■ CS286 AI for Science and Engineering

Lecture 3: Traditional Machine Learning (Part 1)

Jie Zheng (郑杰)

PhD, Associate Professor

School of Information Science and Technology (SIST), ShanghaiTech University

Fall Semester, 2020

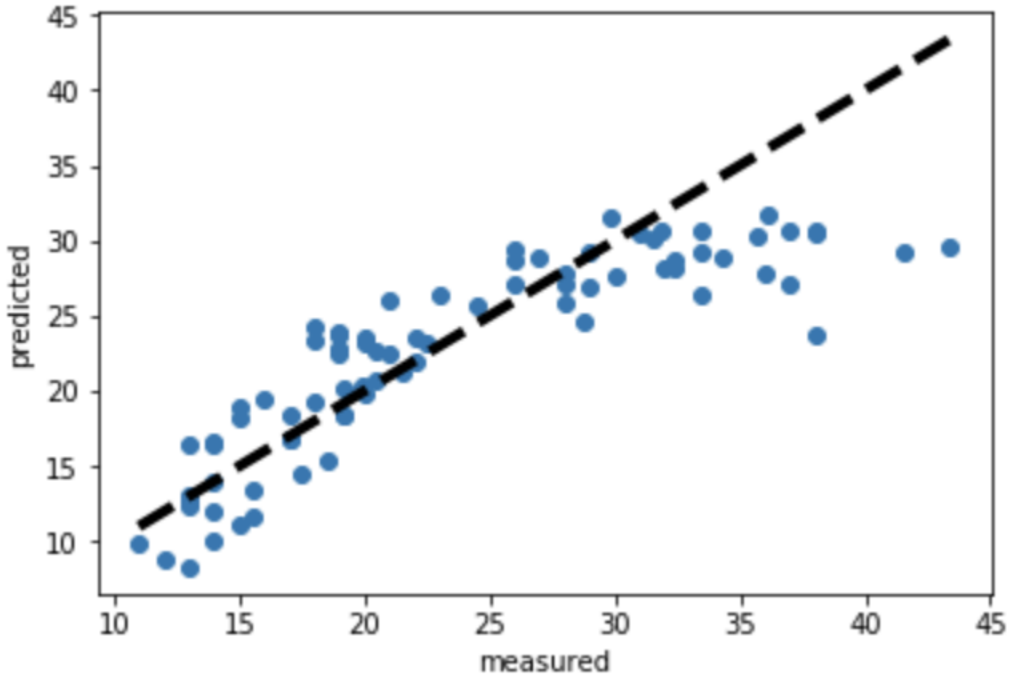
Outline

- Regression
- Bayesian statistics
- Support vector machines

Regression

What is regression?

- Given a set of attributes $\boldsymbol{x} : (x_1, x_2, x_3, \dots, x_n)$ of an object, estimate the mapping function from input $\boldsymbol{x}$ to a continuous output variable y base on training examples.



What is Linear Regression?

- **Linear Regression model** : A linear model makes a prediction by simply computing a weighted sum of the input features $\boldsymbol{x}$:

$$\hat{y} = h_\theta(\boldsymbol{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ \ldots \theta_n x_n$$

  - $\hat{y}$: predicted value
  - n : number of the features
  - x_i : the i^{th} feature value
  - θ_j : the j^{th} model parameter (including bias term θ_0)

Linear Regression

- A Linear Regression model (in **vectorized** form):

$$\hat{y} = \boldsymbol{\theta} \cdot \boldsymbol{x} = \boldsymbol{\theta}^T \boldsymbol{x}$$

  - $\boldsymbol{\theta}$: model parameter vector, containing a bias term and feature weights from θ_0 to θ_n
  - $\boldsymbol{x}$: feature vector of instances
  - $\boldsymbol{\theta} \cdot \boldsymbol{x}$ is the dot product of the vectors $\boldsymbol{\theta}$ and $\boldsymbol{x}$ which is equal to : $\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ \dots \theta_n x_n$

Linear Regression 's objective

- How to obtain a proper Linear Regression model from training examples?
  - Training a model means setting its parameters so that the model best fits the training dataset
- What does "best fit" mean?
  - The most common performance measure of a regression model is the **Root Mean Square Error (RMSE)**:

$$\mathrm{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2}$$

  - In practice, it is simpler to minimize the **Mean Square Error (MSE)**:

$$\mathrm{MSE}\left(\mathbf{X}, h_{\boldsymbol{\theta}}\right) = \frac{1}{m}\sum_{i=1}^{m}\left(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)}\right)^2$$

  - Lower RMSE or MSE scores represent better model fitting

Linear Regression’s objective

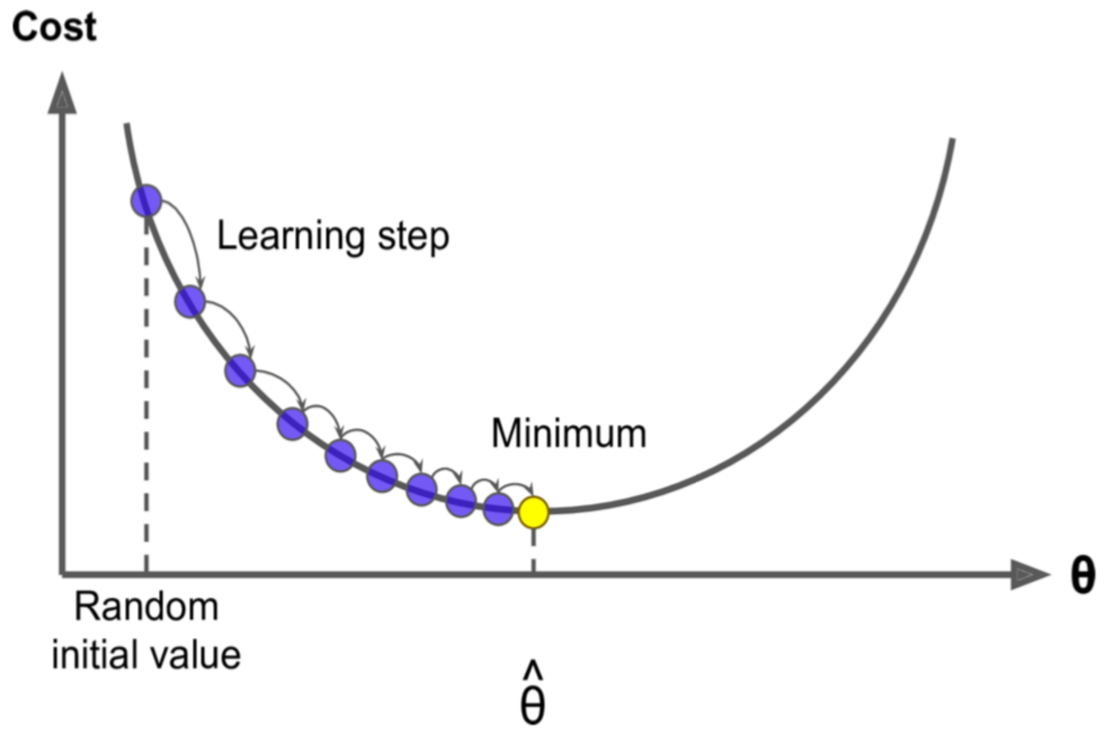
- After combining the MSE and Linear Regression model, we can build a cost function :

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(\boldsymbol{\theta}^T\boldsymbol{x}^{(i)} - y^{(i)}\right)^2$$

- Our objective is to **minimize the MSE cost** by tuning the parameter θ, i.e. to make the model fit to the training samples.

Gradient Descent

- Goal: To decrease the MSE cost via updating parameter θ
- Approach: **Gradient Descent**
  - Filling θ with random values
  - Keep changing θ to reduce the objective function $J(\theta)$



Gradient Descent

- How to change the parameter $\boldsymbol{\theta}$?

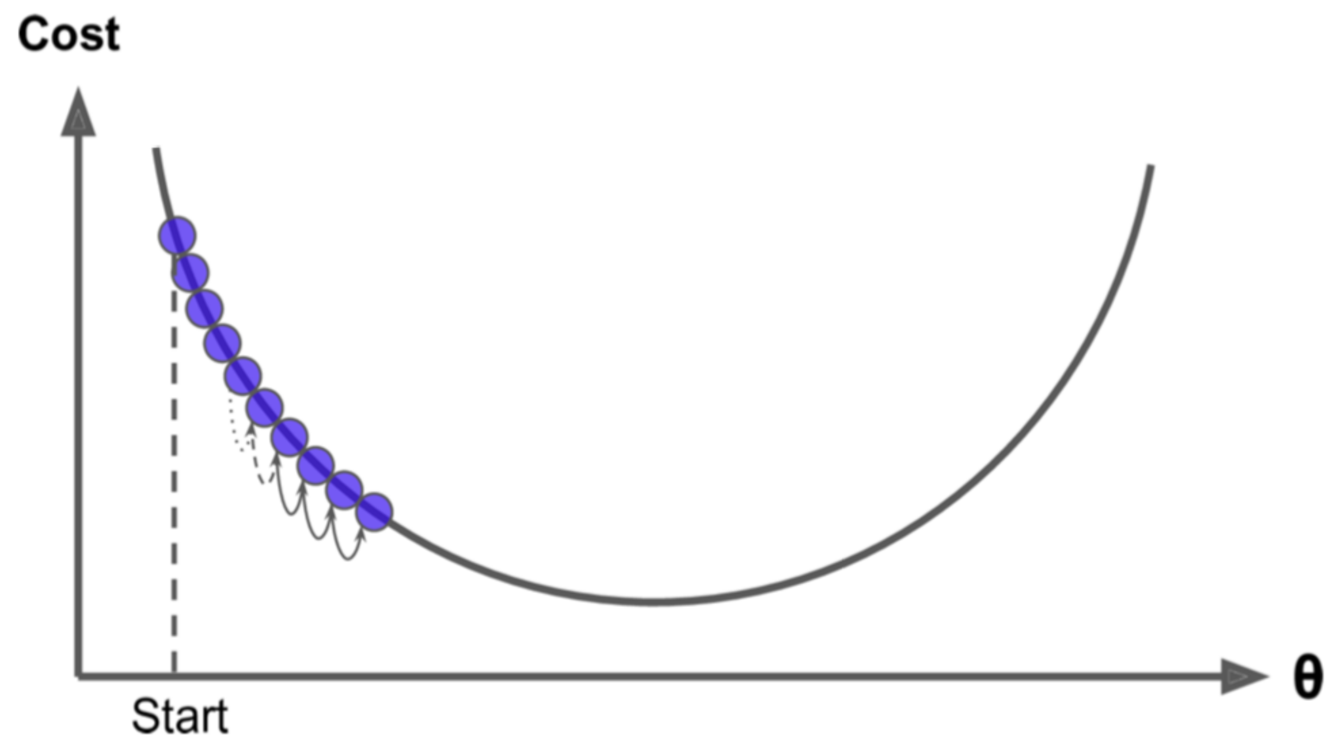
$$\theta_j^{k+1} = \theta_j^k - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}^k)$$

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}^k) = \frac{2}{m} \sum_{i=1}^{m} \left(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)} - y^{(i)}\right) x_j^{(i)}$$

- k : iteration count
- θ_j^k : the j^{th} parameter in the k^{th} iteration
- α : step size (also known as learning rate)

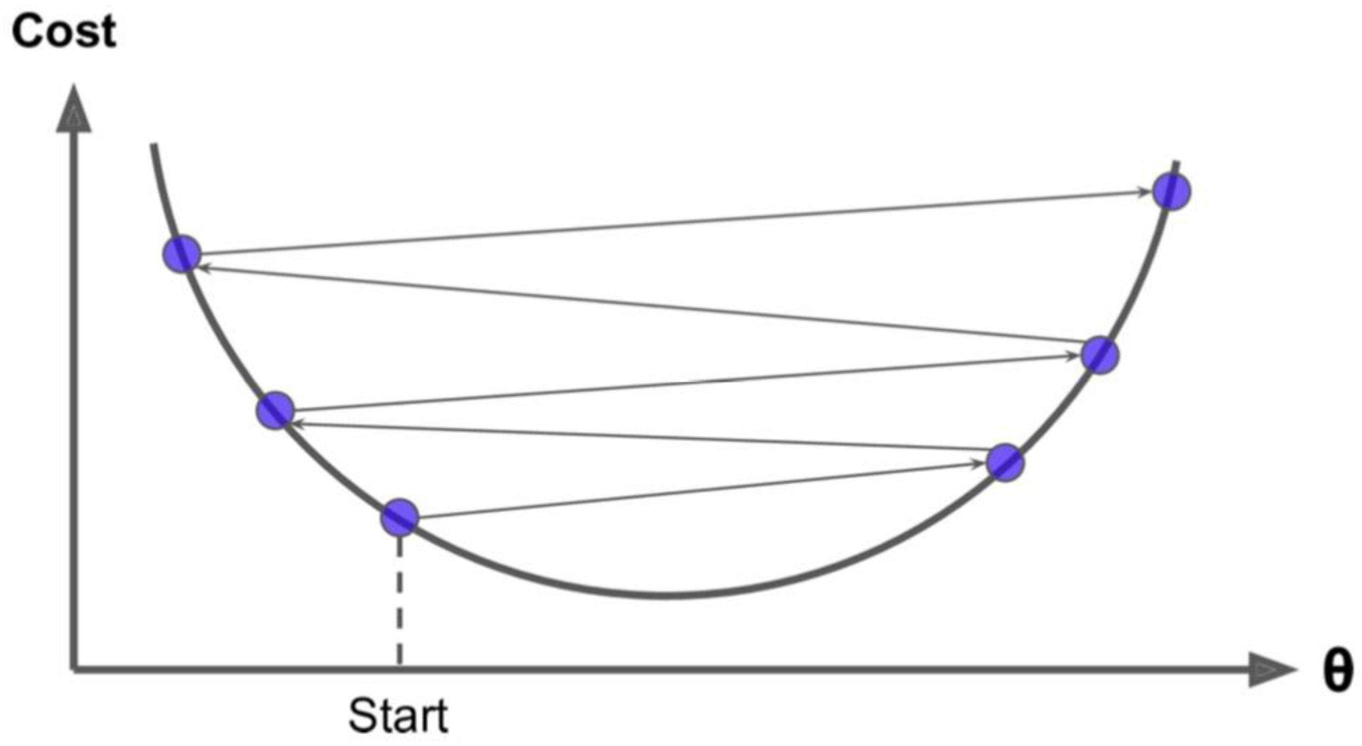
Learning rate in Gradient Descent

- If the learning rate is too small, it is hard to converge



Learning rate in Gradient Descent

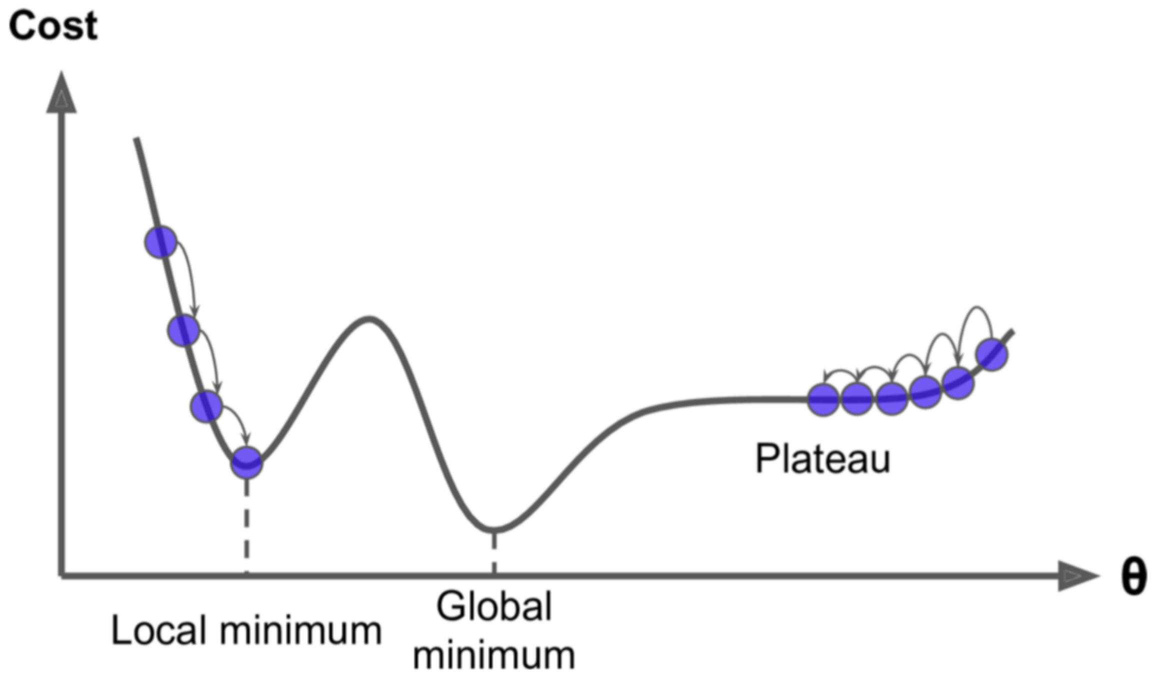
- If the learning rate is too large, results are unstable



Pitfalls of Gradient Descent

Some cost functions may cause problems:

- Local Minimum : It cannot reach the optimal solution by being stuck in local minimum
- Plateau : It takes a long time to cross the plateau



Other Gradient Descent methods

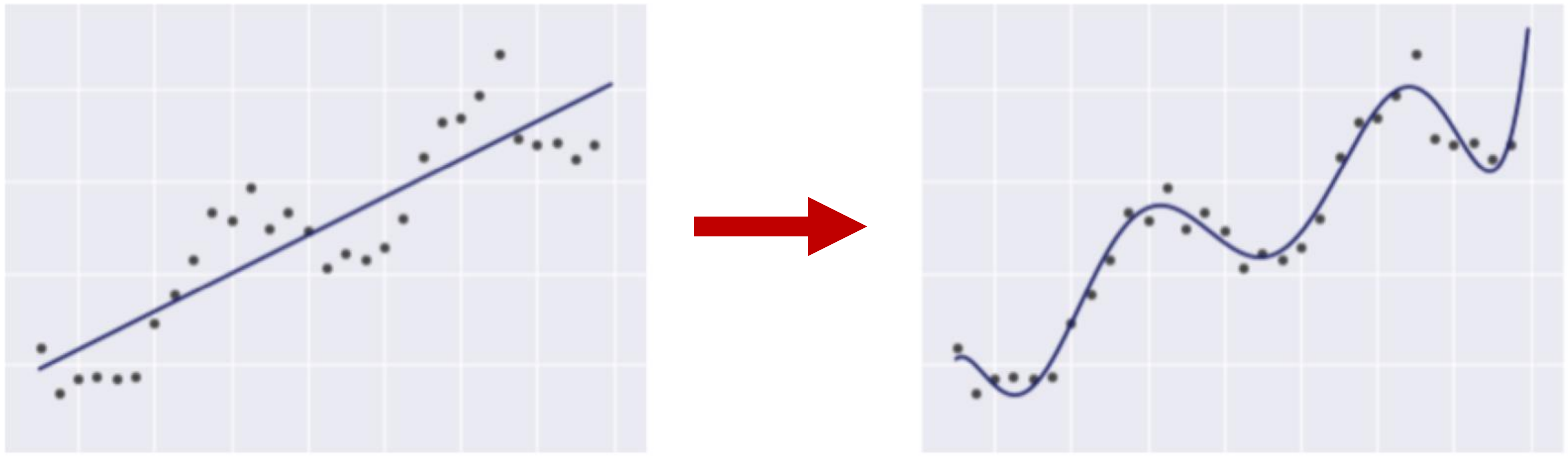
- **Batch Gradient Descent**： Instead of individually computing the partial derivatives, solve them all in one in vectorized form

- **Stochastic Gradient Descent (SGD)**： Pick one random instance in training set each iteration to compute gradient based on that instance

- **Mini-batch Gradient Descent**： Compute the gradients on small random sets of instances

Polynomial Regression

- Linear Regression sometimes cannot fit the training sample well, if the data is nonlinear
- How to use a linear model to fit nonlinear data?
- **Polynomial Regression** is a technique that adds the powers of each feature as new features, and then train a linear model on this extended set of features

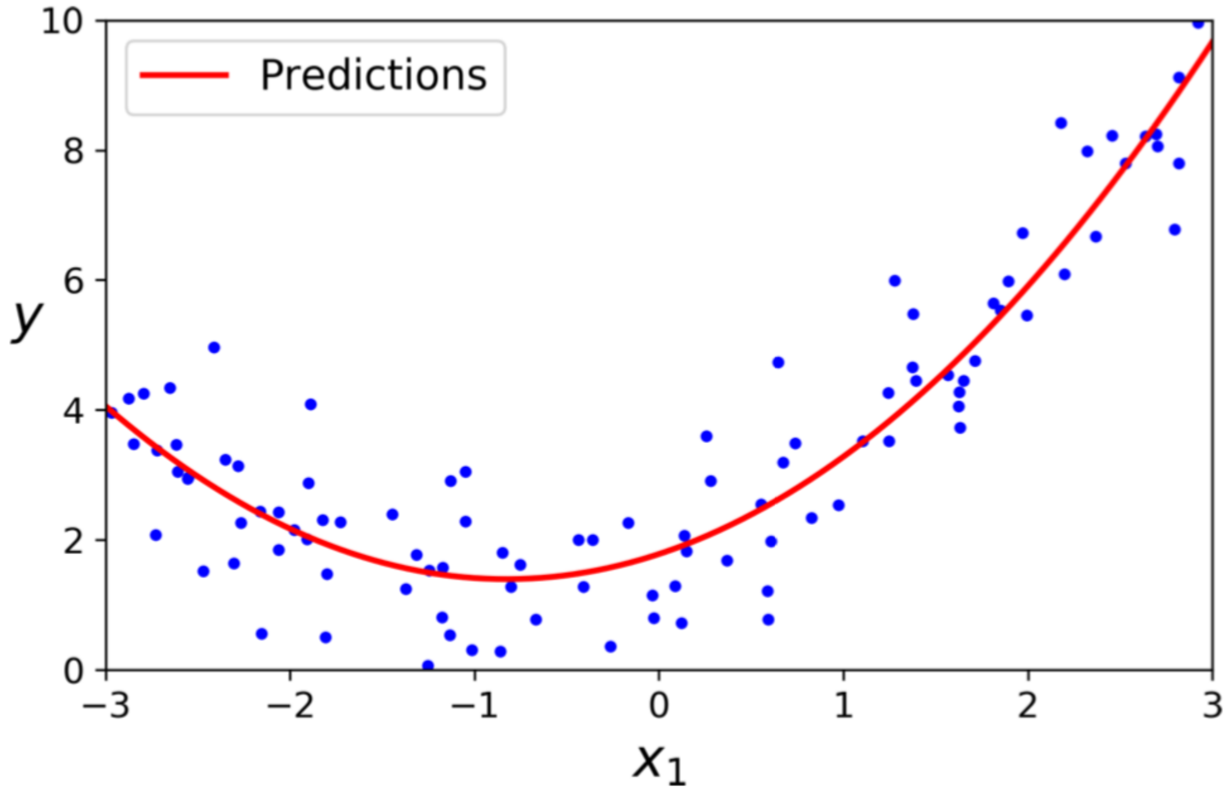


Polynomial Regression

- An example of Polynomial Regression:

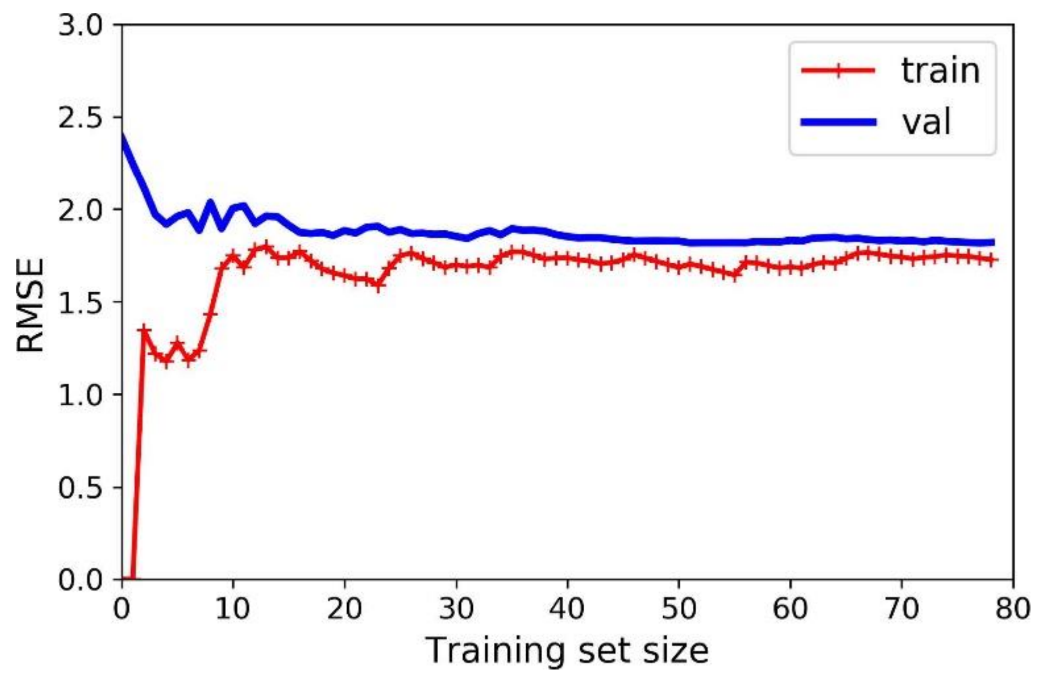
$$\hat{y} = \theta_2 x_1^2 + \theta_1 x_1 + \theta_0$$

- The example input $\boldsymbol{x}$ only has one feature x_1
- The degree of this model is 2

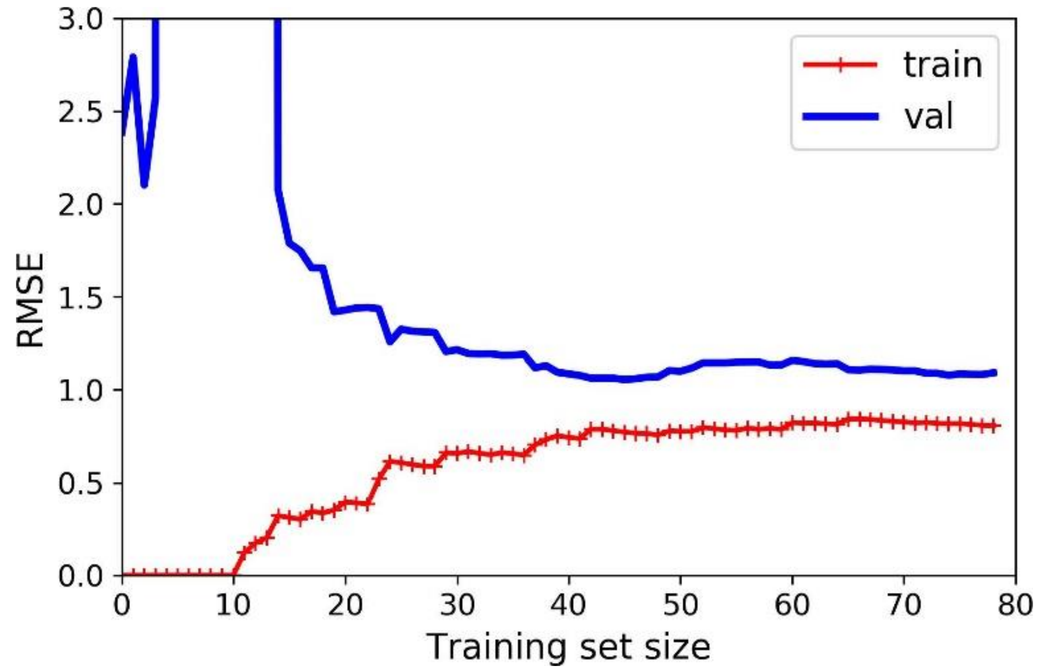


Learning curves

- Learning curves are plots of the model performance (e.g. cost function RMSE) on the training and validation sets as the training set size (or training iteration) changes
  - e.g. training two models on the same data:



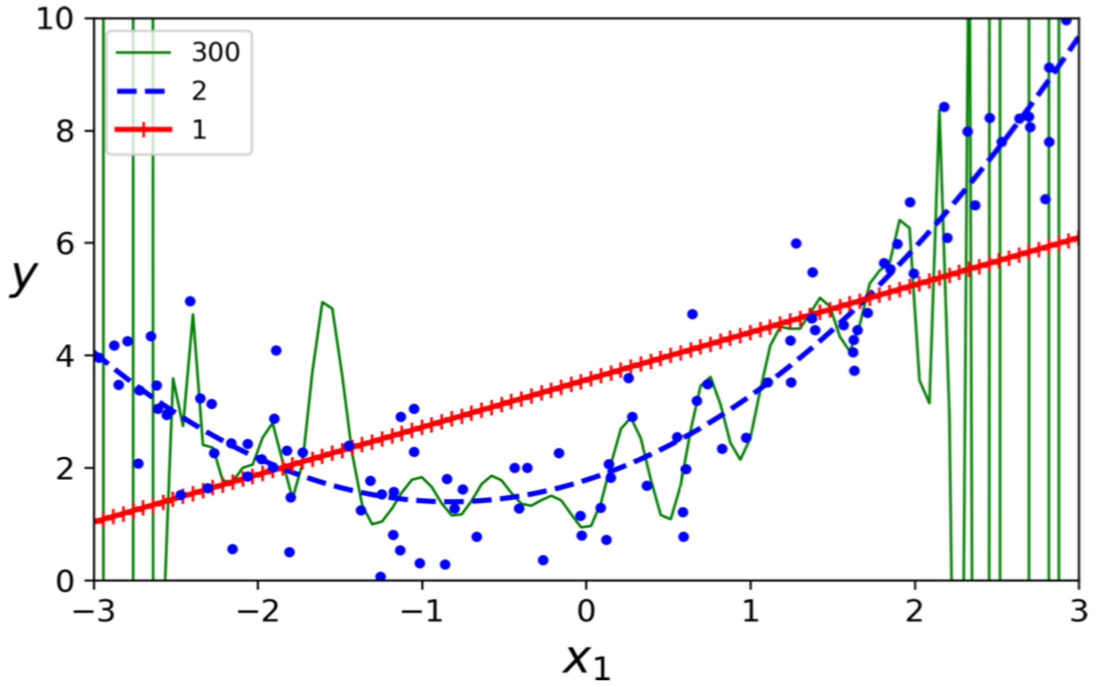
Linear regression: **Underfitting**



Polynomial regression: **Overfitting**

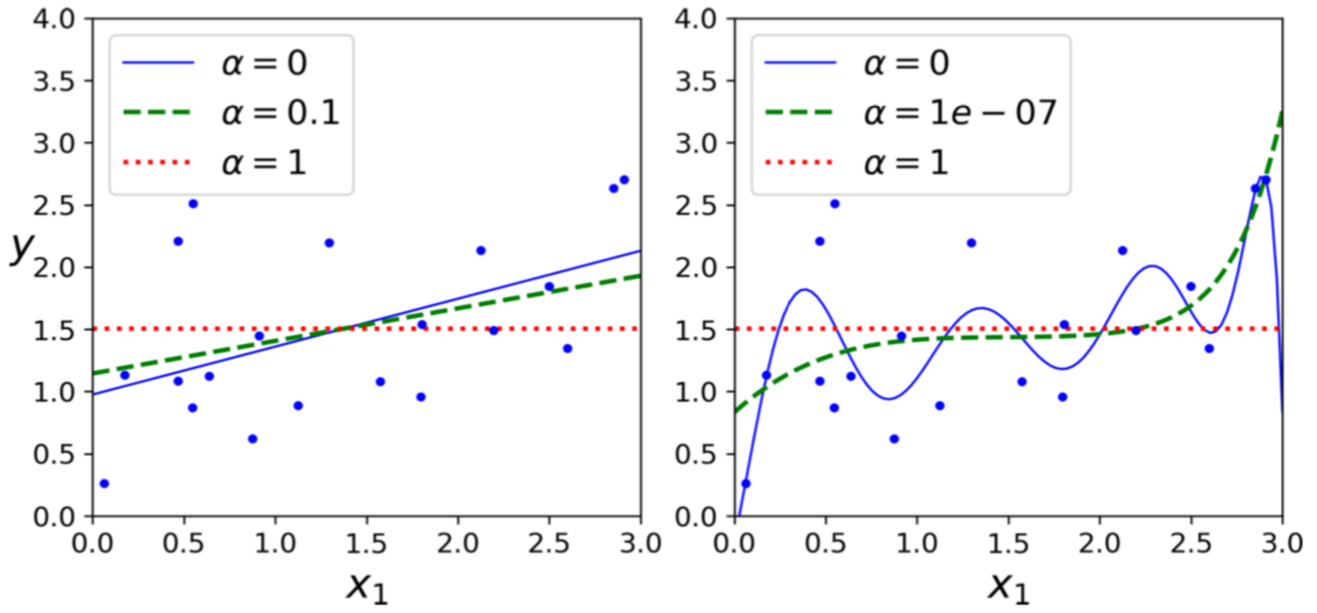
Regularized linear model

- Can we do better for point fitting?
  - Increase the degree of model
- High-degree Polynomial Regression:
  - Low error
  - **Overfitting!**
- **Regularization**
  - Constraining the model to make it simpler and harder for it to overfit the data
  - Example: to regularize a polynomial model by reducing the number of polynomial degrees



Regularized linear model

- Add a **regularized term** in Linear Regression Cost
- **Ridge Regression** : $J(\theta) = MSE(\theta) + \alpha\frac{1}{2}\sum_{i=1}^{n}\theta_i^2$
  - Enforce the parameter $\boldsymbol{\theta}$ to be small
- **Lasso Regression** : $J(\theta) = MSE(\theta) + \alpha\frac{1}{2}\sum_{i=1}^{n}|\theta_i|$
  - Enforce the parameter $\boldsymbol{\theta}$ to be sparse
- α controls the extent to which you want to regularize a model



Lasso Regression with different α values

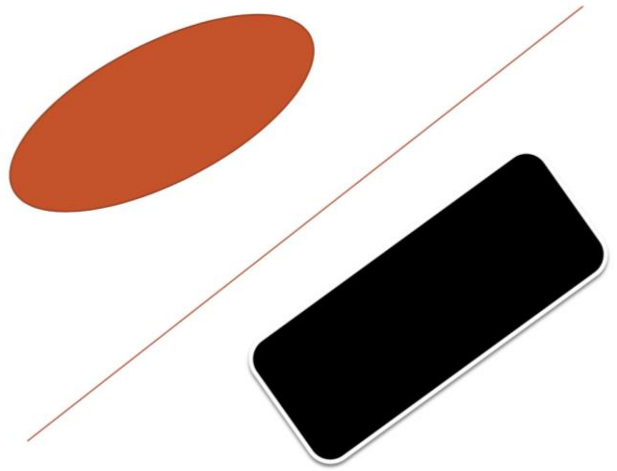
Logistic Regression

- Regression can be used for classification
- Probability is naturally considered in this case

$$\hat{p} = h_\theta(\boldsymbol{x})$$

  - Let $\hat{p}$ denote the probability of $\boldsymbol{x}$ with label 1
  - Then, $1 - \hat{p}$ is the probability of $\boldsymbol{x}$ with label 0

- To make binary prediction: $\hat{y} = \begin{cases} 0 & \text{if } \hat{p} < 0.5 \\ 1 & \text{if } \hat{p} \geq 0.5 \end{cases}$

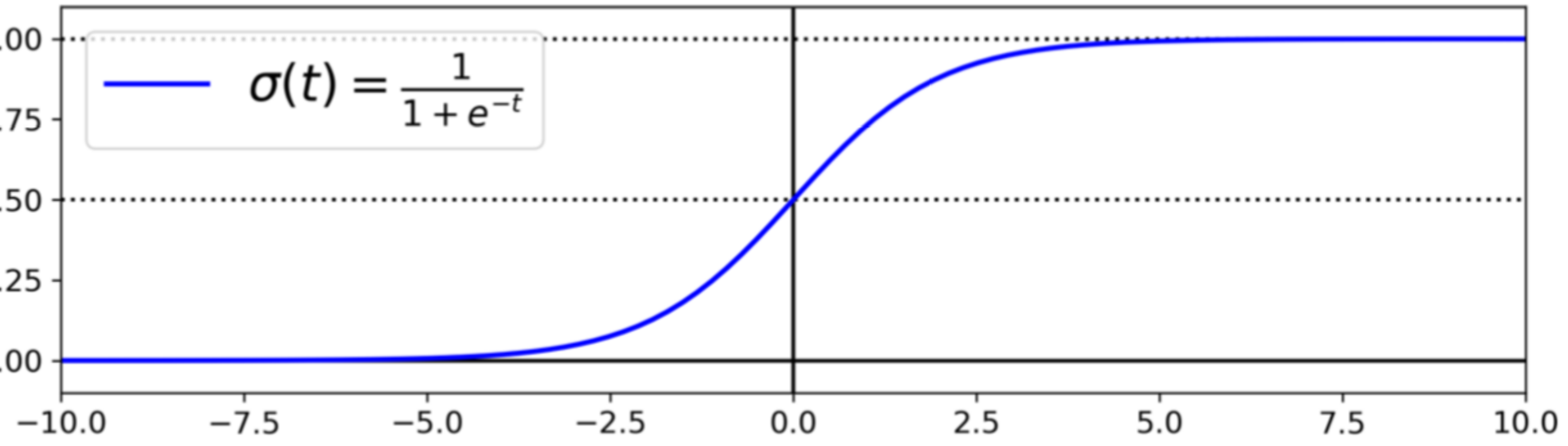


Sigmoid function

- Using the **Sigmoid function** to define the probability $\hat{p}$

$$\hat{p} = h_\theta(\boldsymbol{x}) = \sigma(\boldsymbol{x}^T\boldsymbol{\theta})$$

where $\sigma(t) = \frac{1}{1+\exp(-t)}$



Logistic Regression Cost Function

- **Training objective**: To set the parameter vector $\boldsymbol{\theta}$ so that the model estimates high probabilities for positive instances and low probabilities for negative instances
- Cost function of a single training instance :

$$c(\boldsymbol{\theta}) = \begin{cases} -\log(\hat{p}) & \text{if } y = 1 \\ -\log(1-\hat{p}) & \text{if } y = 0 \end{cases}$$

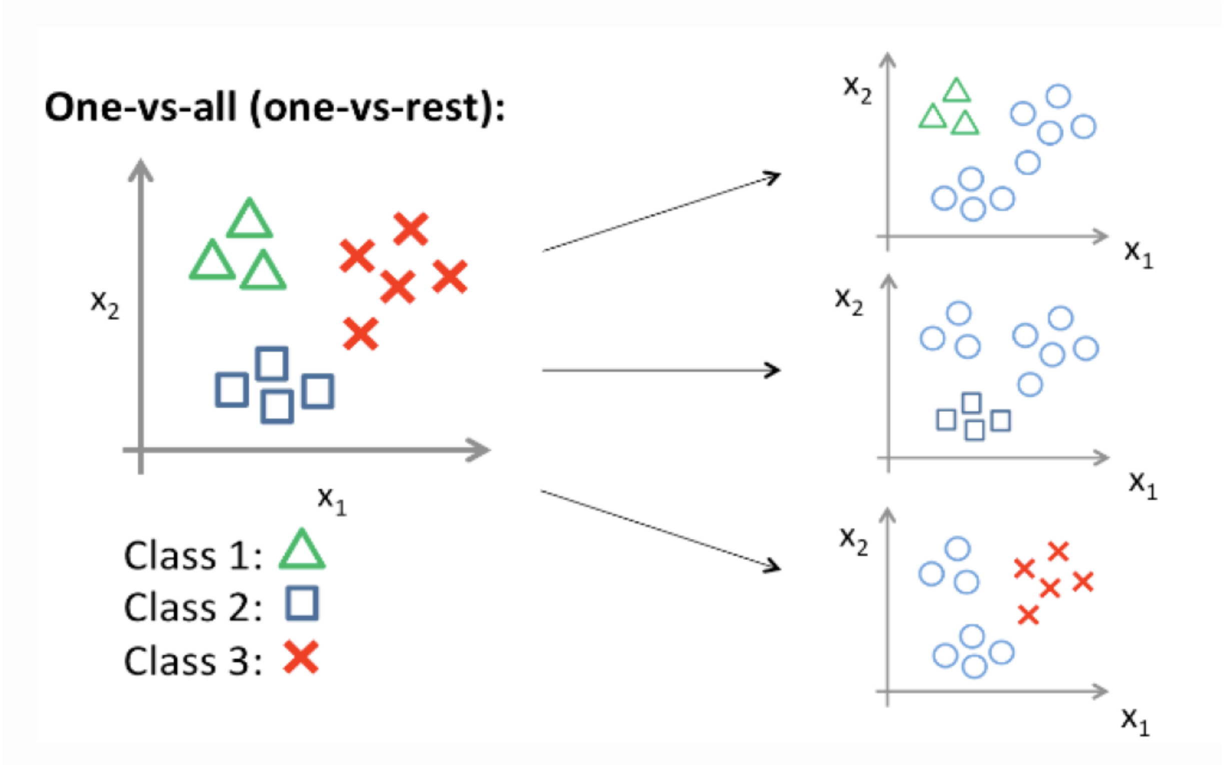
- **Logistic Regression cost function** is the average over all training instances:

$$J(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(\hat{p}^{(i)}\right) + \left(1-y^{(i)}\right)\log\left(1-\hat{p}^{(i)}\right)\right]$$

Softmax Regression

- Logistic Regression can only handle binary classification
- But Logistic Regression can be generalized to support multiple classes, using **Softmax Regression**
- Basic idea :
  1. Compute a score $s_k(\boldsymbol{x})$ for each class k
  2. Estimate the probability of each class by applying the softmax function to the scores

$$\hat{p}_k = \sigma(\mathbf{s}(\mathbf{x}))_k = \frac{\exp\left(s_k(\mathbf{x})\right)}{\sum_{j=1}^{K} \exp\left(s_j(\mathbf{x})\right)}$$



Bayesian Statistics

Conditional probability

- **Joint probability** $P(e_1, e_2)$ is the probability that two (or more) events, say e_1 and e_2, have happened together and is given by the probability of the intersection of two events.
- **Conditional probability** $P(e_1 | e_2)$ of two events, say e_1 given e_2 , is the probability of event e_1 given that the event e_2 (the *prior* event) has already occurred, and is given by

$$P(e_1 | e_2) = \frac{P(e_1, e_2)}{P(e_2)}$$

- Therefore, $P(e_1, e_2) = P(e_1 | e_2) P(e_2)$
- **Independent events**: The probability of one event does not depend on the other event. Therefore, $P(e_1 | e_2) = P(e_1)$ and $P(e_2 | e_1) = P(e_2)$
- That is, the joint probability of independent events is equal to the product of their individual probabilities:

$$P(e_1, e_2) = P(e_1) P(e_2)$$

Bayes’ Theorem

- For two events e_1 and e_2 , since $P(e_1, e_2) = P(e_1 | e_2)P(e_2) = P(e_2 | e_1)P(e_1)$,

  Bayes’ theorem is given by

$$P(e_1 | e_2) = \frac{P(e_2 | e_1)P(e_1)}{P(e_2)}$$

- For a given set of mutually exclusive and exhaustive event $e, e_1, e_2, \ldots, e_n$, the probability of event e is give by **the total probability formula (marginalization)**:

$$P(e) = \sum_{i=1}^{n} P(e, e_i) = \sum_{i=1}^{n} P(e | e_i)P(e_i)$$

- For such events, the Bayes’ theorem is extended to

$$P(e_i | e) = \frac{P(e | e_i)P(e_i)}{P(e)} = \frac{P(e | e_i)P(e_i)}{\sum_{j=1}^{n} P(e | e_j)P(e_j)}$$

Probabilistic modeling

- Most biological phenomena are due to random events or interpreted with probability
- How to build probabilistic models for biological data, processes, or phenomena?
  - E.g. sequences (DNA, protein, etc.), gene expressions, protein structures, evolution
- Such a model assigns high probability to data/information when it fits the phenomenon well, and low probability for those it does not fit well
- Issues in probabilistic modeling:
  - What is the best model? multiple models; methods to assess how well a given dataset D fits a model instance M; the model selection problem
  - What is the learning algorithm? That is, how to determine the parameters of the model?
  - Amount of data available for model training (or parameter estimation)?
  - Characteristics of data (noise, independence, redundancies, biases)?
  - Unless stated otherwise, the data $d \in D$ are generally assumed to be independent

Maximum Likelihood (ML)

- Once the model M is chosen, the parameters of the model have to be inferred from the data. This is referred to as **learning** or **training** of the model
- Given the model M and its parameters α, the **likelihood** of data D is given by

$$P(D \mid \alpha, M)$$

- The likelihood indicates how well the model predicts the data
- The **maximum likelihood (ML)** estimator maximizes the likelihood of the data given the model. That is, it finds the optimal set of parameters α^{ML} that maximize the likelihood:

$$\alpha^{\mathrm{ML}} = \arg\max_{\alpha} P(D \mid \alpha, M)$$

- Often the log-likelihood (natural logarithm of the likelihood function) is used for computational efficiency.

Strengths and drawback of ML

- **Consistency**: Maximum likelihood is **consistent** in the sense that the true (unknown) parameter α_0 will also, in the limit of a large amount of data, be the value that maximizes the likelihood, i.e., $\alpha \to \alpha_0$ as the data size $n \to \infty$
- Other nice properties of ML:
  - Efficient: needs less data than other estimators to achieve a given performance
  - Invariance to parameter transformation: If θ* is the MLE of θ, then for any function f(θ), the MLE of f(θ) is f(θ*)
- Drawback: When the data are scanty, ML can give poor results
  - Example: In rolling a die, to estimate the probabilities of the 6 faces, $\theta_1, \theta_2, \ldots, \theta_6$, if we use only 3 different rolls of the die, then the ML estimate is

$$\theta_i = n_i / \sum n_k$$

  - But then, at least 3 of the 6 parameters have values 0, a bad estimator
  - Solution: To incorporate *prior* knowledge (e.g. $\theta_1, \theta_2, \ldots, \theta_6$ should all be near 1/6)

Maximum a posteriori (MAP)

- Given the data *D* and the model *M*, the **posterior probability** is the probability of parameters, $P(\alpha \mid D, M)$
- From Bayes' theorem:

$$P(\alpha \mid D, M) = \frac{P(D \mid \alpha, M) P(\alpha, M)}{P(D, M)} = \frac{P(D \mid \alpha, M) P(\alpha \mid M) P(M)}{P(D, M)}$$

- Because the parameters α do not depend on the terms $P(M)$ or $P(D, M)$,

$$P(\alpha \mid D, M) \propto P(D \mid \alpha, M) P(\alpha \mid M)$$

- The **maximum a posteriori (MAP)** estimator gives the parameters that maximize the posterior probability of the parameters, i.e. the MAP estimator is given by

$$\alpha^{\text{MAP}} = \arg\max_{\alpha} P(D \mid \alpha, M) P(\alpha \mid M)$$

- The **prior probability** of parameters $P(\alpha \mid M)$ is chosen in some reasonable manner to incorporate *prior* (biological) knowledge (Bayesian statistics)

Bayesian modeling

- The Bayes’ theorem is from:

$$P(B|A) = P(A \text{ and } B)/P(A)$$

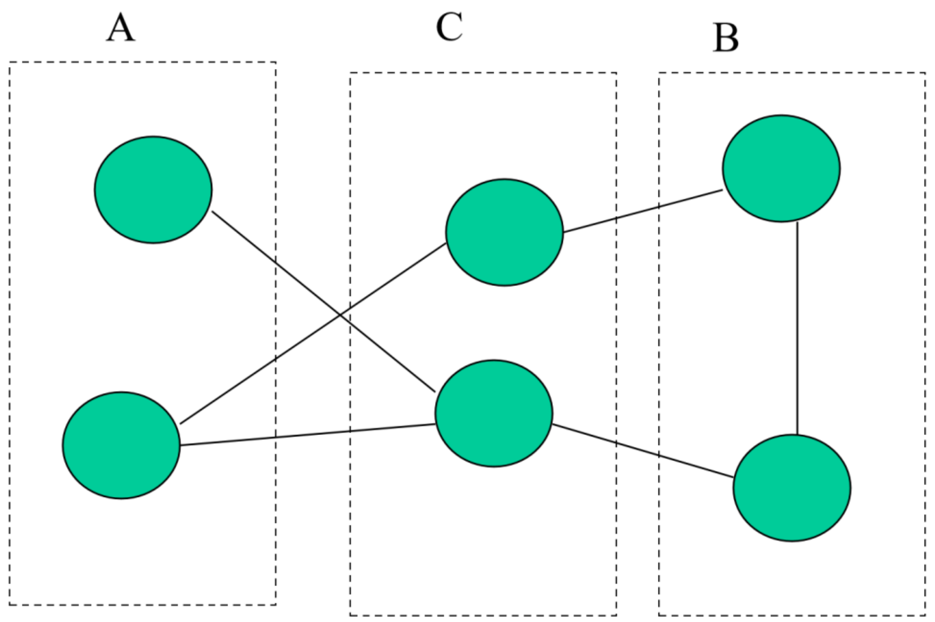
  - A and B usually represent observed data Y and parameters θ, and the goal is to infer the posterior distribution of the parameters

$$\pi(\theta|Y) = \pi(\theta)P(Y|\theta)/P(Y)$$

  - Example: Y is the genetic marker data for a person, θ is the ethnic origin of the person (e.g. Caucasian, Asian or African)
- Key elements:
  - Model specifications, needed to evaluate $P(Y|\theta)$
  - Prior specifications, needed to define $\pi(\theta)$
  - Computational methods needed to infer the posterior distribution $\pi(\theta|Y)$
- Tips for modeling:
  - The model should be comprehensive enough to appropriately model the obtained data
  - The degree of knowledge about the model parameters can be reflected by the prior distributions
  - Posterior distributions are often inferred using Markov chain Monte Carlo (MCMC)

Graphical model

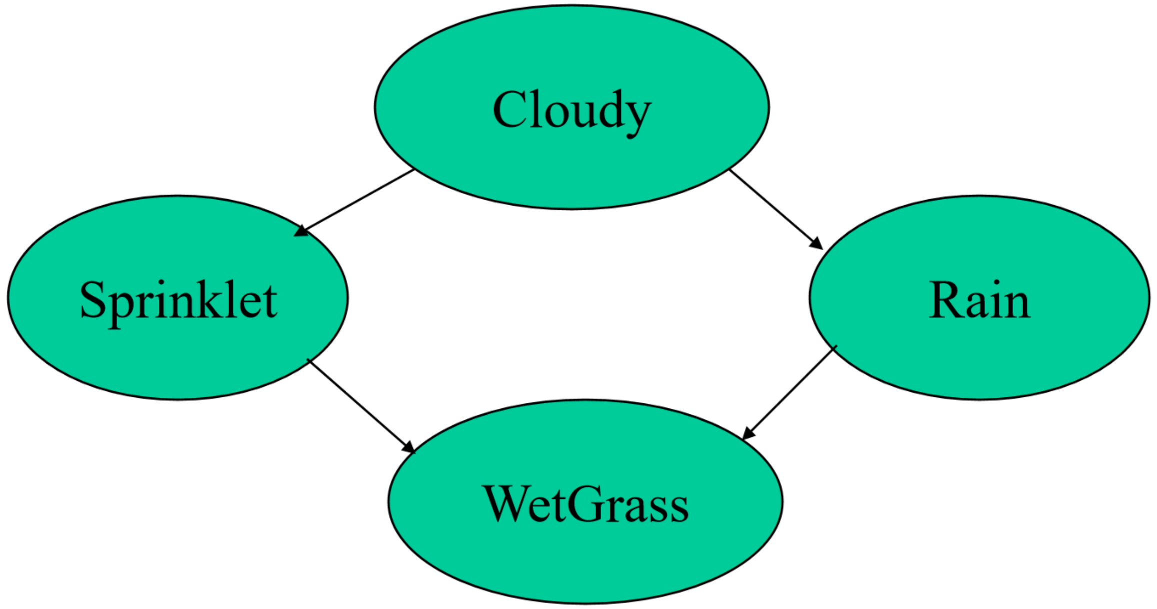
- **Probabilistic graphical models** are graphs in which nodes represent random variables, and the (lack of) arcs represent dependence (conditional independence)
  - It provides a compact representation of joint probability distribution
- **Markov random field**: undirected graphical models (also called **Markov networks**)
- Two sets of nodes A and B are **conditionally independent given a third set C** if all paths between A and B are separate by a node in C



$A \perp B \mid C$

Bayesian Networks

- **Bayesian networks** are directed graphical models (also called **Belief networks**)
- Popular with AI and statistics communities
- A model with both directed and undirected arcs is called a **chain graph**
- Compared with undirected graphical models, directed models:
  - A->B can encode causal relationship
  - Can encode deterministic relationship and are easier to learn (i.e. fit to data)

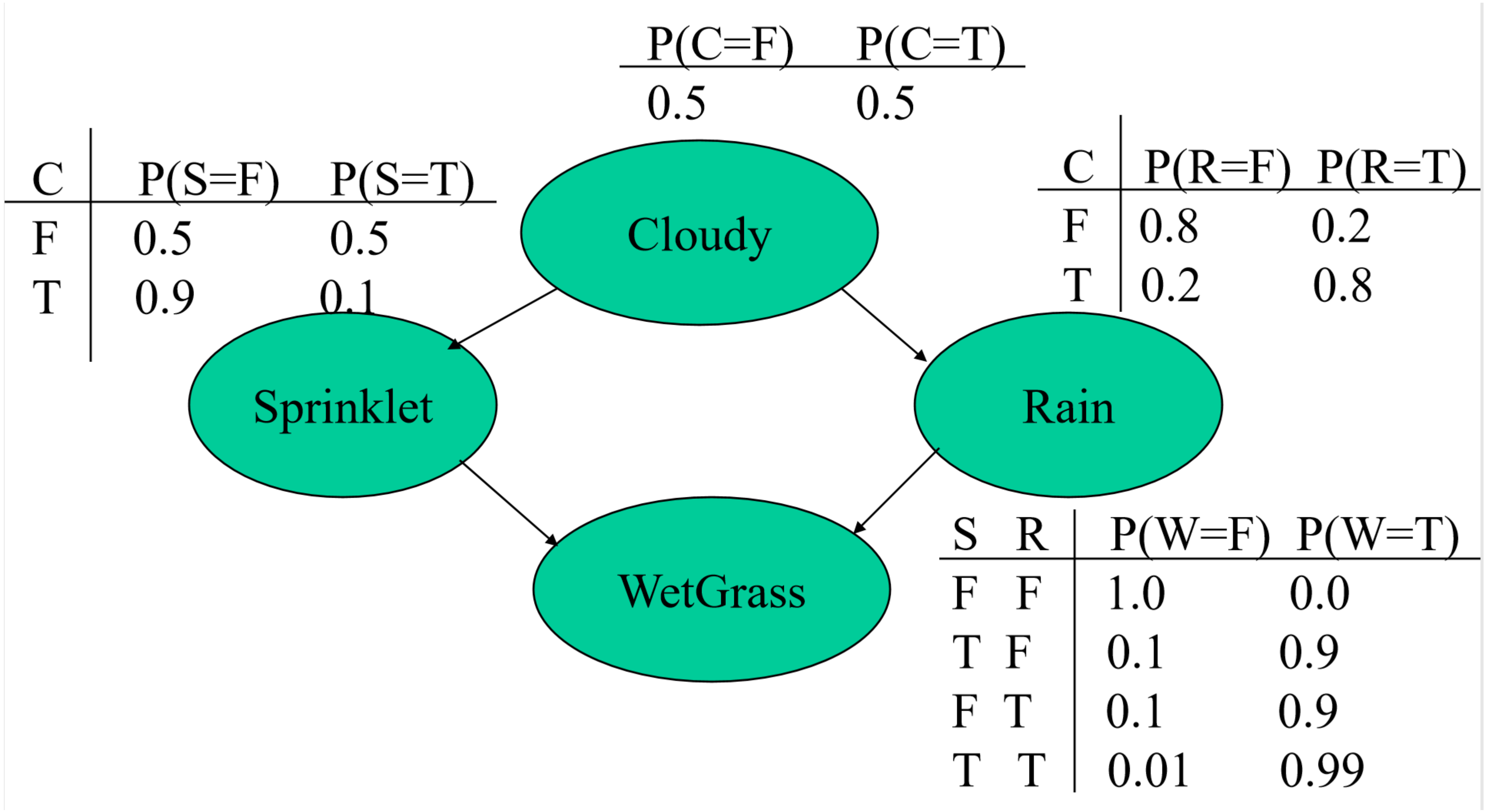


Advantages of Bayesian networks

- Compact and intuitive representation
- Captures causal relationships
- Efficient model learning (parameters and structure)
- Deals with noisy data
- Integration of prior knowledge
- Effective inference algorithms

Conditional probability distribution

- Discrete variable: CPT (conditional probability table)



P(C=F)	P(C=T)
0.5	0.5

C	P(S=F)	P(S=T)
F	0.5	0.5
T	0.9	0.1

C	P(R=F)	P(R=T)
F	0.8	0.2
T	0.2	0.8

S	R	P(W=F)	P(W=T)
F	F	1.0	0.0
T	F	0.1	0.9
F	T	0.1	0.9
T	T	0.01	0.99

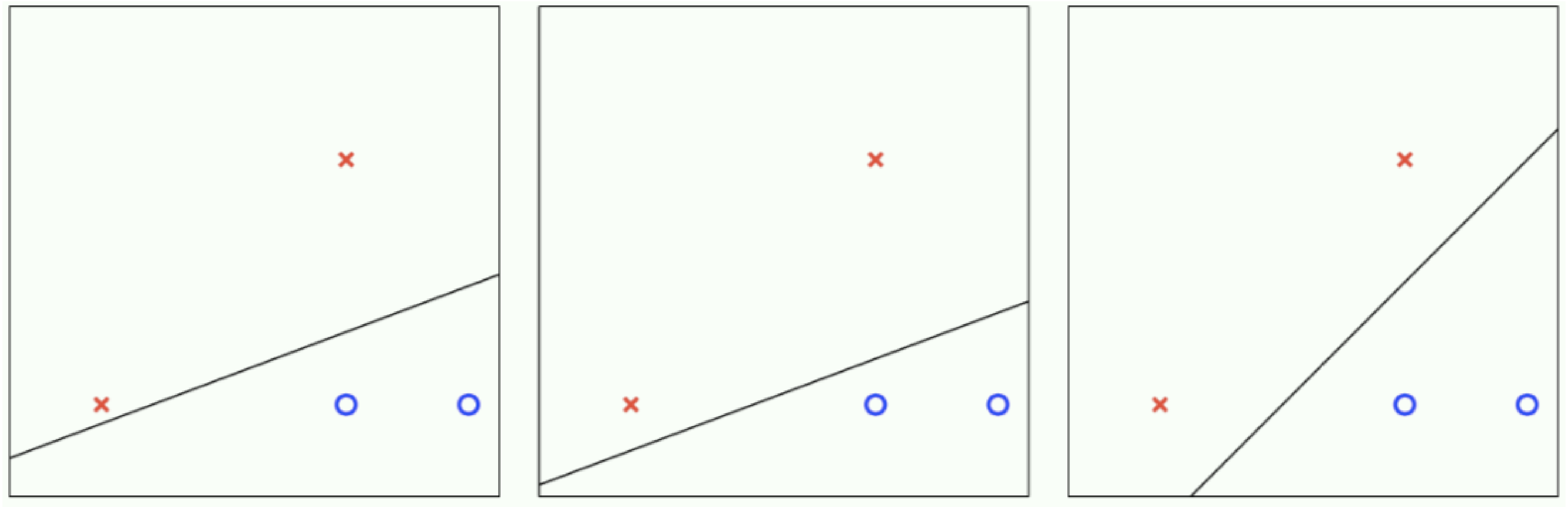
Inference with Bayesian networks

- **Probabilistic inference** is one of the most common tasks that Bayesian networks are used to solve
- Example: Suppose we observe that the grass is wet. There are two possible causes for this:
  - (1) It is raining, or
  - (2) the sprinkler is on
- Which is more likely? We can use Bayes' rule to compute the posterior probability of each explanation

Support Vector Machines

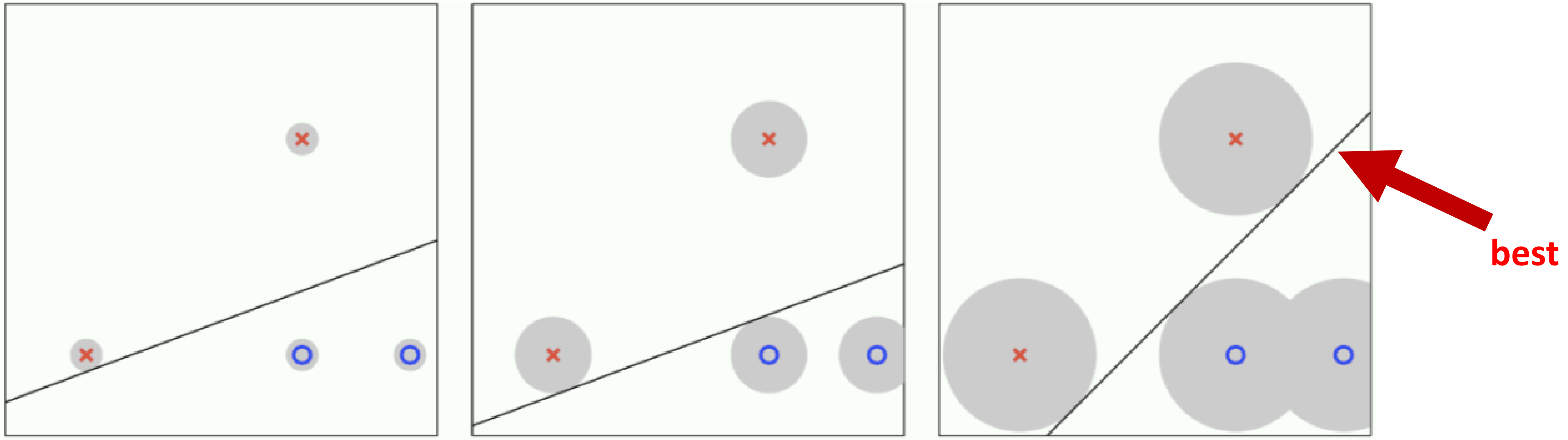
Linear SVM classification

- To separate the points, which line is better ?



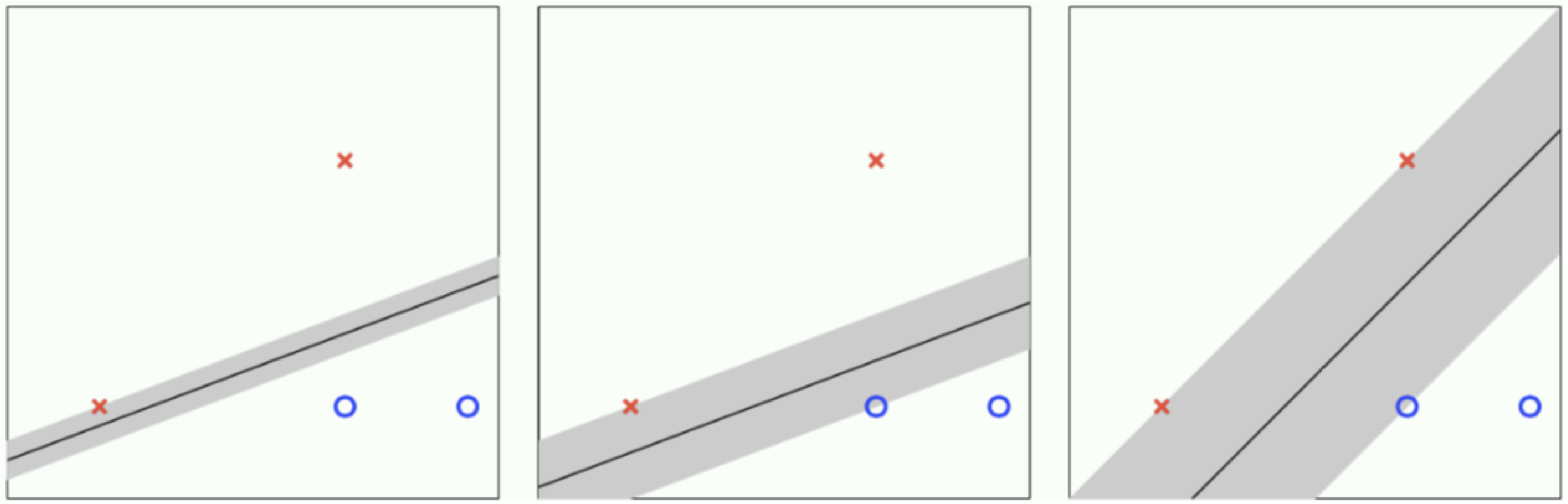
Linear SVM classification

- The line can deal with the more noise data is better



Linear SVM classification

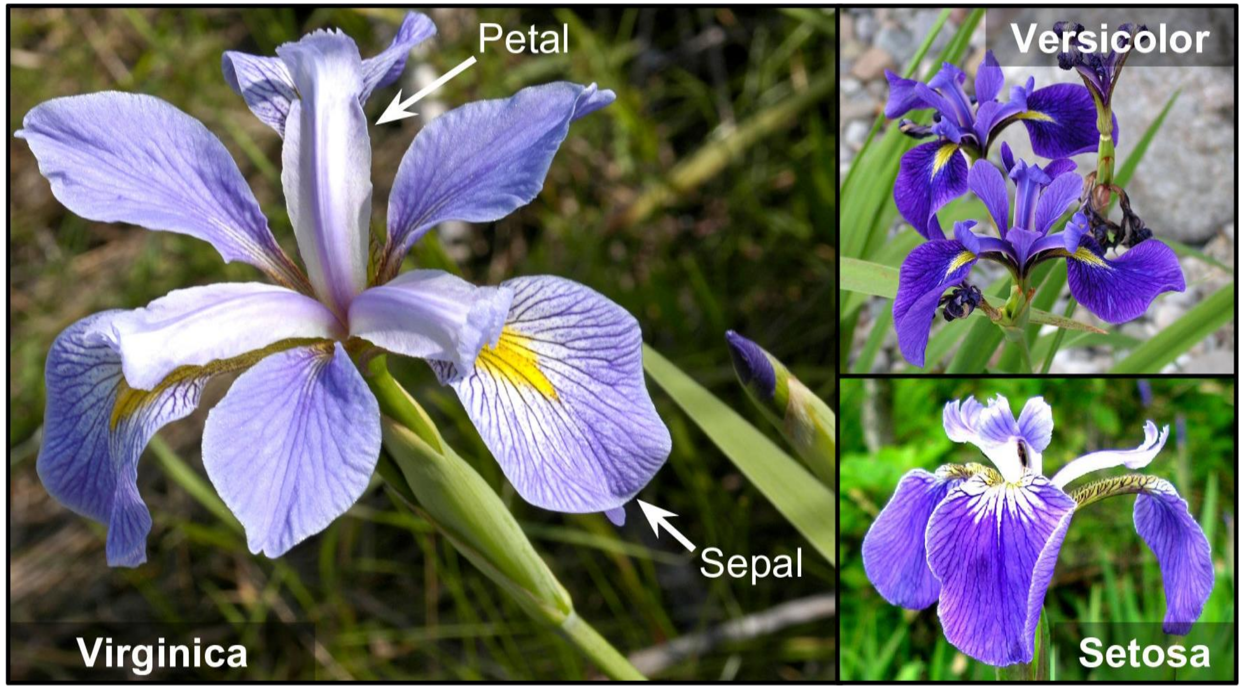
- The margin between two sets should be as large as possible



- Linear SVM aims to find a line that separates two sets with the widest margin, i.e. fitting the widest possible street between the classes

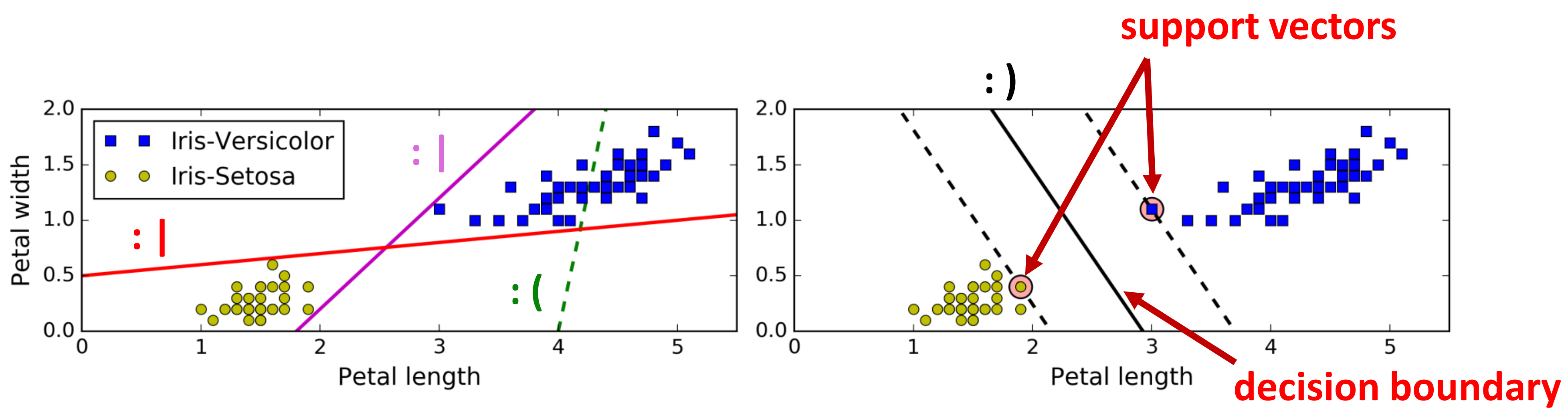
iris dataset

- Sepal and Petal length and width of 150 iris flowers
- There are 3 different species:
  - Iris-Setosa
  - Iris-Versicolor
  - Iris-Virginica



Linear SVM classification

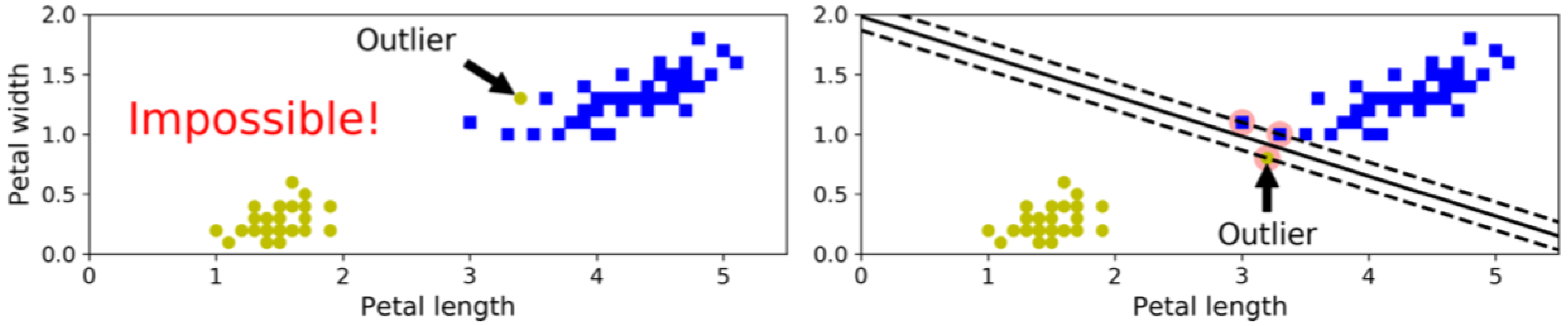
- Build a classifier to distinguish Iris-Virginica and Iris-Setosas based on the two features of petal width and petal length



- Adding more training instances "off the street" will not affect the decision boundary at all:
  - it is fully determined (supported) by the instances located on the edge of the street, which are called support vectors

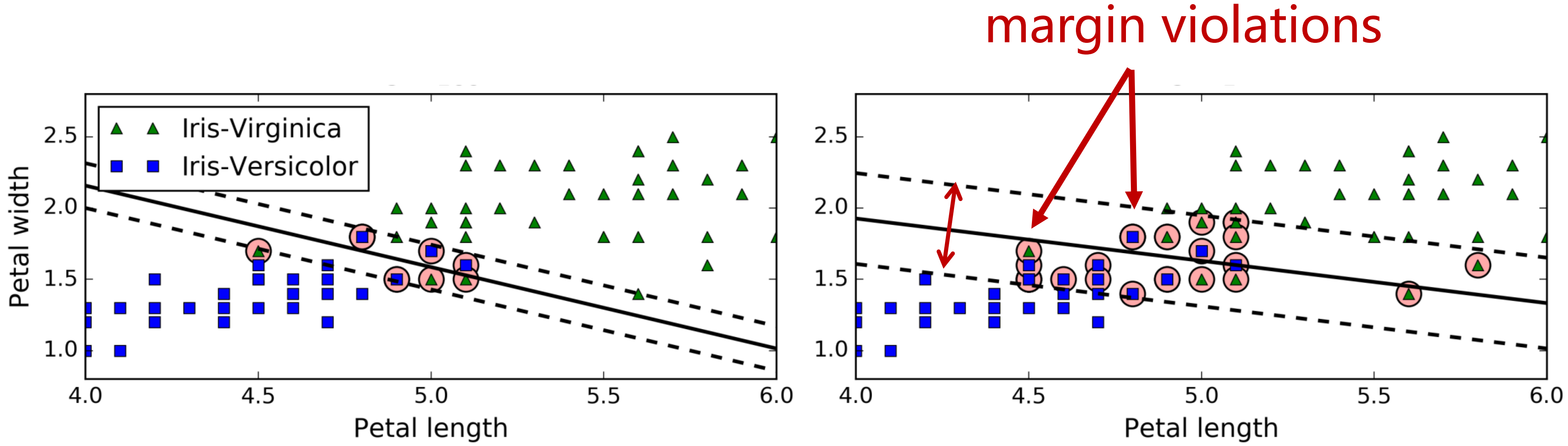
Hard margin classification

- Hard margin classification : all instances must be
  - off the street, and
  - on the right side
- Two issues:
  - It only works if the data is linearly separable
  - It is quite sensitive to outliers (i.e. instances that are not in accord with the overall data distribution)



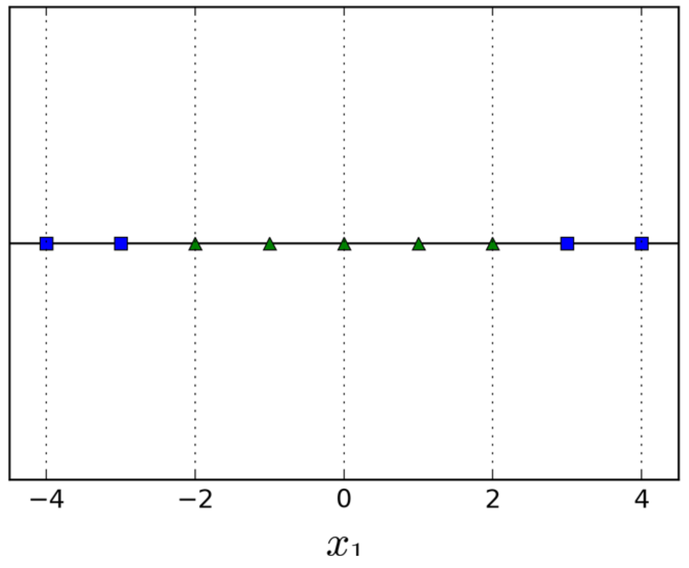
Soft margin classification

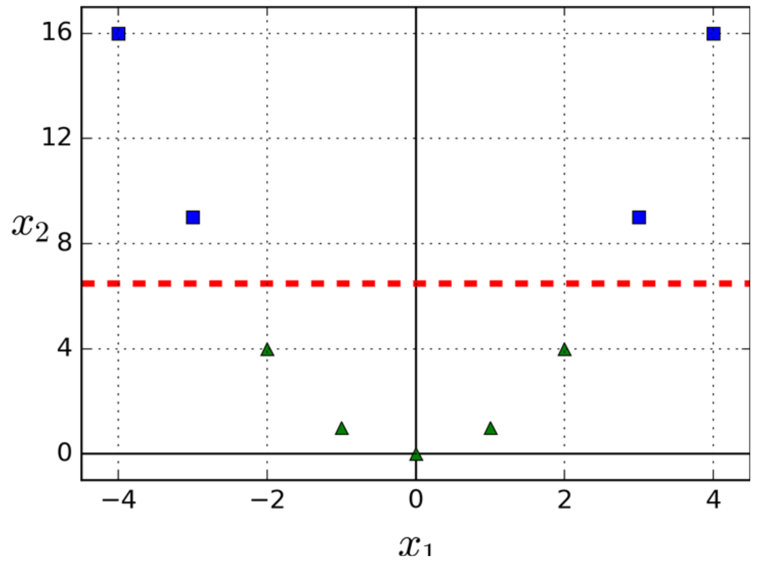
- To avoid the two issues, it is preferable to find a good balance between:
  - keeping the street as large as possible, and
  - limiting the margin violations,
- This is called **soft margin classification** (allowing some mistakes to make the classifier generalize better)



Nonlinear SVM classification

- Another idea is to use nonlinear lines tosssss separate the sets
- Adding features can make this possible:
  - Left figure represents a simple dataset with just one feature, which is not linearly separable
  - If we add a second feature $x_2 = (x_1)^2$, the resulting 2D dataset is perfectly linearly separable

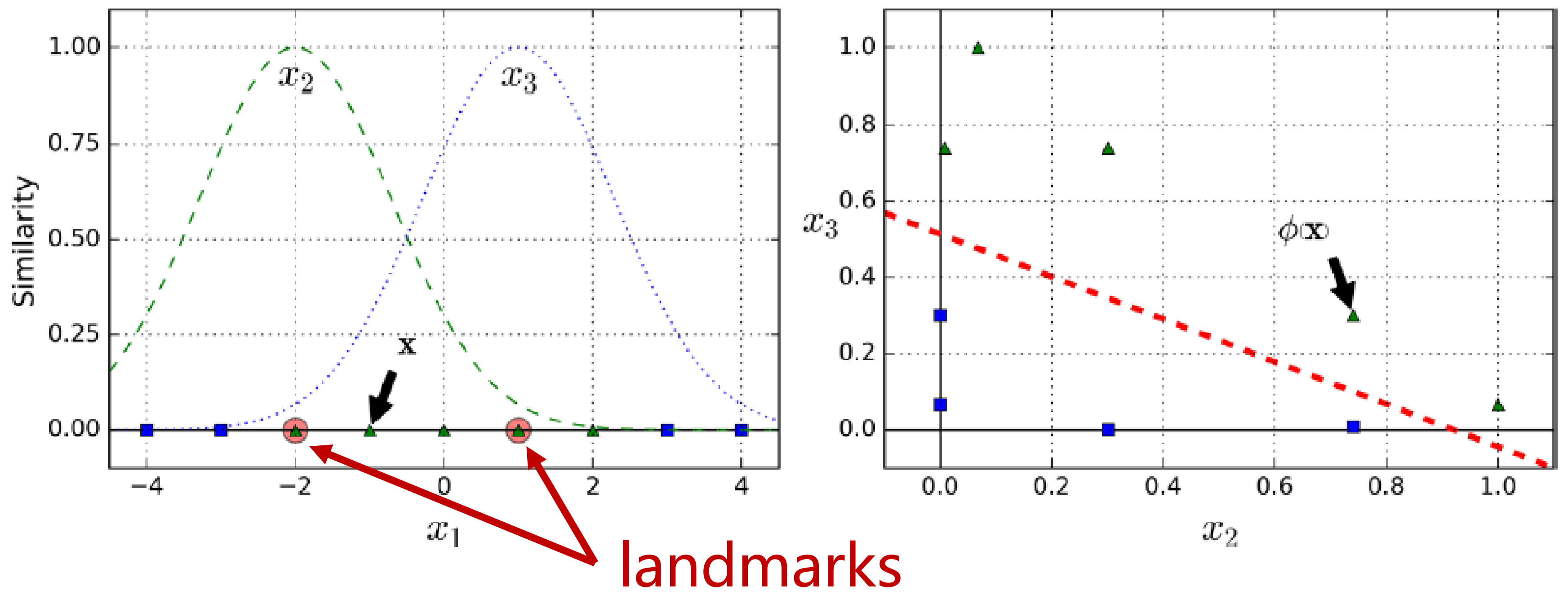




Nonlinear SVM classification (continued)

- Adding features computed using a similarity function that measures how much each instance resembles a particular landmark
  - e.g. Gaussian Radial Basis Function (RBF)

$$\phi\gamma(\mathbf{x}, \ell) = \exp\left(-\gamma \| \mathbf{x} - \ell \|^2\right)$$



Kernelized SVM

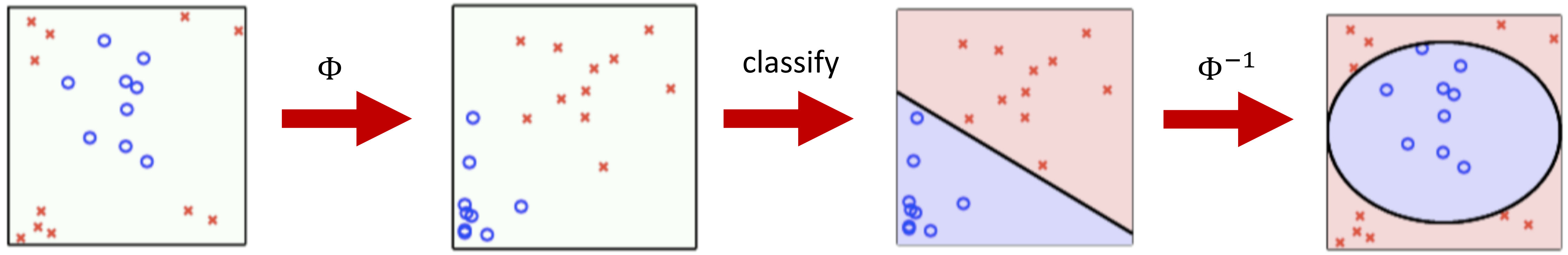
Issue: Adding all features is computationally expensive, especially on large training sets, making the model too slow

Kernel trick

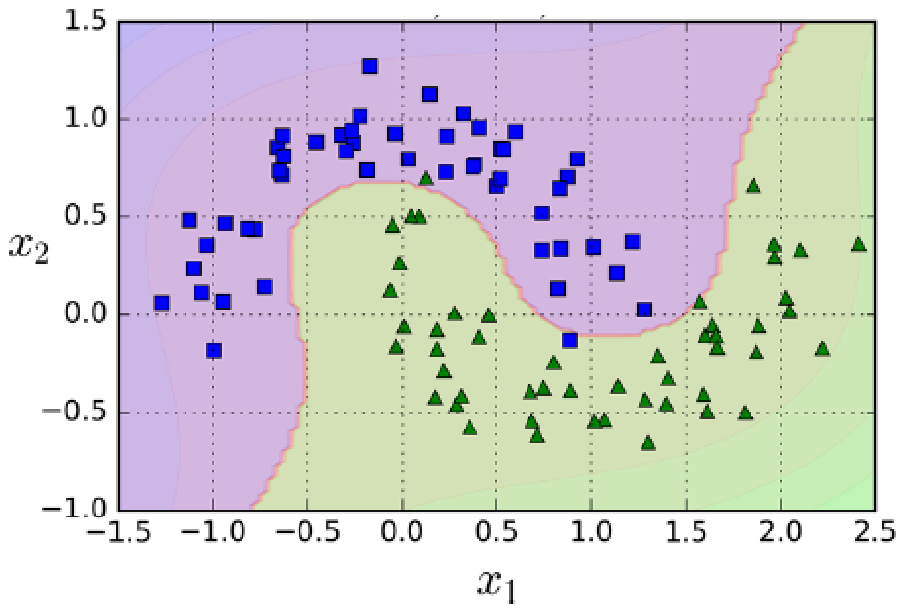
- Using mapping function Φ to transfer original data $\boldsymbol{x}$ into a higher dimensional space, then the transformed data $\Phi(\boldsymbol{x})$ might be separable
- It gives the same result *as if* you have added many similarity features, without actually adding them

Nonlinear SVM classification

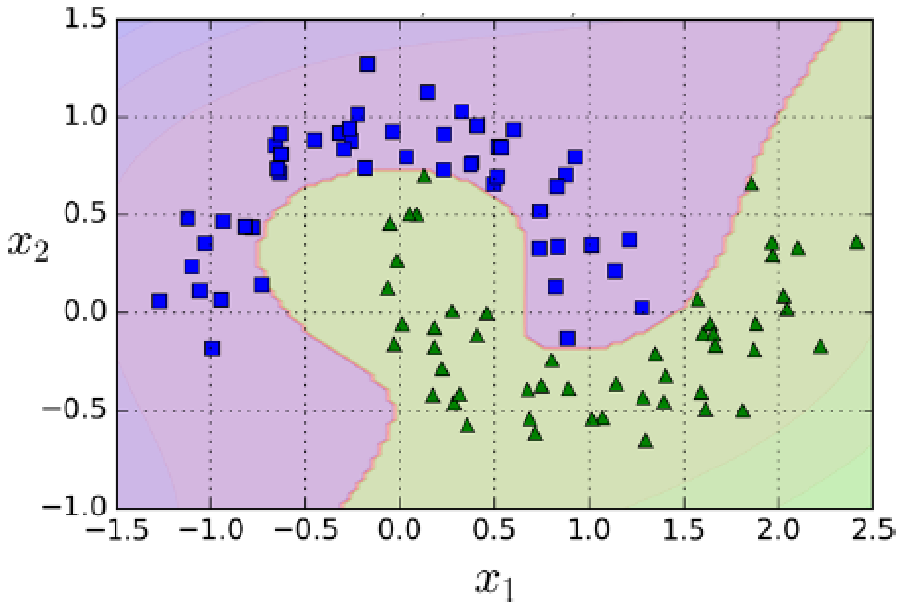
- When we make inverse transformation of the transformed data, the decision boundary become nonlinear



Polynomial kernel



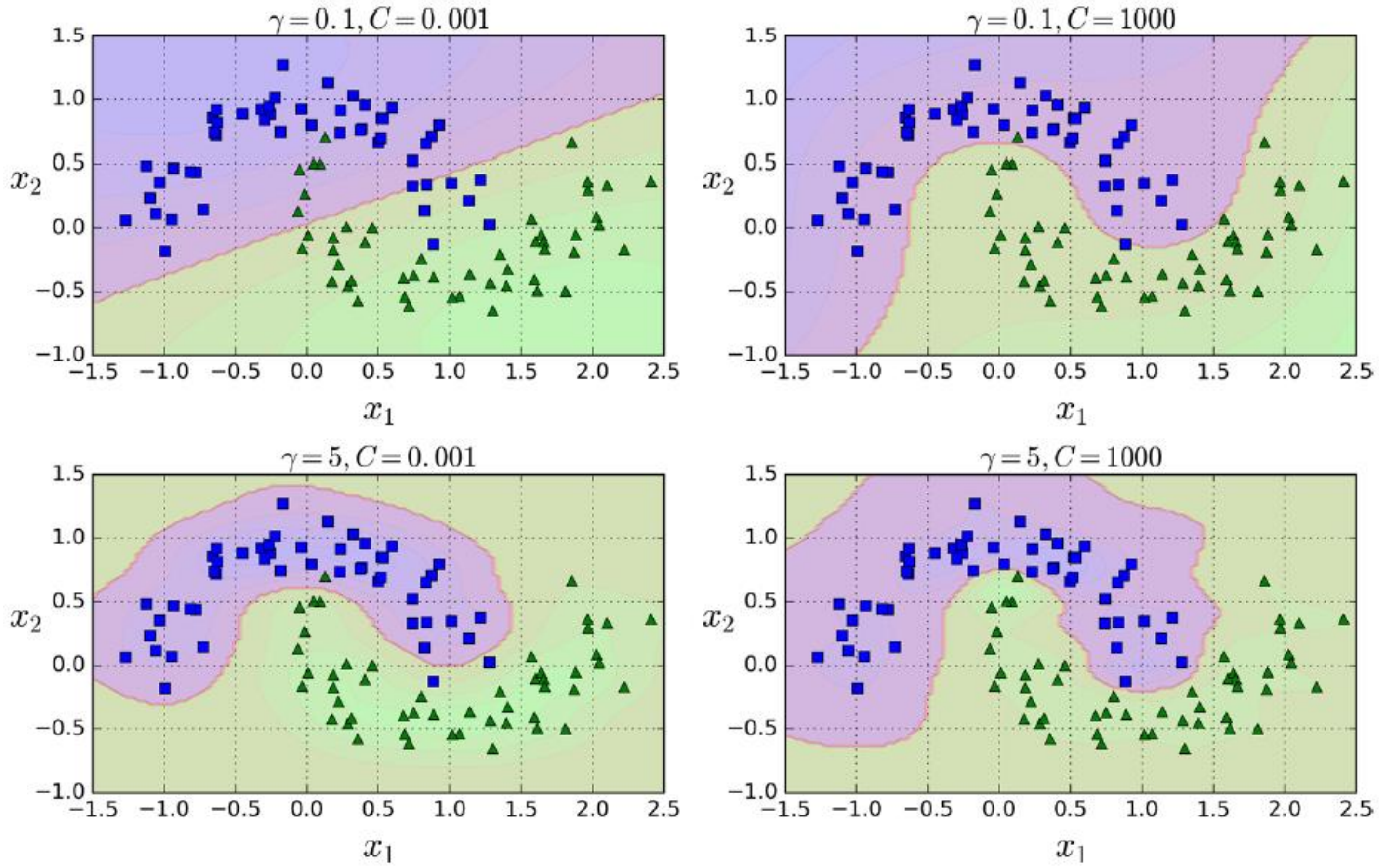
3rd -degree polynomial kernel



10rd -degree polynomial kernel

- Control the parameter of classifier (here, the degree of polynomial) in case of underfitting / overfitting

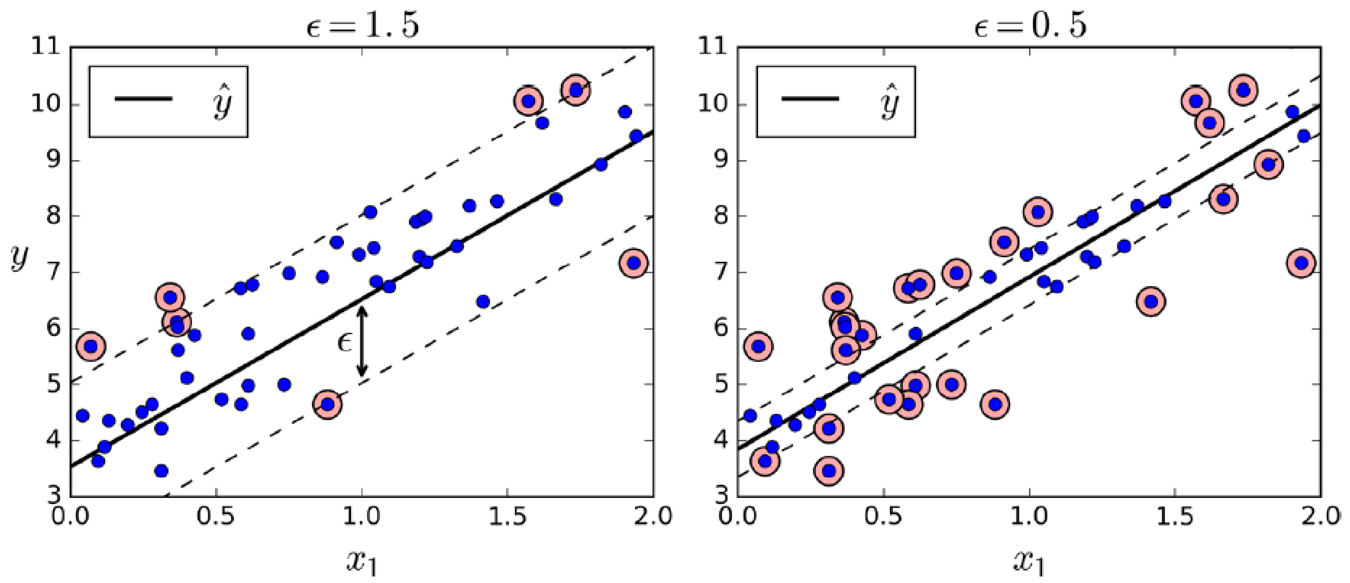
Gaussian RBF kernel



- C controls the bell-shape of curve
- γ acts like a regularization hyperparameter (underfitting / overfitting)

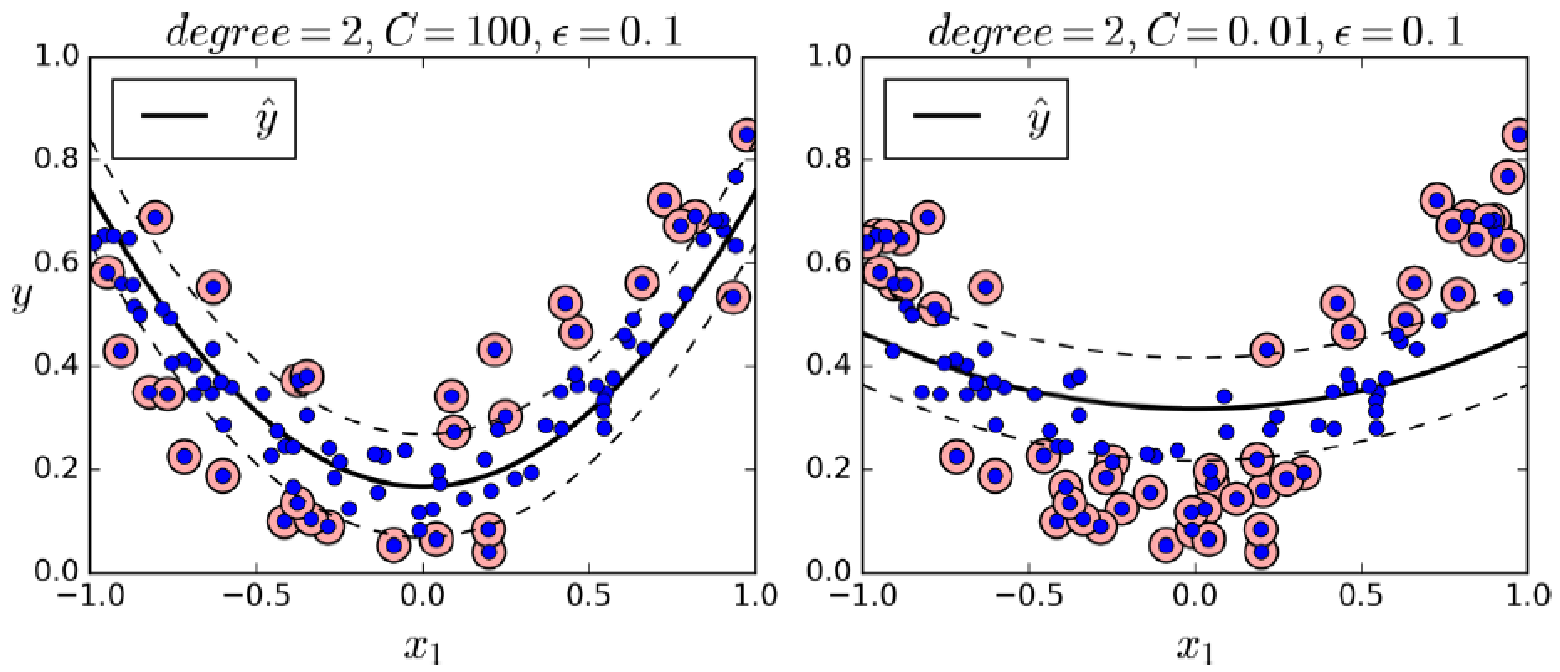
SVM regression

- Objective: To fit as many instances as possible *on* the street while limiting margin violations (i.e., instances *off* the street)
- ϵ, as the hyperparameter, controls the width of the street



Linear SVM regression

SVM Regression



2nd –degree polynomial regression

Applications of SVM

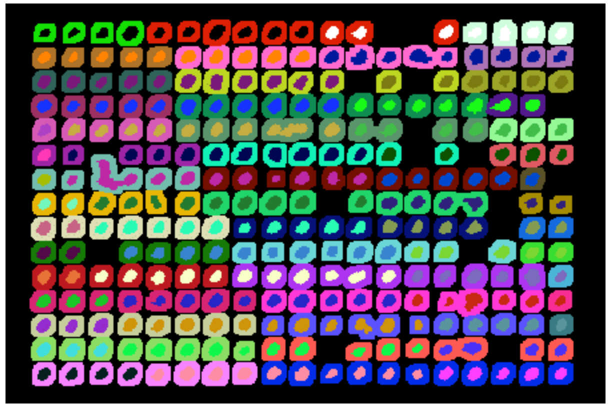
- SVM has been used successfully in many real-world problems:
  - Text (and hypertext) categorization
  - Image classification
  - Bioinformatics (protein classification, cancer classification)
  - Hand-written character recognition
  - …

Example: Cancer classification

- Scores on microarray represent intensity of gene expression after being re-scaled to make each chip equivalent
- **Dataset**: bone marrow samples with two types of labels:
  - acute lymphoblastic leukemia (ALL) (急性淋巴细胞白血病)
  - acute myeloid leukemia (AML) (急性髓细胞样白血病)

	Genes			
Samples	**g-1**	**g-2**	**……**	**g-d**
s-1				
s-2				
……				
s-n				

Cancer classification


Microarray Image File

training data

testing data

```
0.0   1:154 2:72 3:81 4:650 5:698 6:5199 7:1397 8:216 9:71 10:22
0.0   1:154 2:96 3:58 4:794 5:665 6:5328 7:1574 8:263 9:98 10:37
1.0   1:154 2:98 3:56 4:857 5:642 6:5196 7:1574 8:300 9:95 10:35
0.0   1:154 2:72 3:81 4:650 5:698 6:5199 7:1397 8:216 9:71 10:22
0.0   1:154 2:72 3:81 4:650 5:698 6:5199 7:1397 8:216 9:71 10:22
0.0   1:154 2:72 3:81 4:650 5:698 6:5199 7:1397 8:216 9:71 10:22
0.0   1:154 2:96 3:58 4:794 5:665 6:5328 7:1574 8:263 9:98 10:37
1.0   1:154 2:98 3:56 4:857 5:642 6:5196 7:1574 8:300 9:95 10:35
1.0   1:154 2:98 3:56 4:857 5:642 6:5196 7:1574 8:300 9:95 10:35
0.0   1:154 2:72 3:81 4:650 5:698 6:5199 7:1397 8:216 9:71 10:22
0.0   1:154 2:72 3:81 4:650 5:698 6:5199 7:1397 8:216 9:71 10:22
0.0   1:154 2:72 3:81 4:650 5:698 6:5199 7:1397 8:216 9:71 10:22
0.0   1:154 2:96 3:58 4:794 5:665 6:5328 7:1574 8:263 9:98 10:37
1.0   1:154 2:98 3:56 4:857 5:642 6:5196 7:1574 8:300 9:95 10:35
...
1.0   1:154 2:98 3:56 4:857 5:642 6:5196 7:1574 8:300 9:95 10:35
0.0   1:154 2:72 3:81 4:650 5:698 6:5199 7:1397 8:216 9:71 10:22
0.0   1:154 2:72 3:81 4:650 5:698 6:5199 7:1397 8:216 9:71 10:22
0.0   1:154 2:72 3:81 4:650 5:698 6:5199 7:1397 8:216 9:71 10:22
0.0   1:154 2:96 3:58 4:794 5:665 6:5328 7:1574 8:263 9:98 10:37
1.0   1:154 2:98 3:56 4:857 5:642 6:5196 7:1574 8:300 9:95 10:35
```

Labeled Data File

ALL/AML	gene$_1$: intensity$_1$	gene$_2$: intensity$_2$	gene$_3$: intensity$_3$ …
0.0	1:0.852272	2:0.273378	3:0.198784

an instance

Cancer classification

- Associates each feature vector of data ($\boldsymbol{X}_i$) with its known classification (y_i):

$$(\boldsymbol{X}_1, y_1), (\boldsymbol{X}_2, y_2), \ldots, (\boldsymbol{X}_n, y_n)$$

  - where each $\boldsymbol{X}_1$ is a *d*-dimensional vector of real numbers and each y_i is classification label 1 / 0
- Create an SVM model
- Train it on training data, i.e. select the best model parameters, in order to obtain the best classification accuracy
- Then, test this model on test data

Weakness of SVM

- It is sensitive to noise
  - A small number of mislabeled examples can dramatically decrease the performance

- It only considers two classes
  - How to do multi-class classification with SVM?

  1) For m classes, learn m SVM’s
    - SVM 1 learns “Output==1” vs “Output != 1”
    - SVM 2 learns “Output==2” vs “Output != 2”
    - :
    - SVM m learns “Output==m” vs “Output != m”

  2)To predict the output for a new input, just predict with each SVM and find out which one puts the prediction the furthest into the positive region

Summary

- With regression as an example, we learned how to train machine learning models
  - Cost function minimization
  - Gradient Descent
- Techniques to tackle overfitting
  - Regularization
  - Learning curve
- Bayesian statistics (including Bayesian networks) can integrate data with prior knowledge for probabilistic modeling and inference
- SVM is a powerful machine learning technique when the datasets are not too big
- Read A. Geron’ s book “Hands-On Machine Learning with Scikit-Learn & TensorFlow” , Chapters 4 – 5