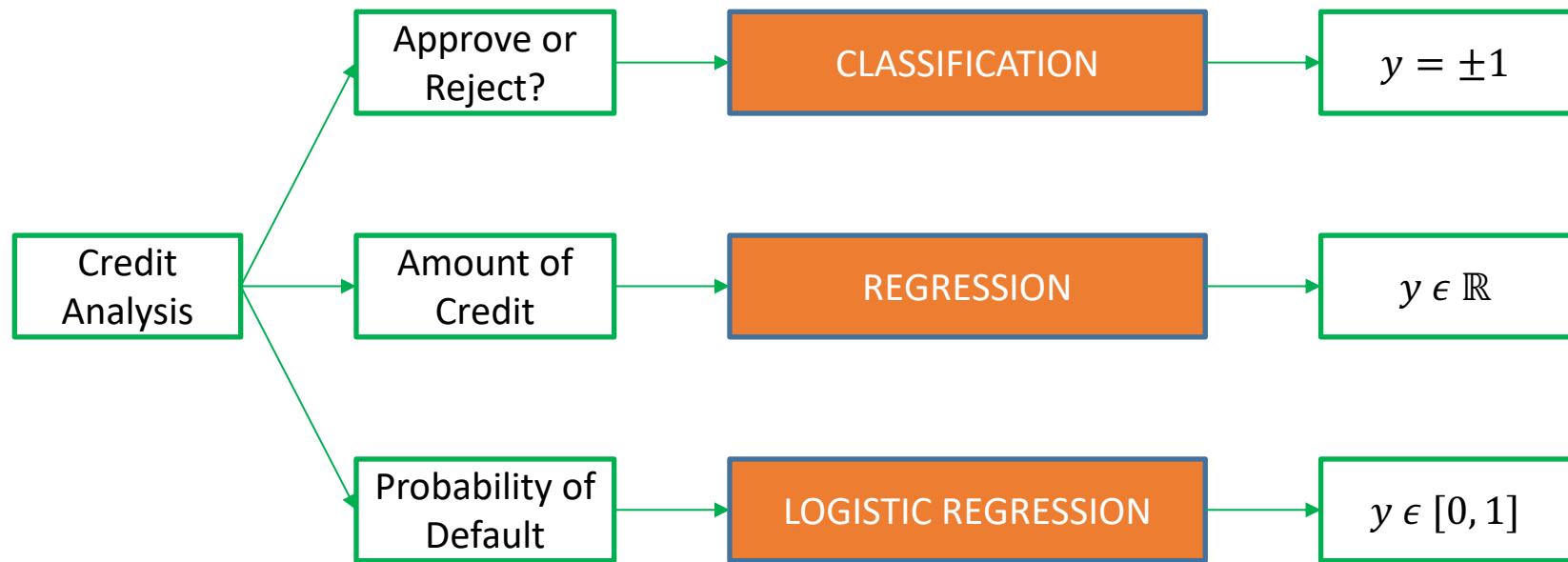
- Three applications of linear models
- Classification
- Regression
- Normal equations
- Logistic regression
- Gradient descent
- Support Vector Machine
- Soft-margin SVM
- Kernel SVM
- Conclusion

Outline

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

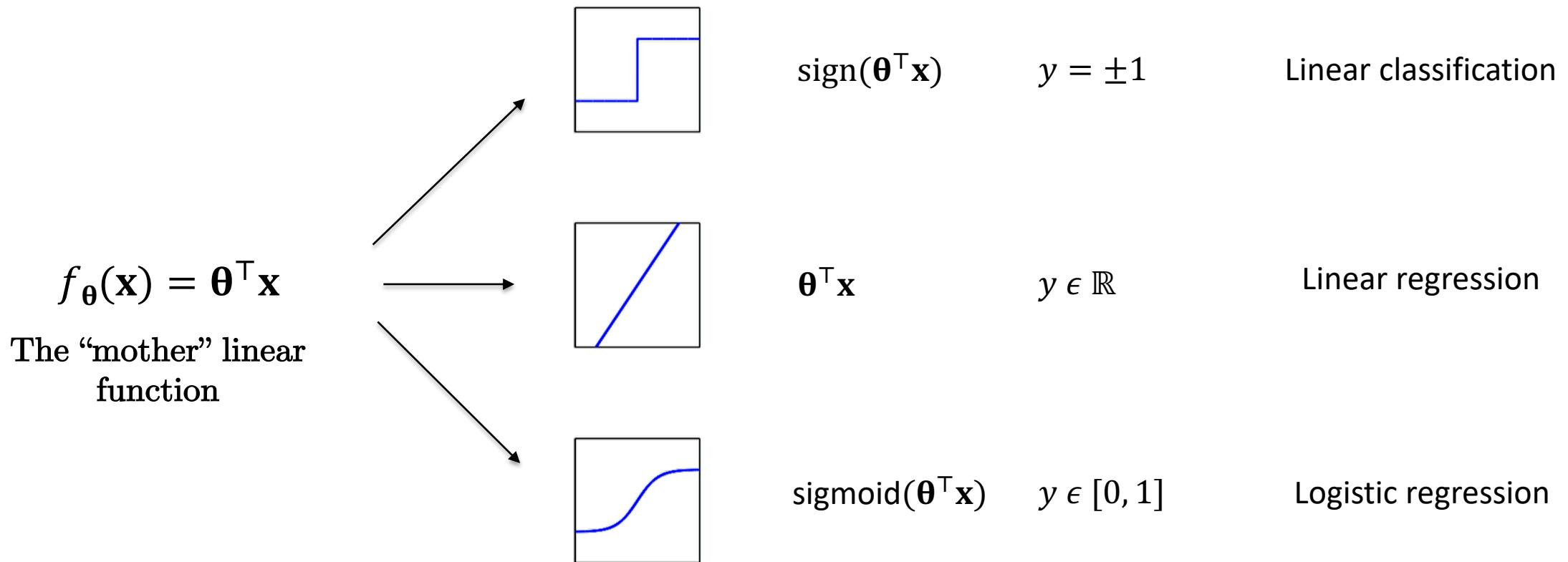
Three learning problems

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



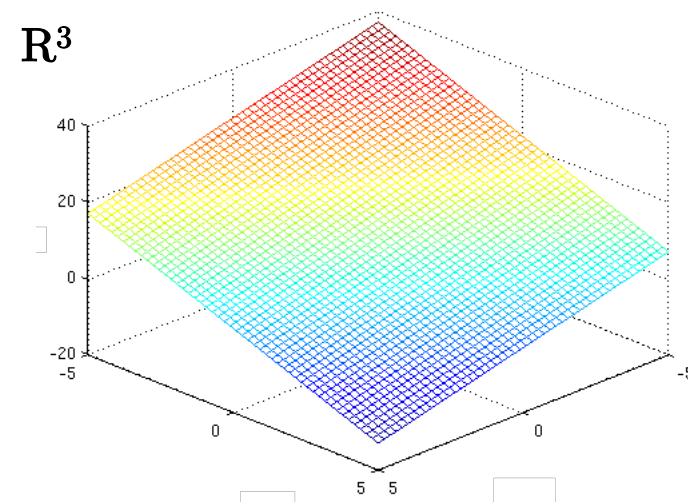
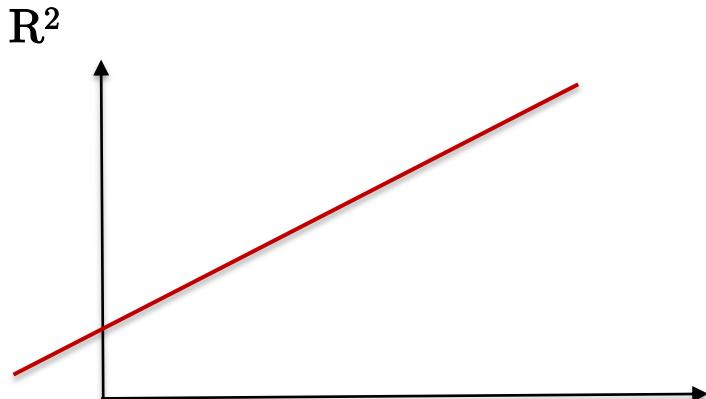
Three learning problems

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



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 (>3 D spaces).
- Parameters of linear models are the slopes and the bias.



Outline

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

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.

$$f_{\theta}(x) = \theta^T x = \sum_{i=1}^d \theta_i x_i$$

Model parameters
Weight/importance of feature x_i

$> 0 \Rightarrow \text{Approve}$
 $< 0 \Rightarrow \text{Reject}$

Decision by thresholding
(binary classification task)

Criterion	Value
Age	32 years
Salary	40 K
Debt	26 K
...	...
Years in Job	1 year
Years at Current Residence	3 years

 x_1 x_2 x_d

$$f_{\theta}(x) = \theta^T x = \theta_d x_d + \dots + \theta_2 x_2 + \theta_1 x_1 + \theta_0$$

Slopes of hyper-plane

bias (offset)

Example

- Extend credit using applicant's age
 - If age is higher than 25 years, accept credit.
 - Otherwise, reject.
- Linear model : $f_{\theta}(x) = x_1 - 25 > 0 \Rightarrow \text{Approve}$
 $< 0 \Rightarrow \text{Reject}$
With $f_{\theta}(x) = \theta^T x = \theta_1 x_1 + \theta_0$, $\theta = (\theta_1, \theta_0) = (1, -25)$ and $x = (x_1, 1)$

- Example :

Criterion	Value
Age	32 years

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

Binary classification

- Binarize a linear function :

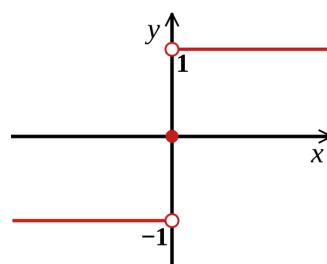
For a data input $\mathbf{x} = (x_1, x_2, \dots, x_n)$ (attributes of a customer)

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

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

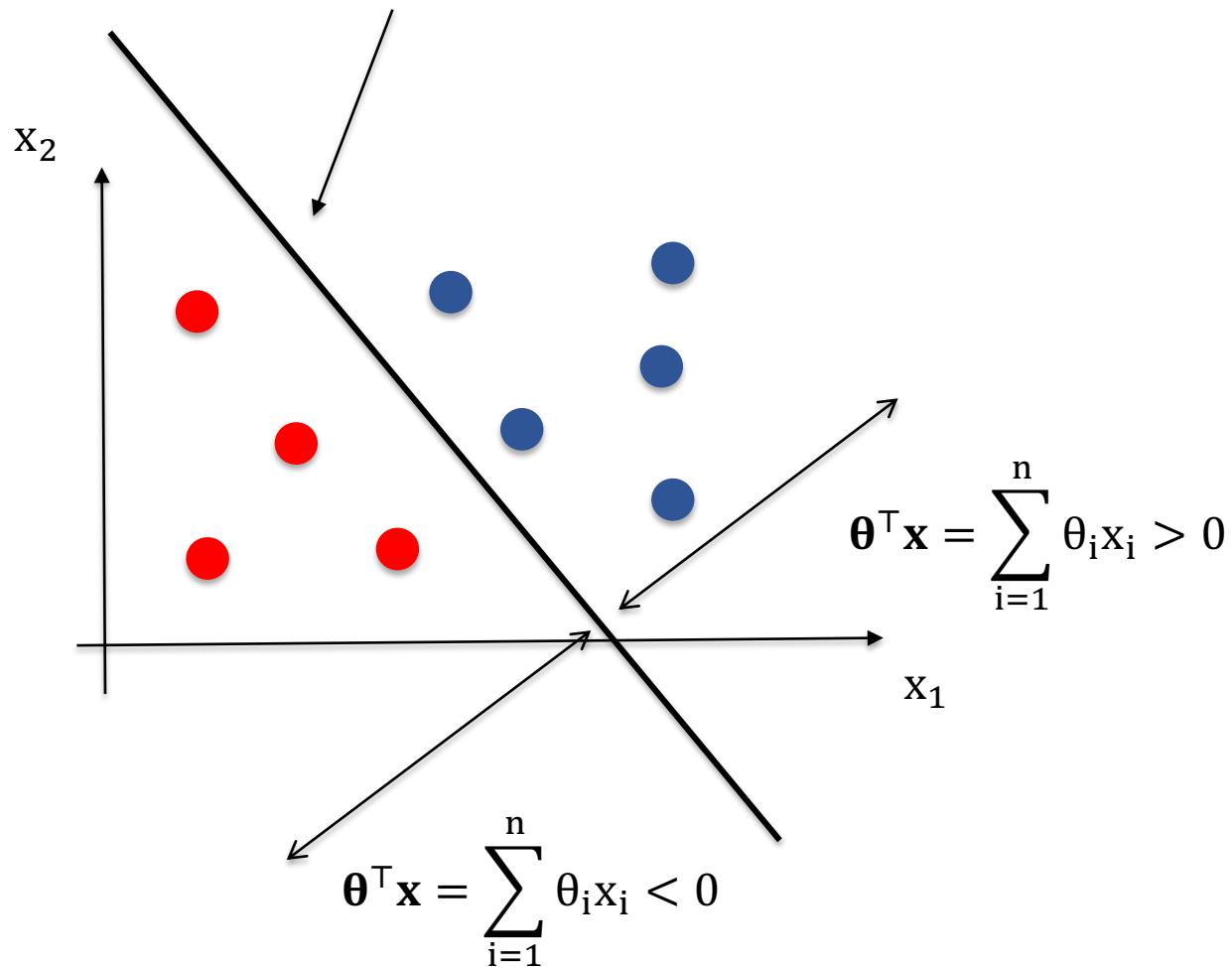
The linear formula f can be written as:

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

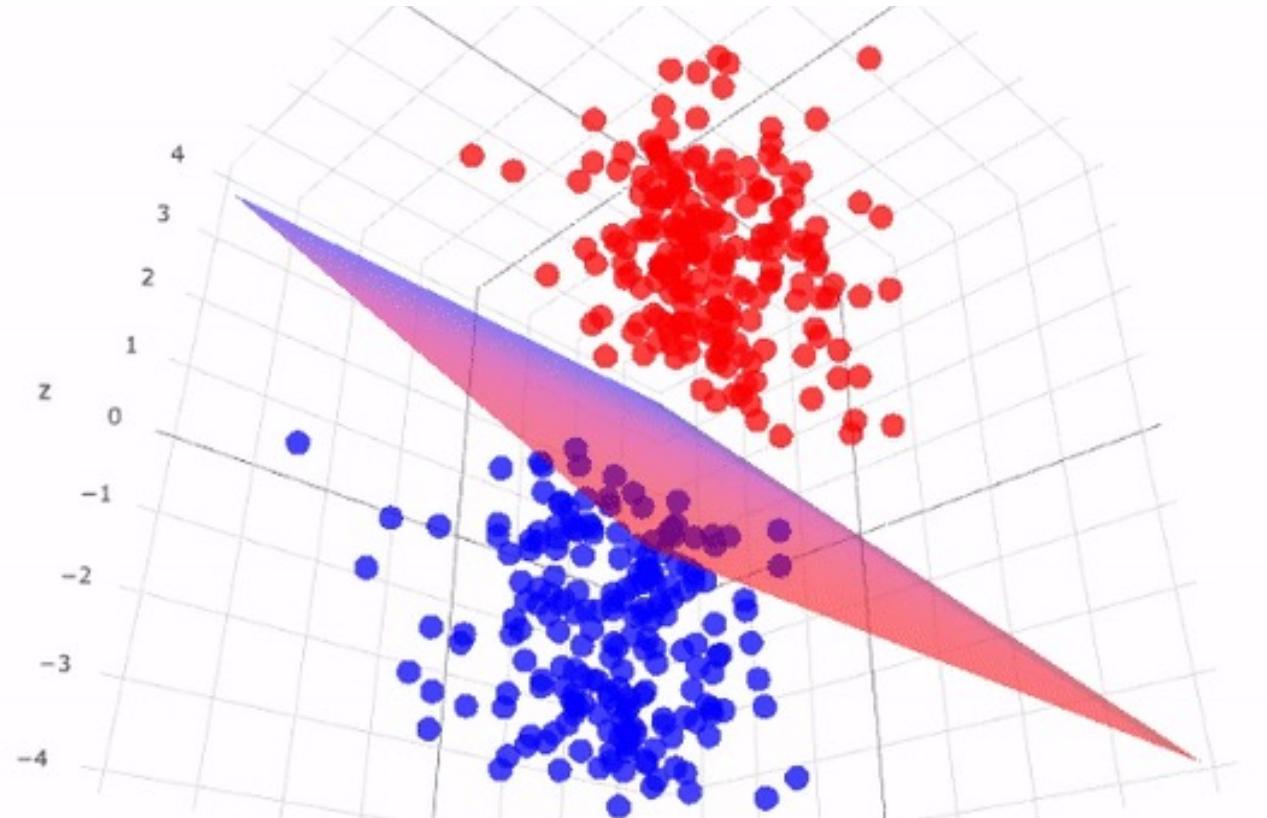


Decision boundary in 2D

Decision boundary
Defined by all points $\mathbf{x} \in \mathbb{R}^d : \boldsymbol{\theta}^\top \mathbf{x} = 0$



Decision boundary in 3D



Outline

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

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.

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

Decision by weighted sum of data features

Criterion	Value
Age	32 years
Salary	40 K
Debt	26 K
...	...
Years in Job	1 year
Years at Current Residence	3 years

x_1

x_2

x_d

Slopes of hyper-plane

$$f_{\theta}(x) = \theta^T x = \theta_d x_d + \dots + \theta_2 x_2 + \theta_1 x_1 + \theta_0$$

bias (offset)

Loss function

- Quality of a predictive function for a training set is evaluated with a loss function.
 - Training set : Collected/historical data $(\mathbf{X}, \mathbf{y}) = (\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots, (\mathbf{x}^{(n)}, y^{(n)})$ where \mathbf{X} is a $n \times d$ 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 y :

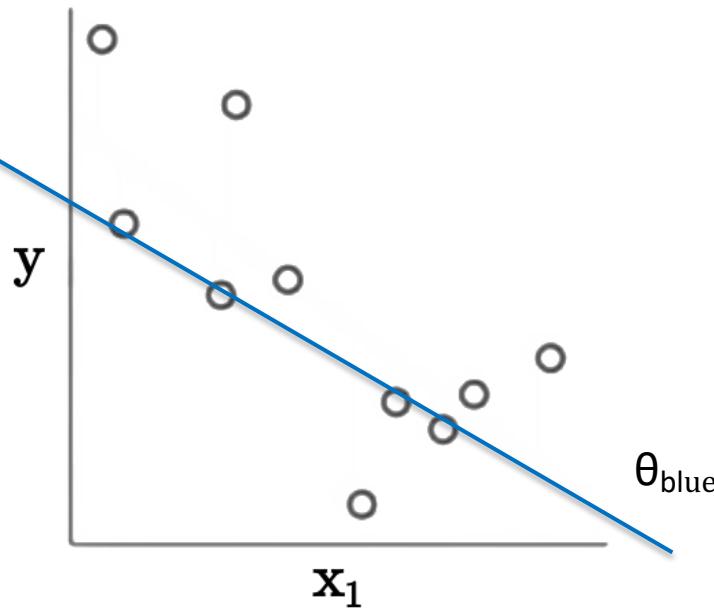
$$L(\theta) = \frac{1}{n} \sum_{j=1}^n (f_{\theta}(x^{(j)}) - y^{(j)})^2$$

How good is the prediction

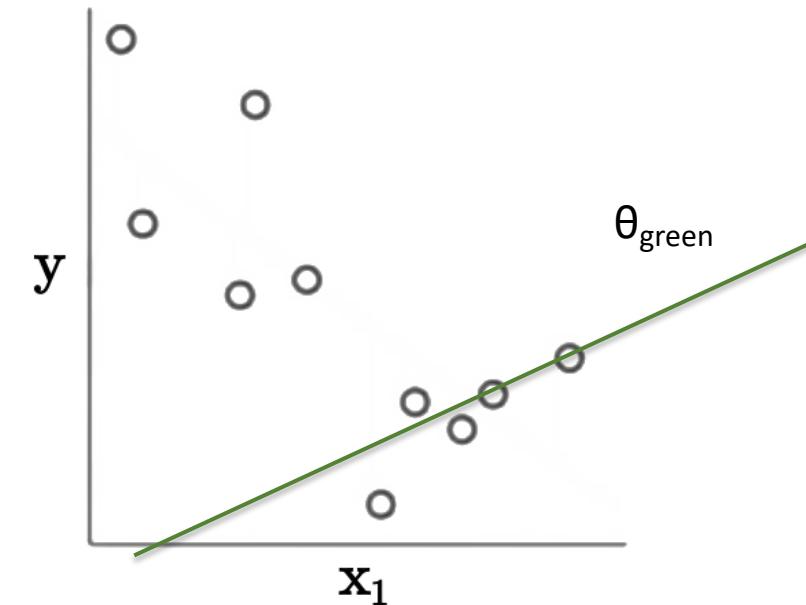
Mean square error
(MSE)

Illustration

- The training set is composed of n data points, each data feature is 1-dimensional, i.e. $\mathbf{x} = \mathbf{x}_1$ and the label is y .
- Find the linear model θ that fits all the data points as best as possible.



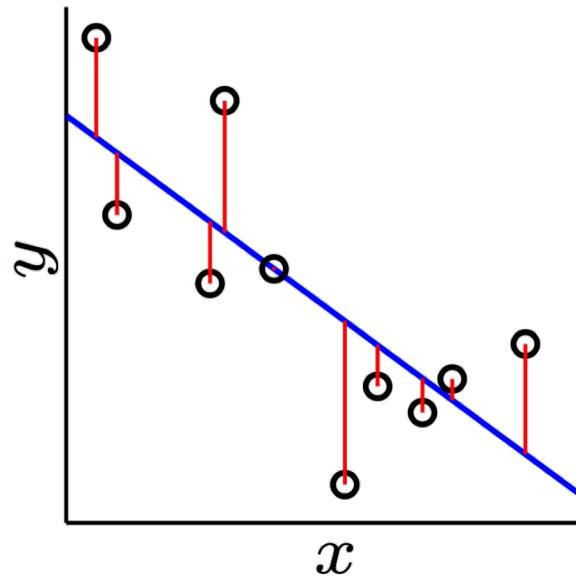
Hypothesis function : $f_{\theta}(\mathbf{x}) = \theta_0 + \theta_1 x_1$



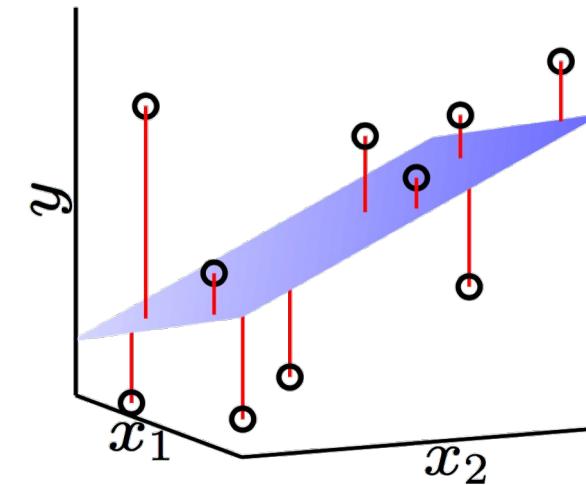
Loss value: $L(\theta_{\text{green}}) \geq L(\theta_{\text{blue}})$

Illustration

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



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



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

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

Outline

- 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?

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

- Two approaches
 - Normal equations
 - Gradient descent

Normal equations

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

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

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

Normal equations

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

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

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

Normal equations

- Expand L :

$$\begin{aligned} L &= \frac{1}{n} \|X\theta - y\|^2 \\ &= \frac{1}{n} (X\theta - y)^\top (X\theta - y) = \frac{1}{n} ((X\theta)^\top - y^\top)(X\theta - y) \\ &= \frac{1}{n} ((X\theta)^\top X\theta - (X\theta)^\top y - y^\top X\theta + y^\top y) \\ &= \frac{1}{n} (\theta^\top X^\top X\theta - 2\theta^\top X^\top y + y^\top y), \text{ with } \theta^\top X^\top y = (X\theta)^\top y = y^\top X\theta \end{aligned}$$

- Compute gradient (with matrix calculus) and get solution :

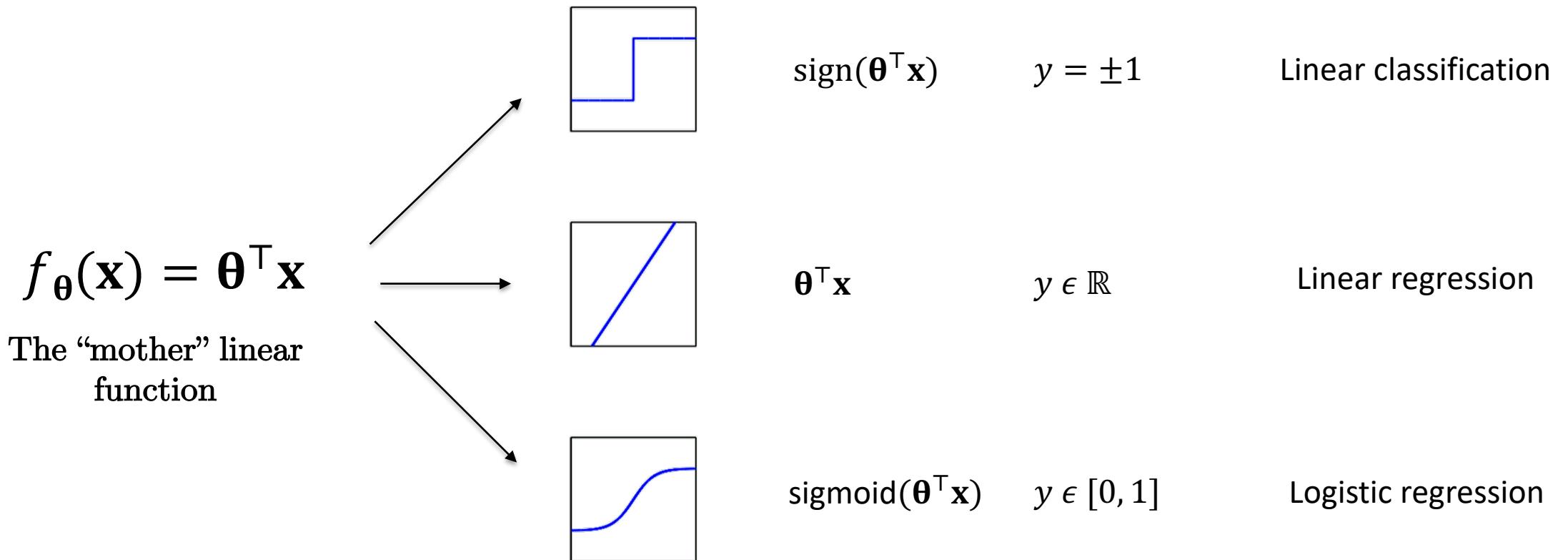
$$\begin{aligned} \frac{\partial L}{\partial \theta} &= \frac{1}{n} (2 \cdot X^\top X\theta - 2 \cdot X^\top y) = \mathbf{0} \\ \Rightarrow X^\top X\theta &= X^\top y \Rightarrow \theta = \underbrace{((X^\top X)^{-1} X^\top)}_{\text{Pseudo-inverse of } X, \text{ a.k.a. } X^\dagger} y = X^\dagger y \end{aligned}$$

Pseudo-inverse of X , a.k.a. X^\dagger
Computationally expensive to
compute if n or m is large.

Outline

- 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



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. $f_{\theta}(x)$ is interpreted as a probability of class $\in [0,1]$.
- Example
 - Prediction of a heart attack
 - Input x : Cholesterol level, age, diabetic, etc.
 - Output $f_{\theta}(x)$: Likelihood of a heart attack $\in [0,1]$
 - $\theta^T x \in \mathbb{R}$ is the risk score.

Logistic regression

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

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



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

Logistic regression

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

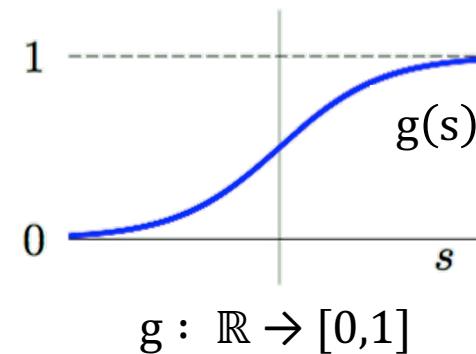
$$f_{\theta}(x) = \text{sigmoid}(\theta^T x) \in [0,1]$$

- Choice of logistic function :

- Function $\text{sigmoid}(\cdot) = g(\cdot)$ is a smooth step function or soft thresholding function.

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

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



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

Logistic regression

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

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

↗
Number of
training data

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

Explained in next slides

Logistic regression

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

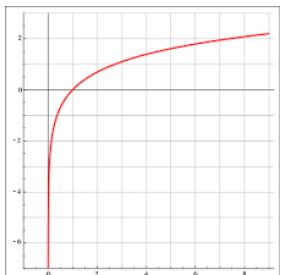
Sigmoid

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

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

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

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



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

Monotonous
increasing function

Logistic regression

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

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

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

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

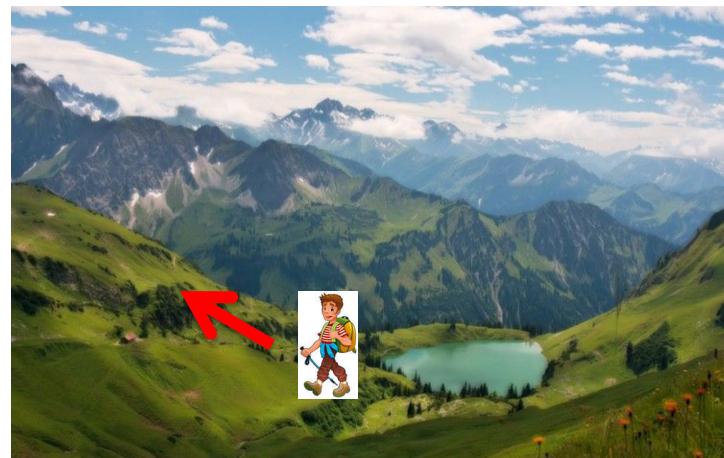
$$\log \prod_{j=1}^n = \sum_{j=1}^n \log$$

Outline

- 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) :

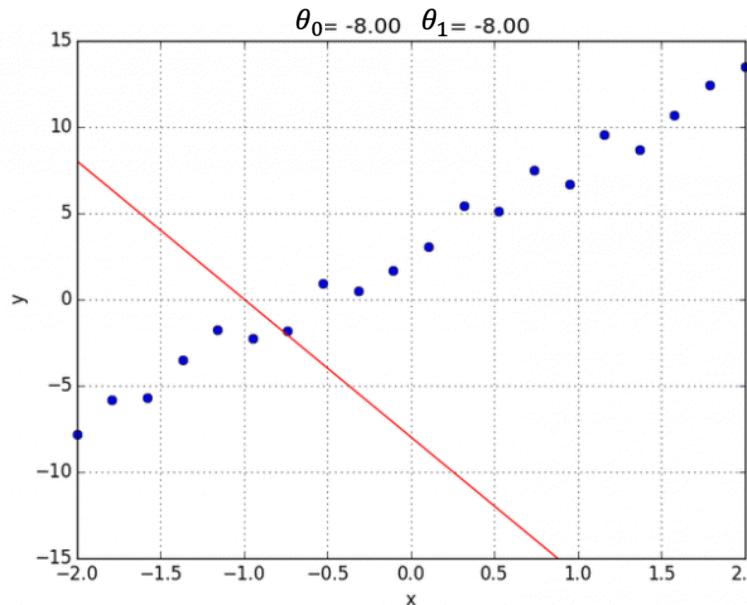
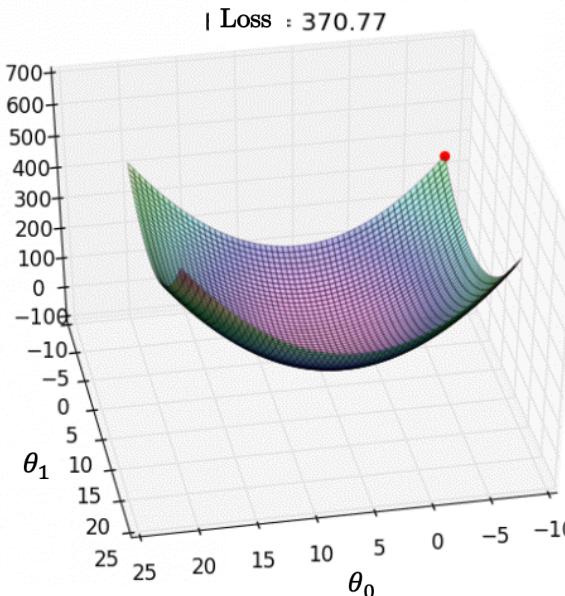
- Prediction function : $f_{\theta}(x) = \theta_0 + \theta_1 x$

- 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):

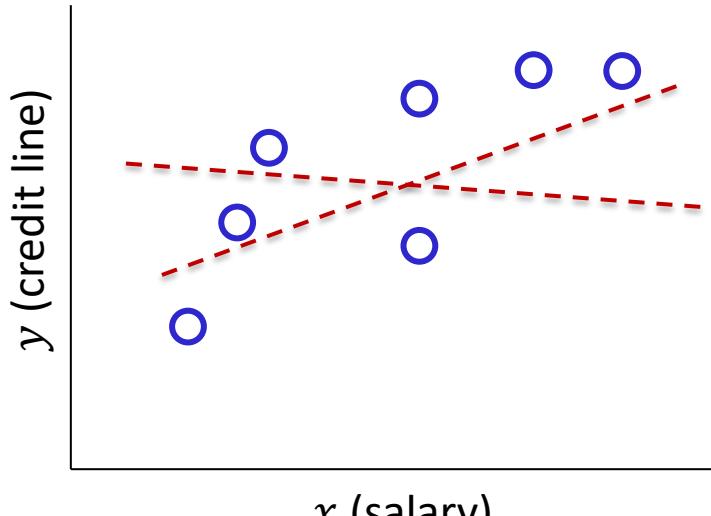
$$\min_{\theta} L(\theta = (\theta_0, \theta_1))$$



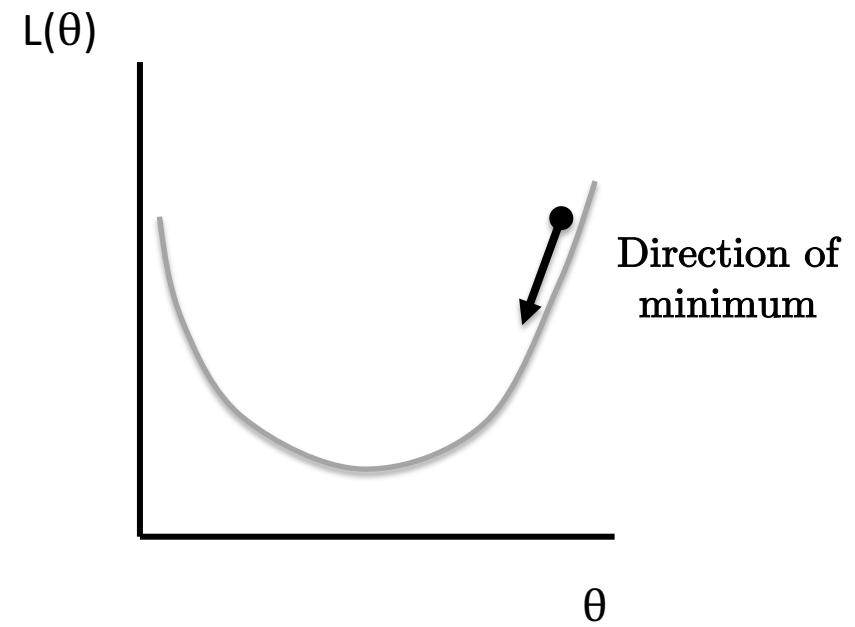
$$f_{\theta}(x) = \theta_0 + \theta_1 x$$

Gradient descent

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

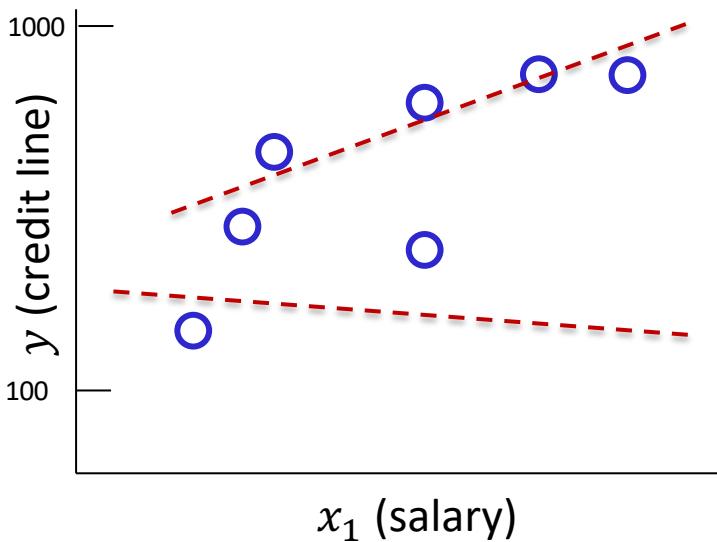


$$f_\theta(\mathbf{x}) = \theta x$$

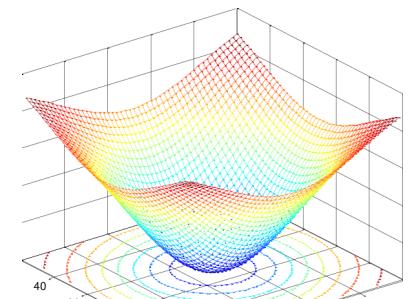
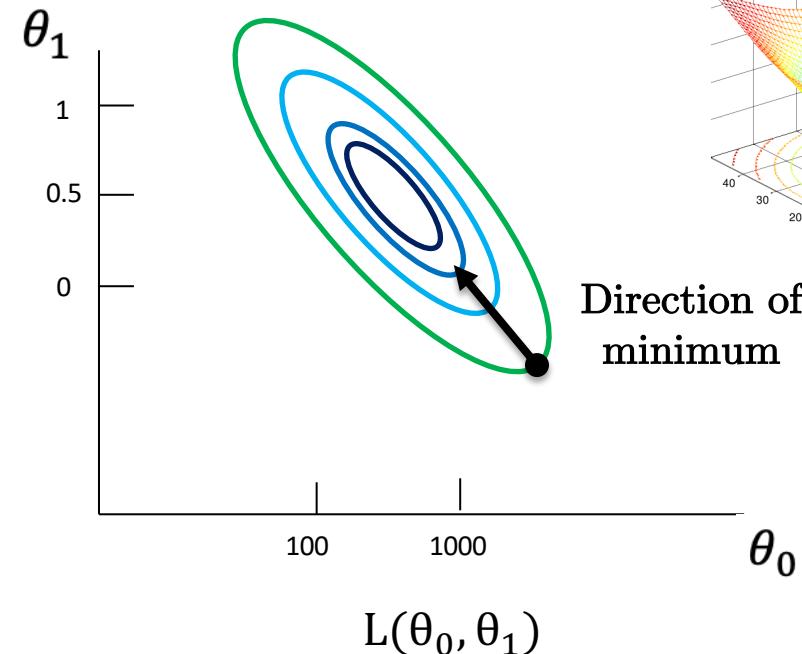


Gradient descent

- 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. $L(\theta_0, \theta_1) = (\theta_0 + \theta_1 x - y)^2$



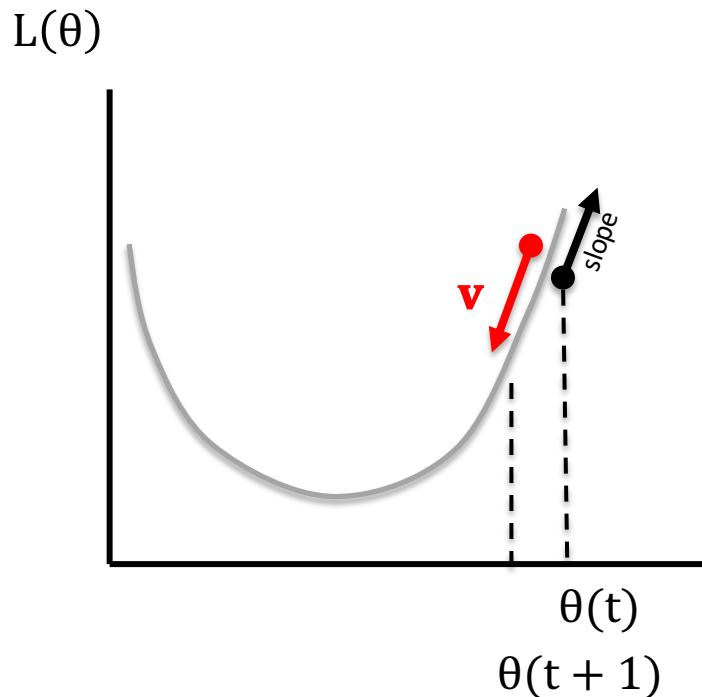
$$f_{\theta}(x) = \theta_0 + \theta_1 x$$



Direction of
minimum

Gradient descent

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



Gradient descent

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

First-order Taylor approximation

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

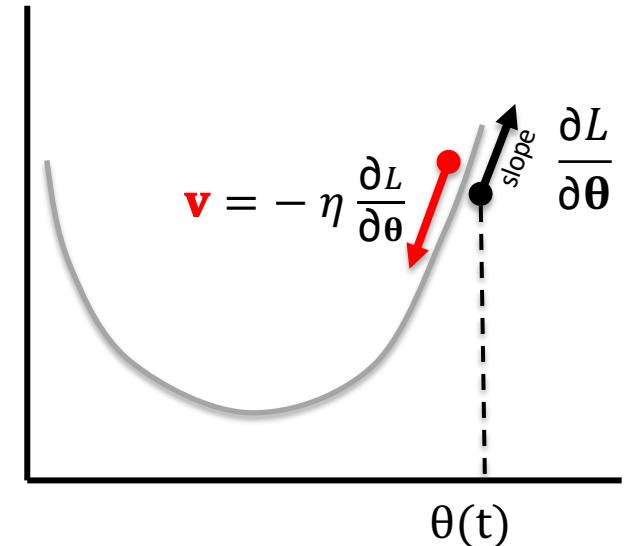
$$\Delta L = L(\theta + \Delta\theta) - L(\theta) \approx \left\langle \frac{\partial L}{\partial \theta}, \Delta\theta \right\rangle$$

$$\min_{\Delta\theta} \Delta L \Rightarrow \Delta\theta = -\eta \frac{\partial L}{\partial \theta}, \quad \eta > 0$$

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

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

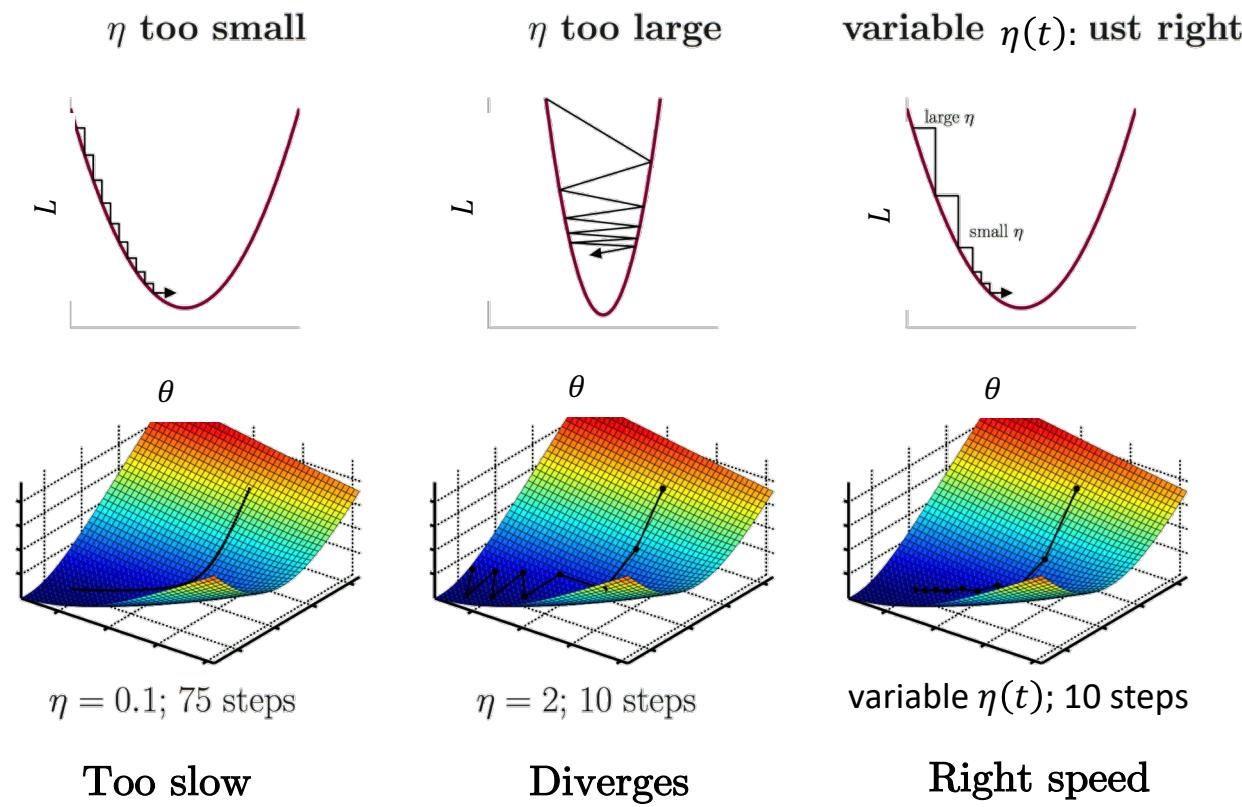
$$L(\theta)$$



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

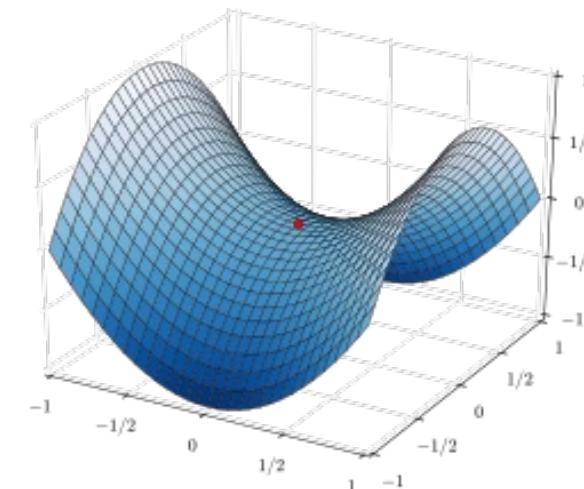
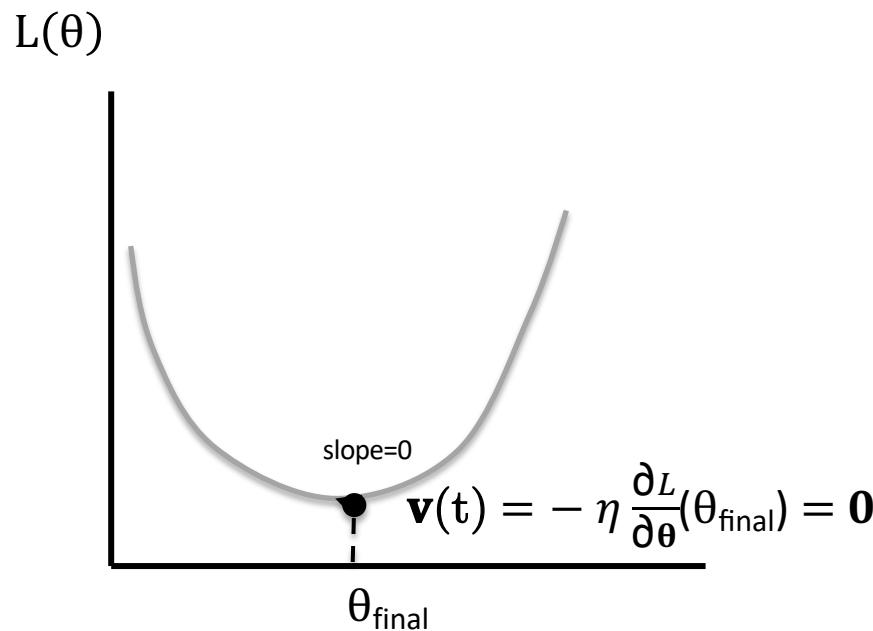
Gradient descent

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



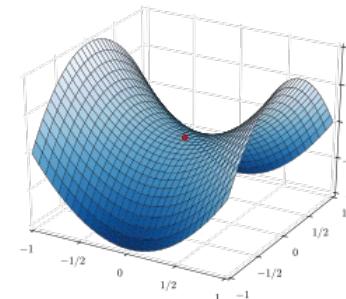
Gradient descent

- 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.



Stochastic gradient descent

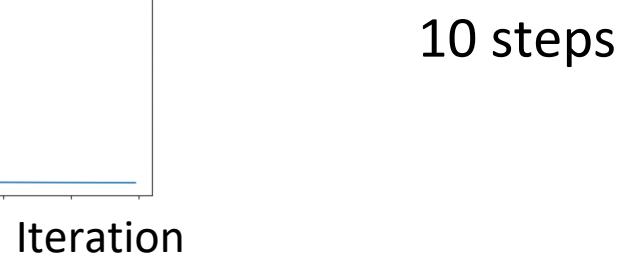
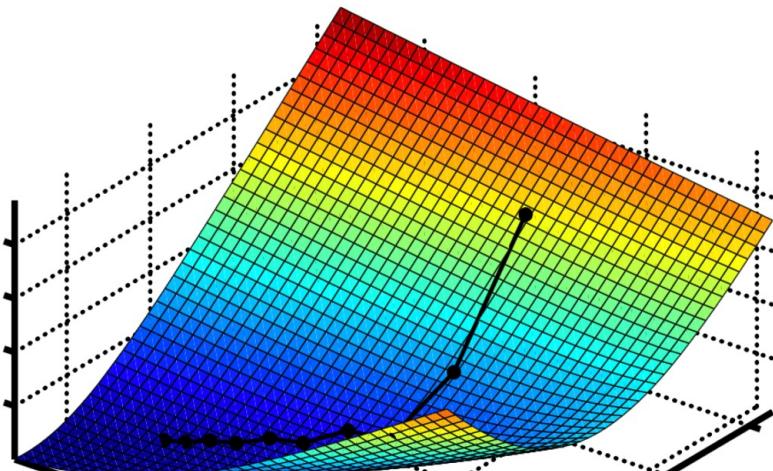
- A fast variation of gradient descent that considers only a small set of data points to update the value of the parameters is stochastic gradient descent (SGD):
 - Pick a (random) small subset of m training data (e.g. a single/512 data points), i.e. $(\mathbf{x}^{(k)}, y^{(k)})$
 - Compute the loss 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. $\theta(t + 1) = \theta(t) - \eta \mathbf{v}$
- 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



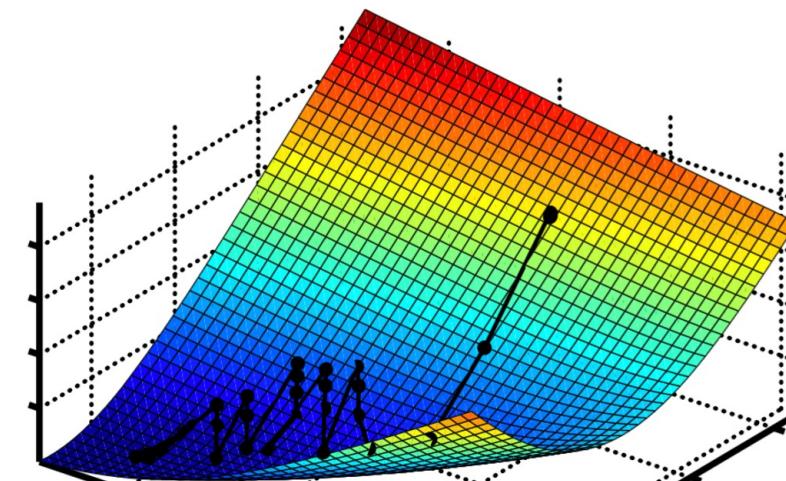
Stochastic gradient descent

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

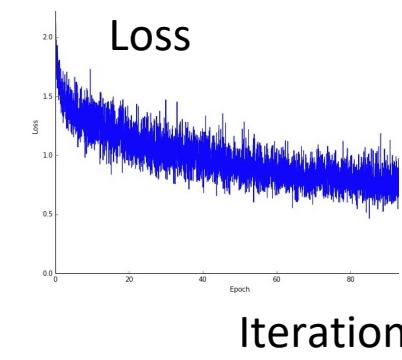
GD



SGD



30 steps



Summary

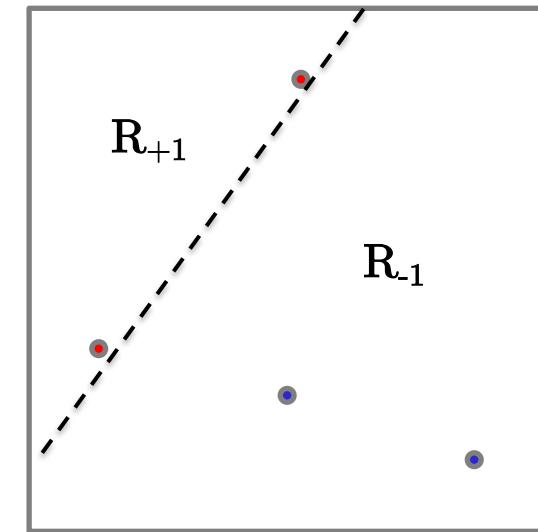
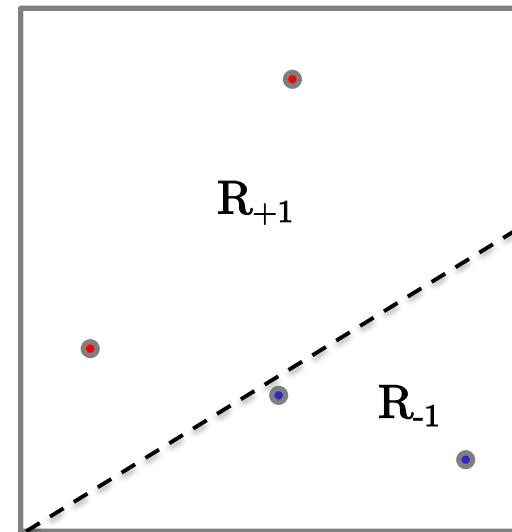
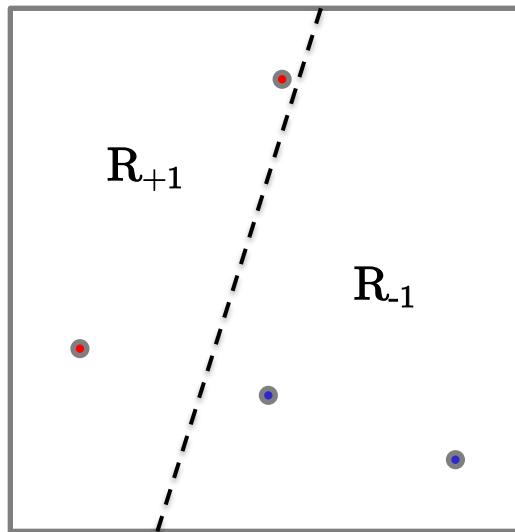
- Two methods for solving the linear regression task :
 - Gradient Descent
 - Works well for very large n , #training data, with SGD (300B tokens with GPT3)
 - Works well for very large d , #data features ($d=1M$ with $1,000 \times 1,000$ images)
 - Works well for very large $|\theta|$, #parameters features (175B with GPT3)
 - Very slow and requires to select time step η
 - Normal Equations
 - Very fast for $n=O(10^6)$ data points, do not require to choose η
 - Need to compute $(\mathbf{X}^T \mathbf{X})^{-1}$, $O(n^3)$ 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

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

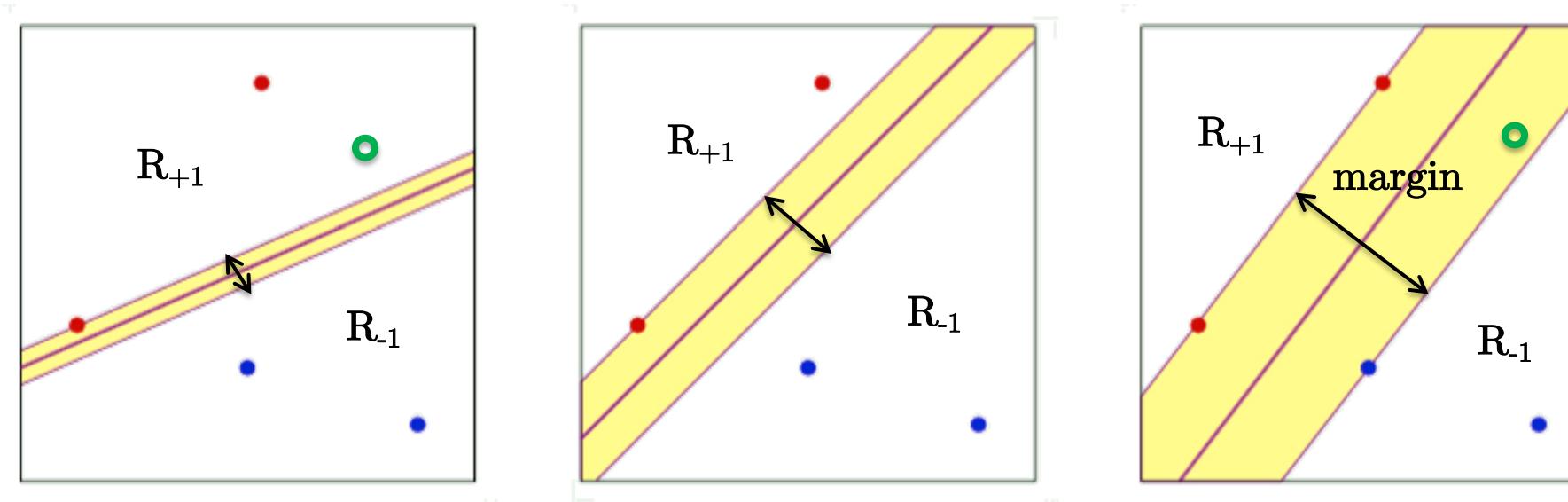
Support Vector Machine

- 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”?



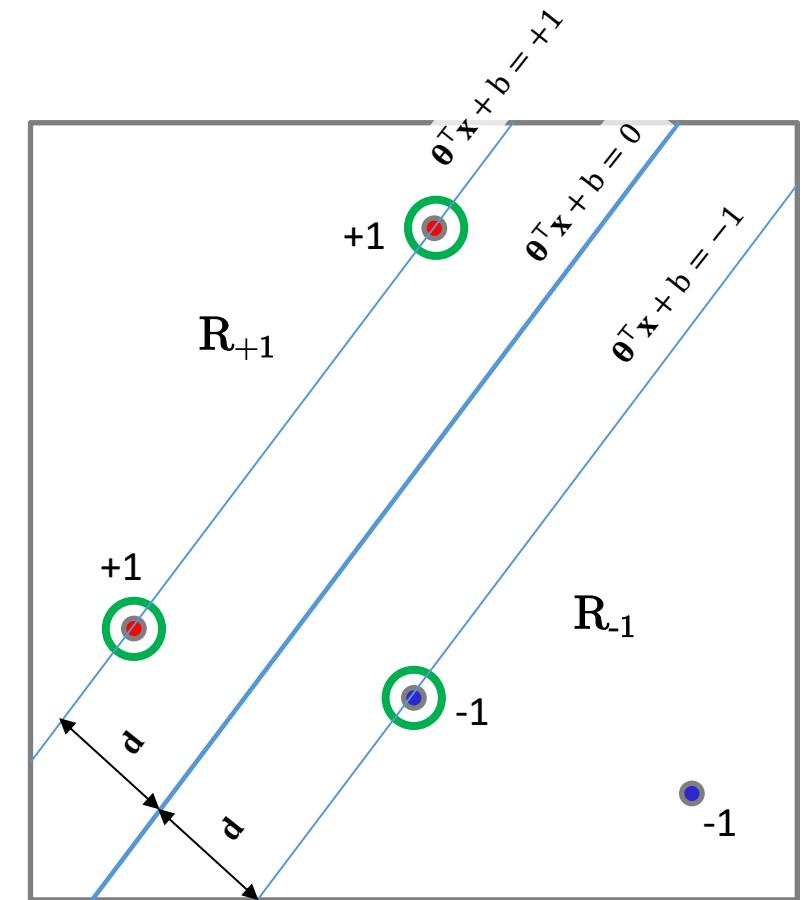
Support Vector Machine

- SVM technique (1964) aims at maximizing the margin between the two classes.
- Maximizing the margin provides better generalization results because of enhanced decision boundary.



Support Vector Machine

- Hyper-plane equation : { \mathbf{x} such that $f_{\theta}(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x} + b = \text{cte}$ }
- Hyper-planes $f_{\theta}(\mathbf{x}) = \pm 1$ are margins.
- Hyper-plane $f_{\theta}(\mathbf{x}) = 0$ is the class separator.
- Training data points $+1$ must 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.
- Margin distance d is computed to be as large as possible.
- What are the parameters $\boldsymbol{\theta}$ that maximizes the margin?
(response next slide)



Support Vector Machine

- What are the parameters θ that maximizes the margin?
- Optimal value θ is solution of a constrained quadratic optimization problem :

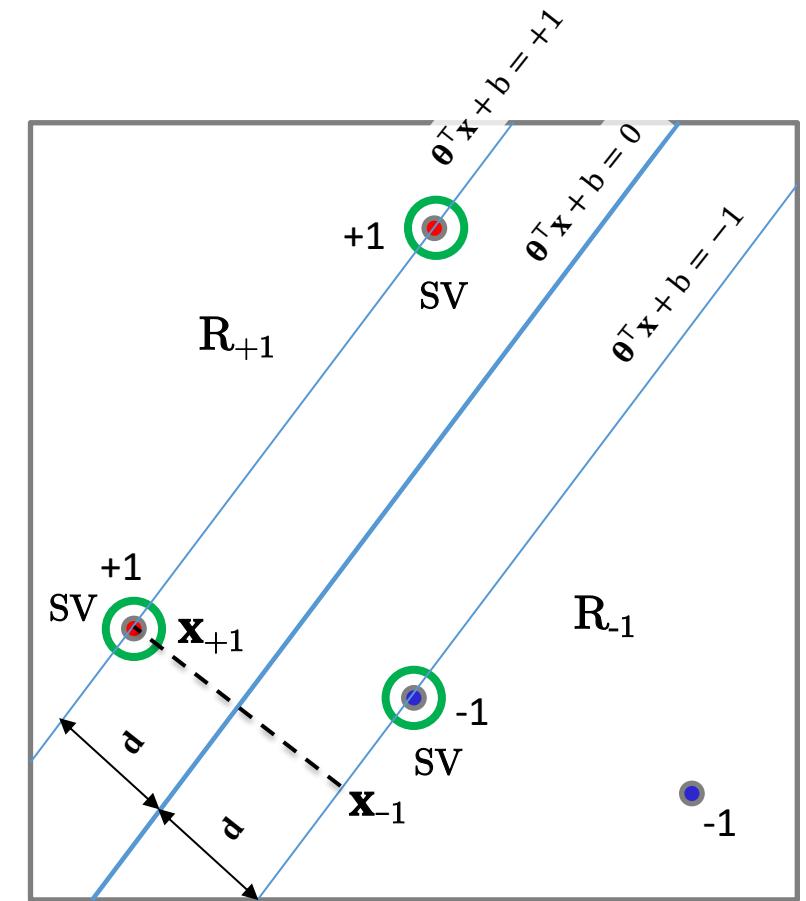
We have $f_{\theta}(\mathbf{x}) = \theta^T \mathbf{x} + b$

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

Margin vector is defined as $\mathbf{d} = \mathbf{x}_{+1} - \mathbf{x}_{-1} \Rightarrow d = \frac{2}{\|\theta\|}$

$$\max_d d = \frac{2}{\|\theta\|} \Leftrightarrow \min_{\theta} \|\theta\|^2 \text{ such that}$$

$$\begin{cases} f_{\theta}(\mathbf{x}) \geq +1 \text{ for } \mathbf{x} \in R_{+1} \\ f_{\theta}(\mathbf{x}) \leq -1 \text{ for } \mathbf{x} \in R_{-1} \end{cases}$$



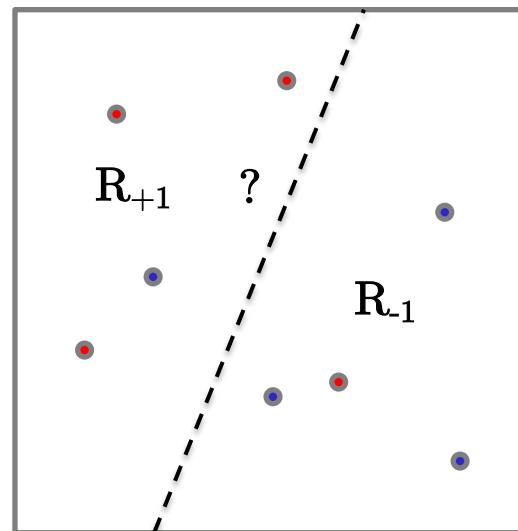
Only a subset of the data points have value $f(\mathbf{x}) = \pm 1$. These are called support vectors (SV).

Outline

- 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

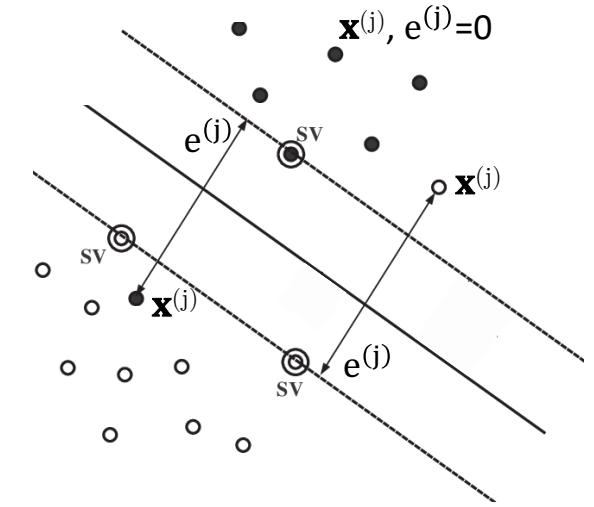
- SVM can be improved to deal with outliers.
 - Soft-margin SVM technique, 1995
 - Idea is to introduce a slack variable $e^{(j)}$ for each data point that represents the prediction error.
 - These errors $e^{(j)}$ will be minimized while simultaneously maximizing the margin :

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



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

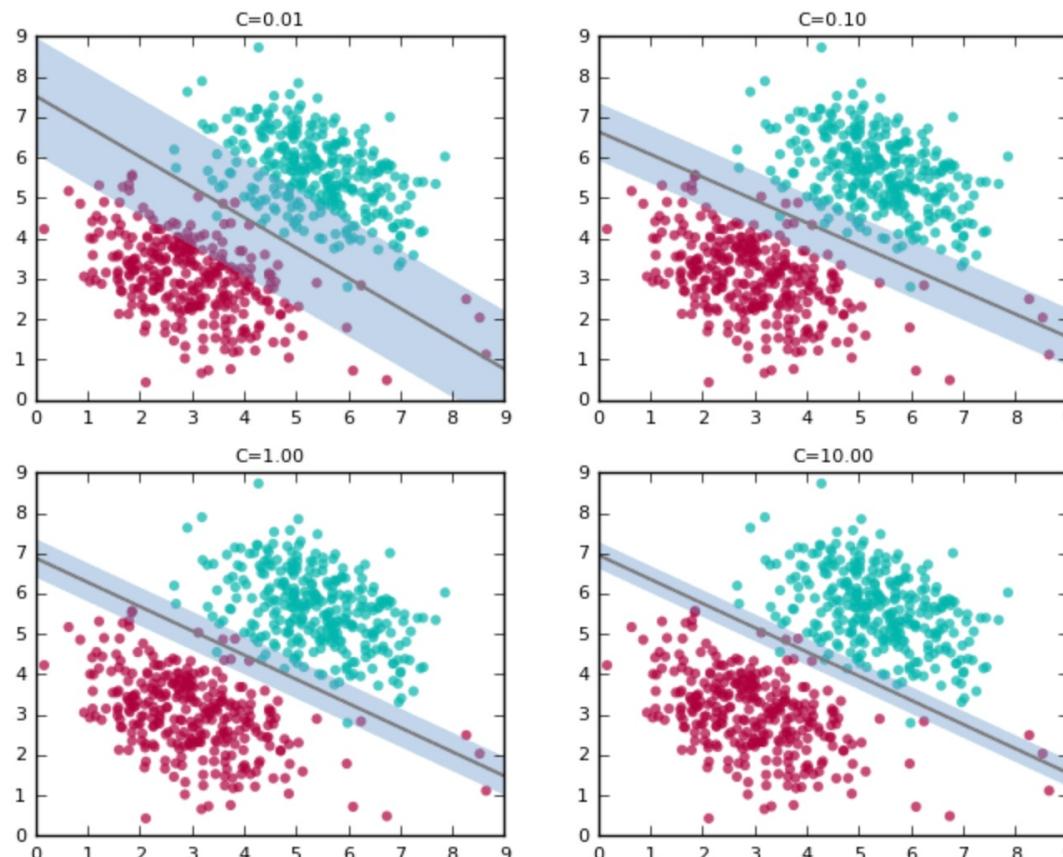
Soft-margin SVM



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

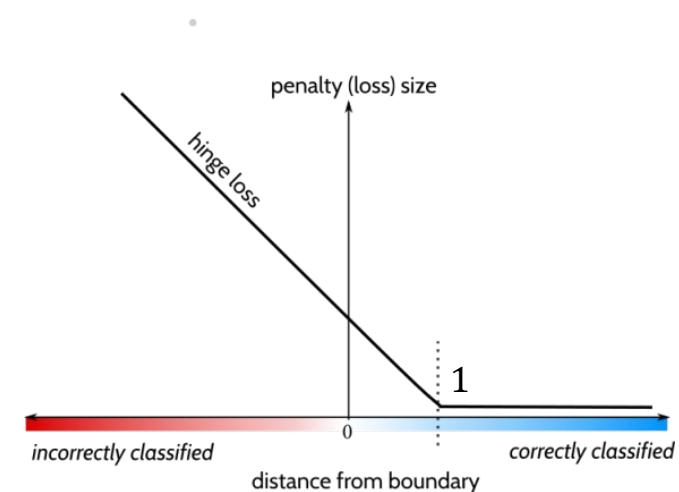
- Soft-margin SVM penalizes
 - Misclassifications,
 - Correct classifications that fall inside the margin.
- Loss of soft-margin : Hinge loss (popular loss function)

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

\Updownarrow (proof not included)

$$\min_{\theta} \|\theta\|^2 + C \cdot \sum_{j=1}^n \max(0, 1 - y^{(j)} f_\theta(\mathbf{x}^{(j)}))$$

$\underbrace{}_{\text{Hinge loss}}$

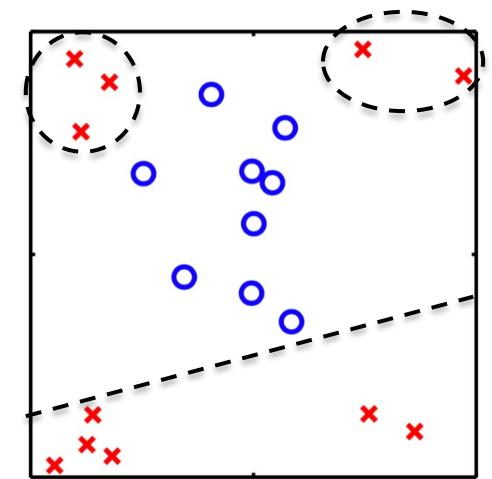


Outline

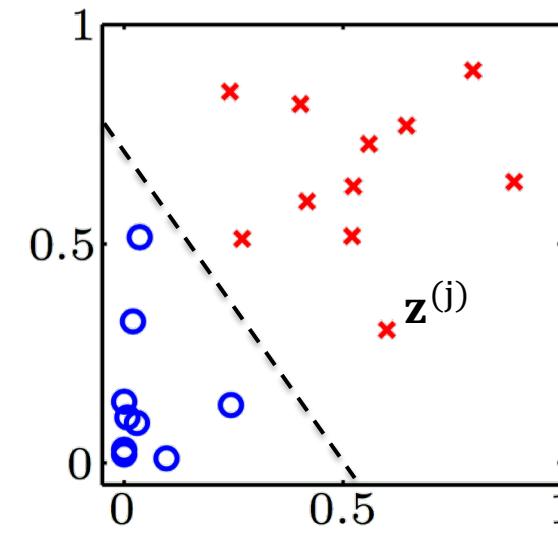
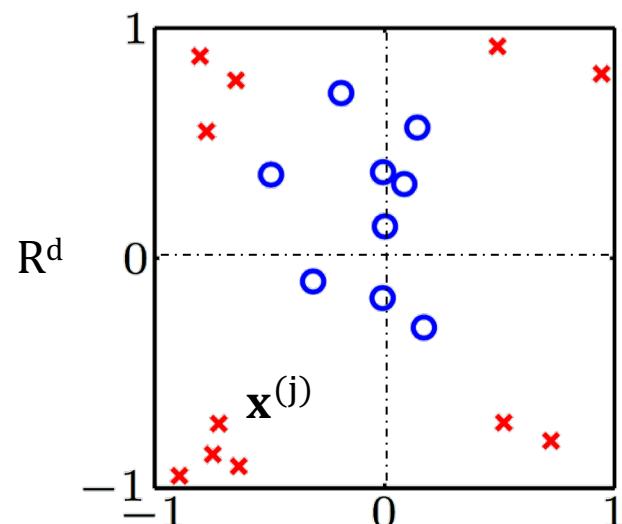
- Three applications of linear models
- Classification
- Regression
- Normal equations
- Logistic regression
- Gradient descent
- Support Vector Machine
- Soft-margin SVM
- **Kernel SVM**
- Conclusion

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 :



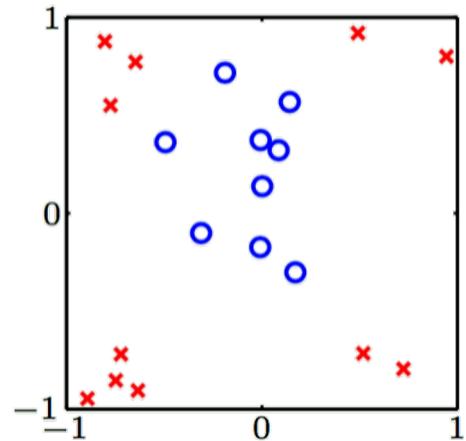
$$\mathbf{x} = (x_1, x_2, \dots, x_n) \xrightarrow{\Phi} \mathbf{z} = (z_1, z_2, \dots, z_n)$$



$R^b, b \gg d$

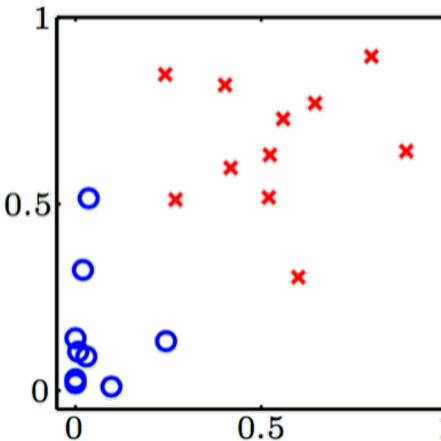
Kernel SVM

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

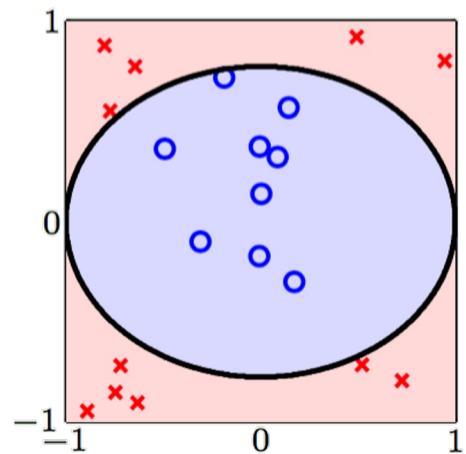


$$\Phi$$

2. Transform the data
 $\mathbf{z}^{(j)} = \Phi(\mathbf{x}^{(j)}) \in \mathbb{R}^b, b \gg d$



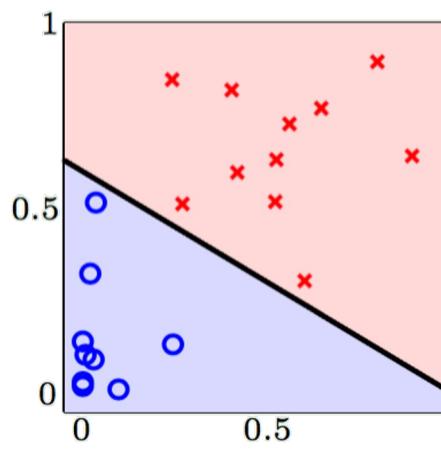
4. Classify in \mathbb{R}^d space
 $\mathbf{x}^{(j)} = \Phi^{-1}(\mathbf{z}^{(j)}) \in \mathbb{R}^d$



$$\Phi^{-1}$$

3. Separate the data in
 \mathbb{R}^b space

$$f_{\tilde{\theta}}(\mathbf{z}) = \text{sign}(\tilde{\theta}^\top \mathbf{z}) \\ = \text{sign}(\tilde{\theta}^\top \Phi(\mathbf{x}))$$



Kernel SVM

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

$$f_{\theta}(x) = \theta^T x \xrightarrow{\Phi} f_{\theta}(x) = \theta^T \Phi(x)$$

$$\Theta = \sum_{j=1}^n \alpha^{(j)} y^{(j)} \mathbf{x}^{(j)} \xrightarrow{\Phi} \Theta = \sum_{j=1}^n \alpha^{(j)} y^{(j)} \Phi(\mathbf{x}^{(j)})$$

Primal variable Dual variable

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

with $\min_{0 \leq \alpha \leq C} \alpha^\top Q \alpha - \alpha^\top \mathbf{1}$ such that $\alpha^\top y = 0$, $Q = Y K Y$, $Y = \text{diag}(y)$, $K(x, y) = \Phi(x)^\top \Phi(y)$
 (proof not included)

Outline

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

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.



Questions?