机器学习基础

南开大学金融学院
赵博
2025

什么是机器学习?

- The science (and art) of programming computers so they can learn from data.

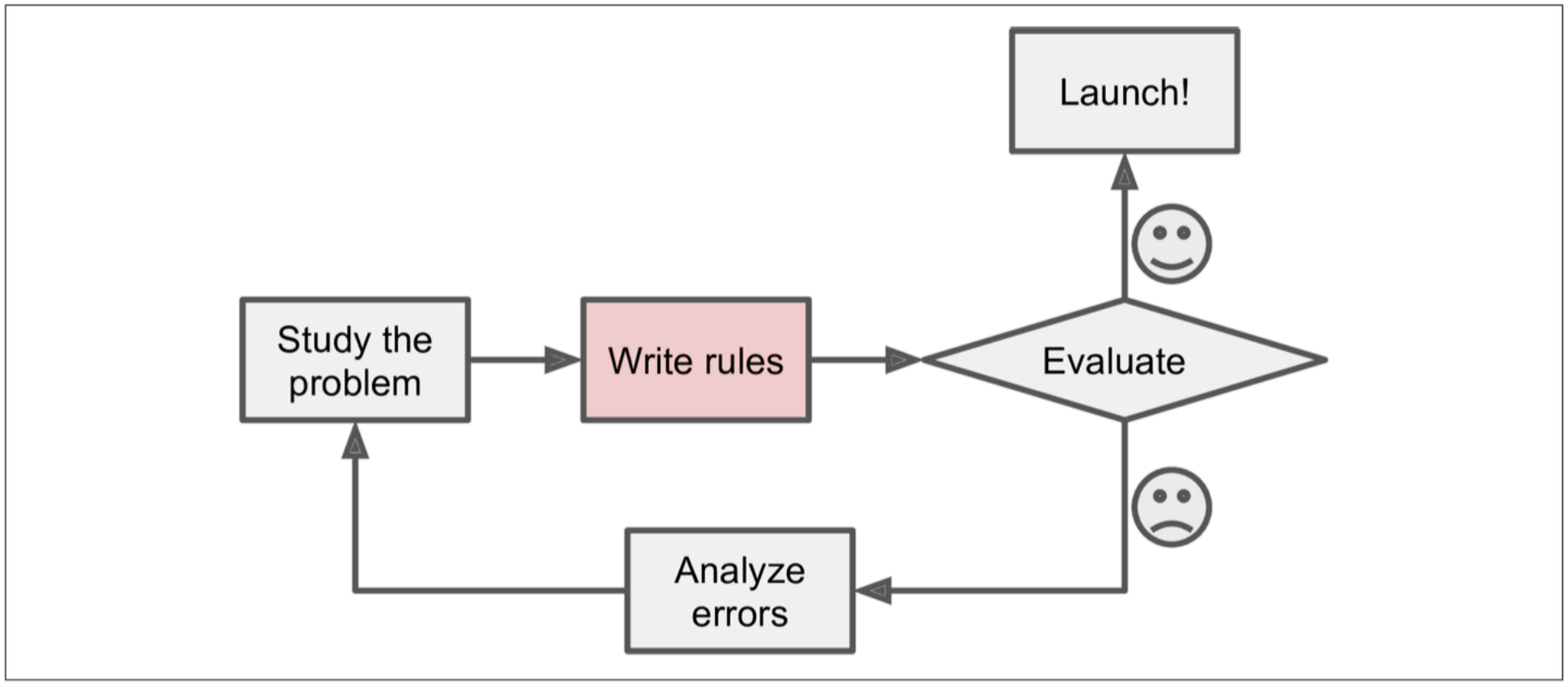
- [Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed. —Arthur Samuel, 1959

- A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E. — Tom Mitchell, 1997

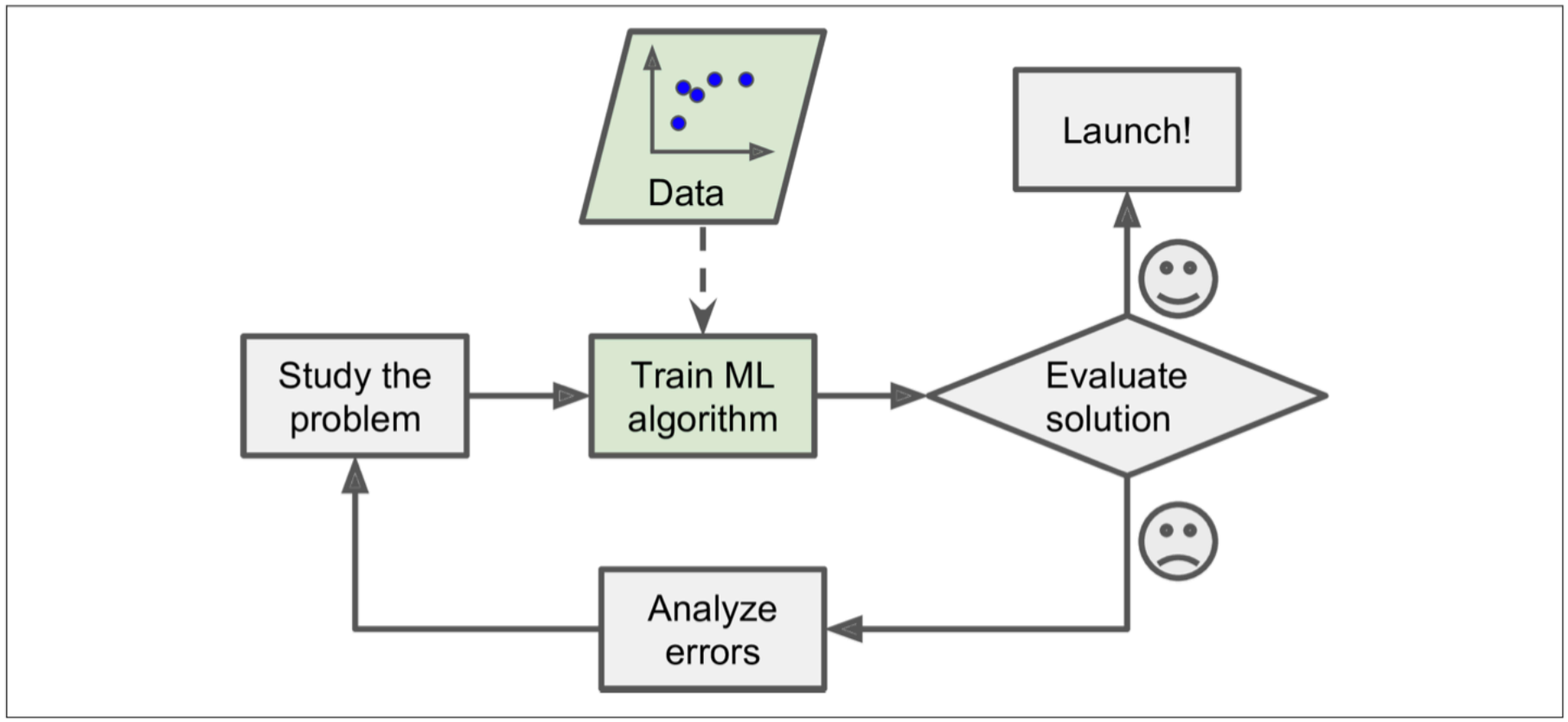
机器学习 vs. 传统人工智能

- 传统人工智能：编程制定规则
- 机器学习：定义目标（loss function，比如最小化MSE），计算机从数据中学习规则
- 例：识别垃圾邮件。
  - 传统人工智能：观察垃圾邮件特点，比如常见词：4U, free, amazing, 优惠，限时，抽奖…，建模。
  - 机器学习：标注大量垃圾邮件和非垃圾邮件，而非直接对邮件特点建模。自动识别垃圾邮件的特点（在给定的模型空间里选择模型）

- 传统人工智能：


Launch!
Study the problem
Write rules
Evaluate
Analyze errors

- 机器学习：


Data
Launch!
Study the problem
Train ML algorithm
Evaluate solution
Analyze errors

什么是计量经济学?

- 提出模型（根据常识、经济学、金融学理论，也即 field knowledge）
- 根据模型的特点设计估计方法（Robust errors、GLS、IV、Structural models, GMM、Panel、DID、PSM…）
- 对模型的结果进行评价
    - 首先要符合常识

> It should be borne in mind that despite the power, or lack thereof, of hypothesis tests, often conclusions are convincing to a researcher only if supported by personal experience. Nelson (1995, p. 141) captures this subjective element of empirical research by noting that "what often really seems to matter in convincing a male colleague of the existence of sex discrimination is not studies with 10000 'objective' observations, but rather a particular single direct observation: the experience of his own daughter."

    - 其次，研究者想表达的故事到底面临哪些问题?
- 现代计量经济学经历了"因果识别"的革命。
    - 本质上：从 structural form —> reduced form
- 模型本身到底对还是不对？无法从数据给出的结果中知道

What distinguishes an econometrician from a statistician is the former's preoccupation with problems caused by violations of statisticians' standard assumptions; owing to the nature of economic relationships and the lack of controlled experimentation, these assumptions are seldom met. Patching up statistical methods to deal with situations frequently encountered in empirical work in economics has created a large battery of extremely sophisticated statistical techniques. In fact, econometricians are often accused of using sledgehammers to crack open peanuts while turning a blind eye to data deficiencies and the many questionable assumptions required for the successful application of these techniques. Valavanis has expressed this feeling forcefully:

> Econometric theory is like an exquisitely balanced French recipe, spelling out precisely with how many turns to mix the sauce, how many carats of spice to add, and for how many milliseconds to bake the mixture at exactly 474 degrees of temperature. But when the statistical cook turns to raw materials, he finds that hearts of cactus fruit are unavailable, so he substitutes chunks of cantaloupe; where the recipe calls for vermicelli he uses shredded wheat; and he substitutes green garment die for curry, ping-pong balls for turtle's eggs, and, for Chalifougnac vintage 1883, a can of turpentine. (Valavanis, 1959, p. 83)

How has this state of affairs come about? One reason is that prestige in the econometrics profession hinges on technical expertise rather than on the hard work required to collect good data:

> It is the preparation skill of the econometric chef that catches the professional eye, not the quality of the raw materials in the meal, or the effort that went into procuring them. (Griliches, 1994, p. 14)

机器学习 vs. 计量经济学

- Arthur Lewbel:

  机器学习（Machine Learning），也称“统计学习”（Statistical Learning），被广泛的应用在学科和业界之中，而计量经济学几乎不出校门，这两者究竟有什么关联和区别，Lewbel教授为我们做出了解答。

  > “机器学习在统计学中非常流行，机器学习的主要功能是它揭示了事物之间的联系，因此有大量的变量。机器学习会告诉我们哪些变量互相联系，因此机器学习的目标是确定相关性。更一般地讲，机器学习不会告诉我们因果关系，也不会告诉为什么会有这样的因果关系。通过计量经济学，我们想知道的不仅仅是相关性，还有因果关系，包括是什么样的因果关系，为什么会有这种因果关系，这种因果关系如何产生。因此，机器学习找到了两个变量之间关系模式，而计量经济学试图找出因果关系，并试图理解这些模式。”

近年来，计量经济学有了新的热点，Lewbel教授就机器学习与人际关系这两个方面进行了简要的梳理。

> **“计量经济学有几个非常热门的领域，其中一个领域涉及机器学习，涉及尝试使用机器学习方法进行计量经济学的研究。因此，可以采用多阶段研究，其中将机器学习应用在某一个阶段中，而在另一个阶段使用常见的计量经济学方法。因此，将机器学习作为一个工具，作为对计算经济学模型研究的输入，是一个热门领域。**
>
> **另一个非常热门的领域涉及人际关系的研究，主要是网络模型、社会交际、网络空间模型。这类模型是研究人类行为的互相影响，在应用计量经济学和计量经济学理论中都是非常流行的领域，其中包括网络如何形成，以及它们会产生什么样的结果。**
>
> **这个领域之前被研究得比较少，而在最近有爆发式增长。近来有许多新的研究已经完成，也有新的结果被发现。”**

- Tom Sargent: http://www.elecfans.com/d/748807.html

  我怎么讲到这儿来了呢？人工智能首先是一些很华丽的辞藻。人工智能其实就是统计学，只不过用了一个很华丽的辞藻，其实就是统计学。好多的公式都非常老，但是我们说所有的人工智能利用的都是统计学来解决问题。有一些新发展，过去二三十年，今天我们统计学的完成质量更高，首先电脑运算速度更快，有新的算法，好多是源于物理学发展，还有统计力学等等，这是过去3到4年从物理学拿来，加速了我们做系列统计学方面的进步。

- 金榕：https://tech.sina.com.cn/it/2018-08-20/doc-ihhxaafz3934293.shtml

  在接受澎湃新闻专访时，他表示：“这位宏观经济学家也许是习惯了用统计去寻找经济的因果关系，因而认为AI也是这样。即使是他提到的动态规划（dynamic programming），也不属于统计学范畴。除了统计，AI中的‘学习’、‘推理’和‘决策’中还使用了代数、逻辑、最优化等许多其他学科知识与方法。此外，有了算法后如何有效实现也非常重要。所以单纯说AI就是统计学，或者说“所有的AI都是利用统计学来解决问题的”都是片面和不准确的。”

Susan Athey:

Key Lessons for Econometrics

Many problems can be decomposed into predictive and causal parts

- Can use off-the-shelf ML for predictive parts

Data-driven model selection

- Tailored to econometric goals
  - Focus on parameters of interest
  - Define correct criterion for model
  - Use data-driven model selection where performance can be evaluated
- While retaining ability to do inference

ML-Inspired Approaches for Robustness

Validation

- ML always has a test set
- Econometrics can consider alternatives
  - Ruiz, Athey and Blei (2017) evaluate on days with unusual prices
  - Athey, Blei, Donnelly and Ruiz (2017) evaluate change in purchases before and after price changes
  - Tech firm applications have many A/B tests and algorithm changes

Other computational approaches for structural models

- Stochastic gradient descent
- Variational Inference (Bayesian models)

See Sendhil Mullainathan et al (JEP, AER) for key lessons about prediction in economics

- See also Athey (Science, 2017)

Empirical Economics in Five Years: My Predictions

Regularization/data-driven model selection will be the standard for economic models

Prediction problems better appreciated

Measurement using ML techniques an important subfield

Textual analysis standard (already many examples)

Models will explicitly distinguish causal parts and predictive parts

Reduced emphasis on sampling variation

Model robustness emphasized on equal footing with standard errors

Models with lots of latent variables

- Hal Varian:

> In fact, my standard advice to graduate students these days is go to the computer science department and take a class in machine learning. There have been very fruitful collaborations between computer scientists and statisticians in the last decade or so, and I expect collaborations between computer scientists and econometricians will also be productive in the future.

-

- 我的看法：
  - 机器学习不完全是只看相关性而不看因果联系。
  - 机器学习越来越关注因果联系。https://www.microsoft.com/en-us/research/group/causal-inference/
  - 以前的计量经济学时常混淆因果问题与预测问题，现在逐渐清晰
  - 因果联系问题本身极其复杂，经济学的问题本身很难被证伪
  - 现在的LLM(ChatGPT等)是否有“独立”意识?
  - 注意“基准”
  - 机器学习和计量经济学的最大差别在于：怎样评估结果

评估模型好坏

- 计量经济学：对模型本身的评价只能由field knowledge而来。样本内的显著性。样本外的评估比较困难。
- 机器学习：模型在未见过的新数据上表现怎么样?
  - 关键假设：训练数据和未见过的新数据性质相同（iid, stationarity, etc.）
  - 如果data generating process改变，则已有的模型会失效

- Testing and Validating
    - Split the data into Training Set and Test Set.
    - Within Training Set, further split the data into Reduced Training and Validation Sets. Why?
    - Train on the Reduced Training Sets, tune hyperparameters on the Validation Set
    - Then train on the full training set
    - Apply the model to the test set

- Types
    - Holdout
    - K-Fold
    - Leave-one-out
- Time series
- Grid search
- Testing and Validating is tricky when the DGP is ever changing.

用词差别

- Econometrics vs. ML
- Control variables, Covariates vs. Features
- Estimation vs. Training
- …

机器学习类别

- 是否有人类监督（打标签）？
  - Supervised，Unsupervised, semisupervised, Reinforcement Learning
- 批量学习(batch learning)还是即时学习(online learning)?
- 以上分类可有交集

Supervised vs. Unsupervised

- Supervised learning: human labeled desired solutions.
    - Classification: spam or ham
    - Regression: numerical value is labeled
- Unsupervised learning: no human attached labels.
    - Clustering: detect groups
    - dimensionality reduction: simplify the data without losing too much information
    - Anomaly detection: detecting unusual credit card transactions
    - Association rule learning: supermarket transactions

Semisupervised learning

- Usually lots of unlabeled data, a little bit of labeled data.
- People in photos: detecting people in photos (unsupervised), needs you to tell it who them are (supervised)

Reinforcement learning

- The ‘agent’ (learning system), observes the ‘environment’, perform an ‘action’, and get ‘rewards’.
- must learn a series of ‘actions’ under different ‘environments’, where the ‘actions’ might affect the ‘environment’ in which the ‘agent’ is in.
- Not necessarily deep

Batch and Online Learning

- Batch learning: the model must be trained using all available data. When launched, it cannot learn new data.
    - If you want to update the system, you need to train once again with all data.

- Online learning: feed new data sequentially, individually or by small groups (mini-batches)

    - learning rate: how fast should the machine learn?

Bias vs. Variance, Underfit vs. Overfit

- MSE = Bias^2 + Variance + Irreducible error
- Some confusions in the decomposition
  - In econometrics:

In econometrics, when estimating a parameter β, β is regarded as some fixed unknown value. In other words, β is non-random. Therefore, the MSE could easily be decomposed as the Bias Square plus the Variance of the estimator, $\hat{\beta}$. Just write the MSE as

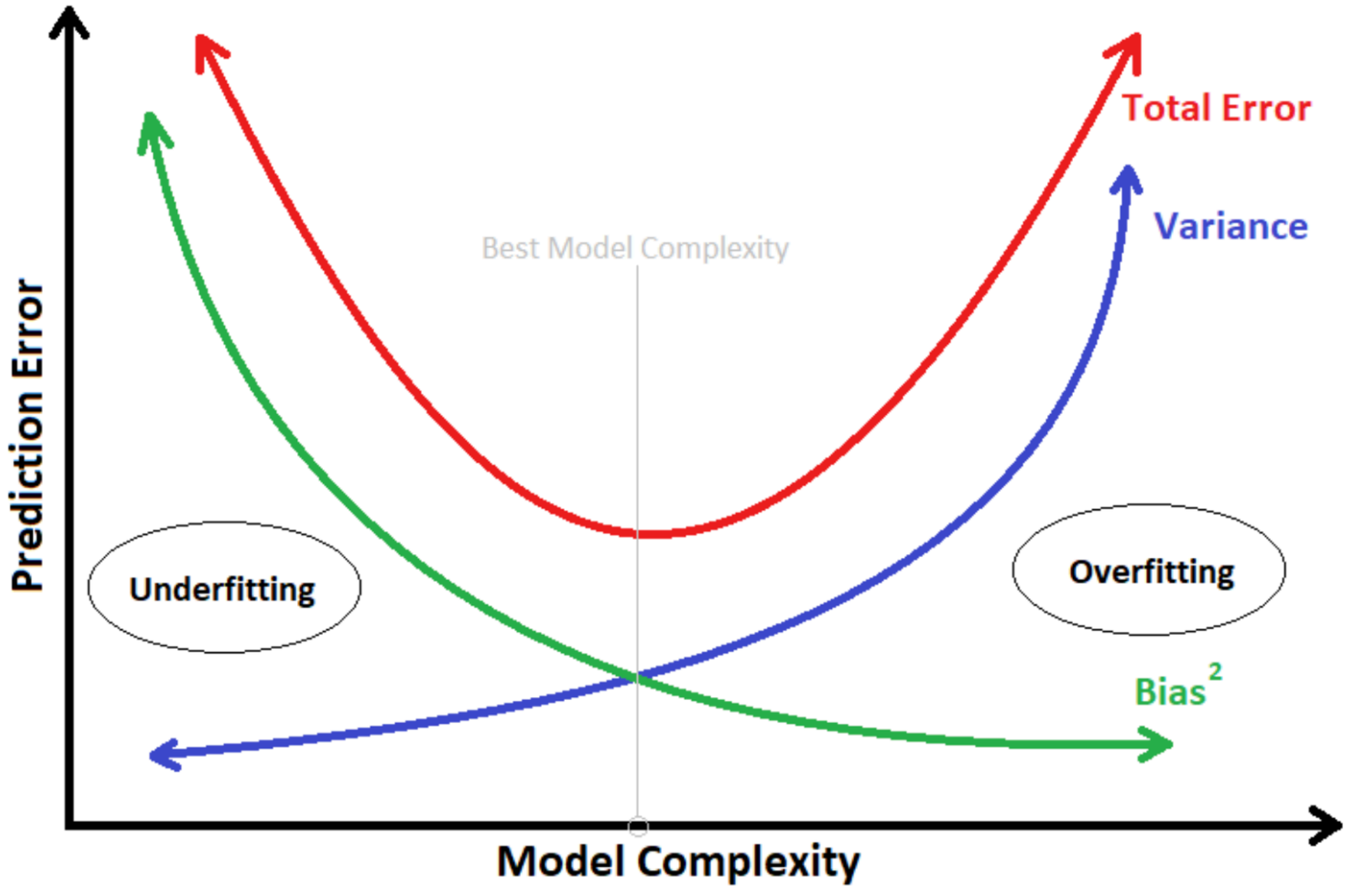
$$\mathrm{E}\left[\left(\beta-\hat{\beta}\right)^2\right]=\mathrm{E}\left[\left(\beta-\mathrm{E}(\hat{\beta})+\mathrm{E}(\hat{\beta})-\hat{\beta}\right)^2\right], \tag{1}$$

- in ML:

When doing machine learning, the problem is that we have a RANDOM variable $y = f(x) + \epsilon$, and an estimate of $f(x)$, the $\hat{f}(x)$. Now the Mean Square Error is defined differently, which is the difference between y and $f(x)$, $\mathrm{E}[(y-\hat{f})^2]$. Now we could write y as $f(x)+\epsilon$, so the MSE here is $\mathrm{E}[(f-\hat{f}+\epsilon)^2]$. The first part $f - \hat{f}$, as in the above, could be decomposed as the Bias Square and Variance. And the last part, the independent ϵ, is the extra part. That's why we see an extra $\mathrm{Var}(\epsilon)$ in the machine learning literature.

The confusion actually lies in the different definition of MSE. In econometrics, it is defined as the mean square difference between f and $\hat{f}$ (using the machine learning notation), while in machine learning, it is defined as y and $\hat{f}$.

- Bias: wrong assumptions of the set of models, e.g. assuming linear model while it is actually nonlinear. Underfit.
    - In econometrics, I think it's very likely that in many cases the model is misspecified; However, a linear projection is not necessarily "wrong". It can still "explain" as a linear approximation, which is in line with mundane intuition.
- Variance: model's excessive sensitivity to small variations in the training data.
- Irreducible error: randomness of the data.


Prediction Error
Total Error
Variance
Best Model Complexity
Underfitting
Overfitting
Bias^2
Model Complexity

The workflow

- Get data
- Exploratory Data Analysis
    - Correlations
    - Plots
        - Clean data
- Define validation strategy
- Feature engineering
- Modelling
- Ensembling

Training models

- Gradient Descent
- Batch Gradient Descent
- Stochastic Gradient Descent
- Mini-batch Gradient Descent
- …

Gradient descent

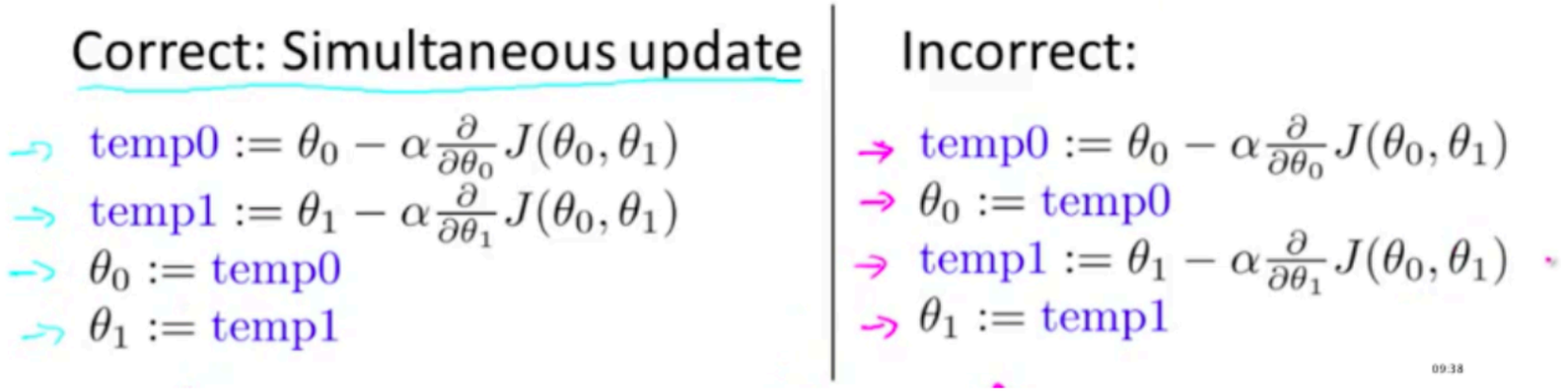
- $\widehat{\boldsymbol{\theta}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$

- The computation complexity of inverting an K*K matrix is about $O(K^{2.4})$ to $O(K^3)$

- Gradient descent:

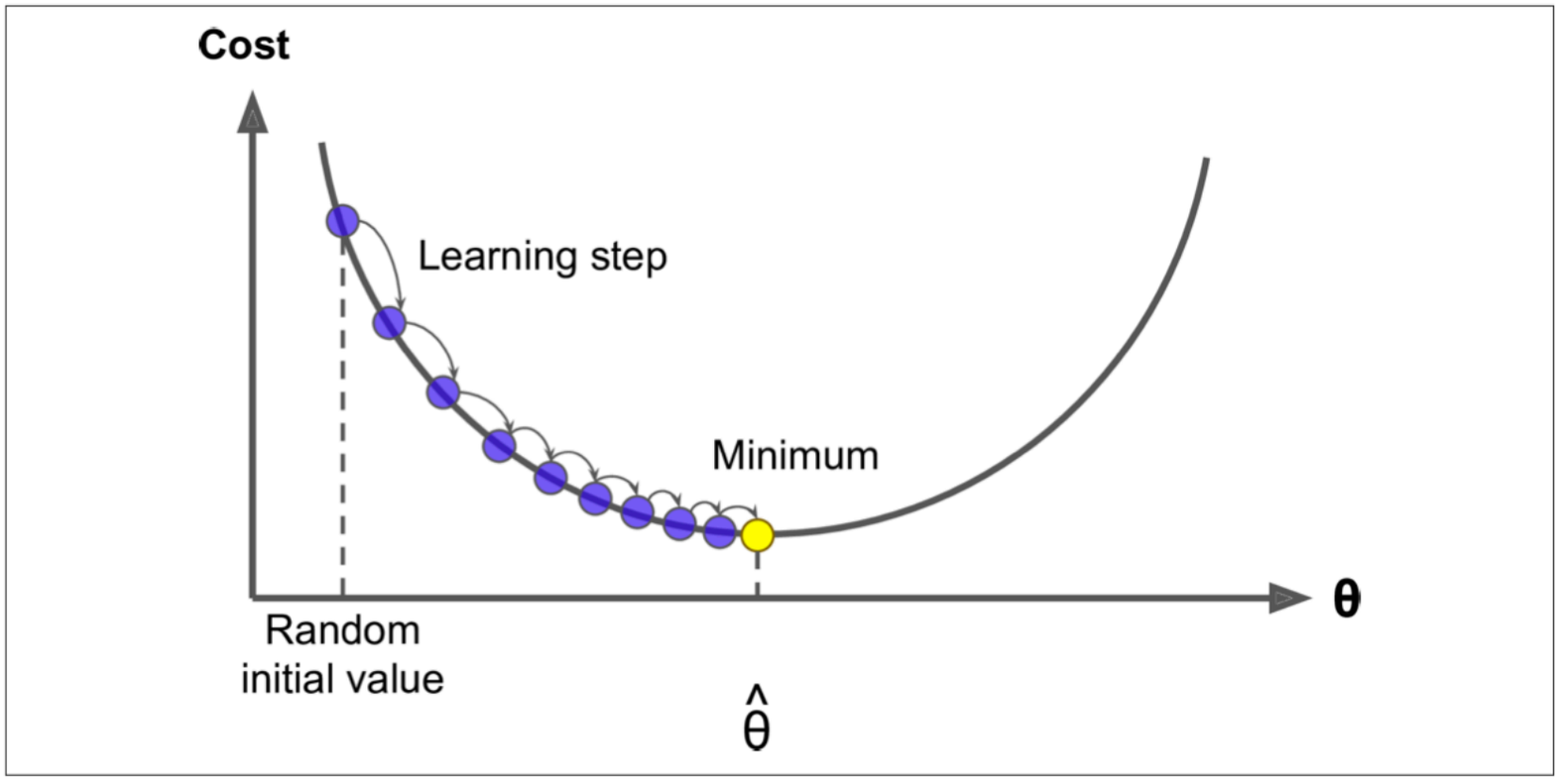
  - initialize $\widehat{\beta}$ with random values

  - improve it bit by bit recursively, until the object function (MSE) reaches the minimum

- $\hat{\theta}_j^{next} := \hat{\theta}_j - \alpha \frac{\partial}{\partial \hat{\theta}_j} J\left(\hat{\theta}_0, \hat{\theta}_1\right)$
- 

- $\hat{\boldsymbol{\theta}}^{\text{next}} = \hat{\boldsymbol{\theta}} - \alpha \nabla_{\boldsymbol{\theta}} \text{MSE}(\hat{\boldsymbol{\theta}})$

Intuition



- *Figure 4-3. Gradient Descent*

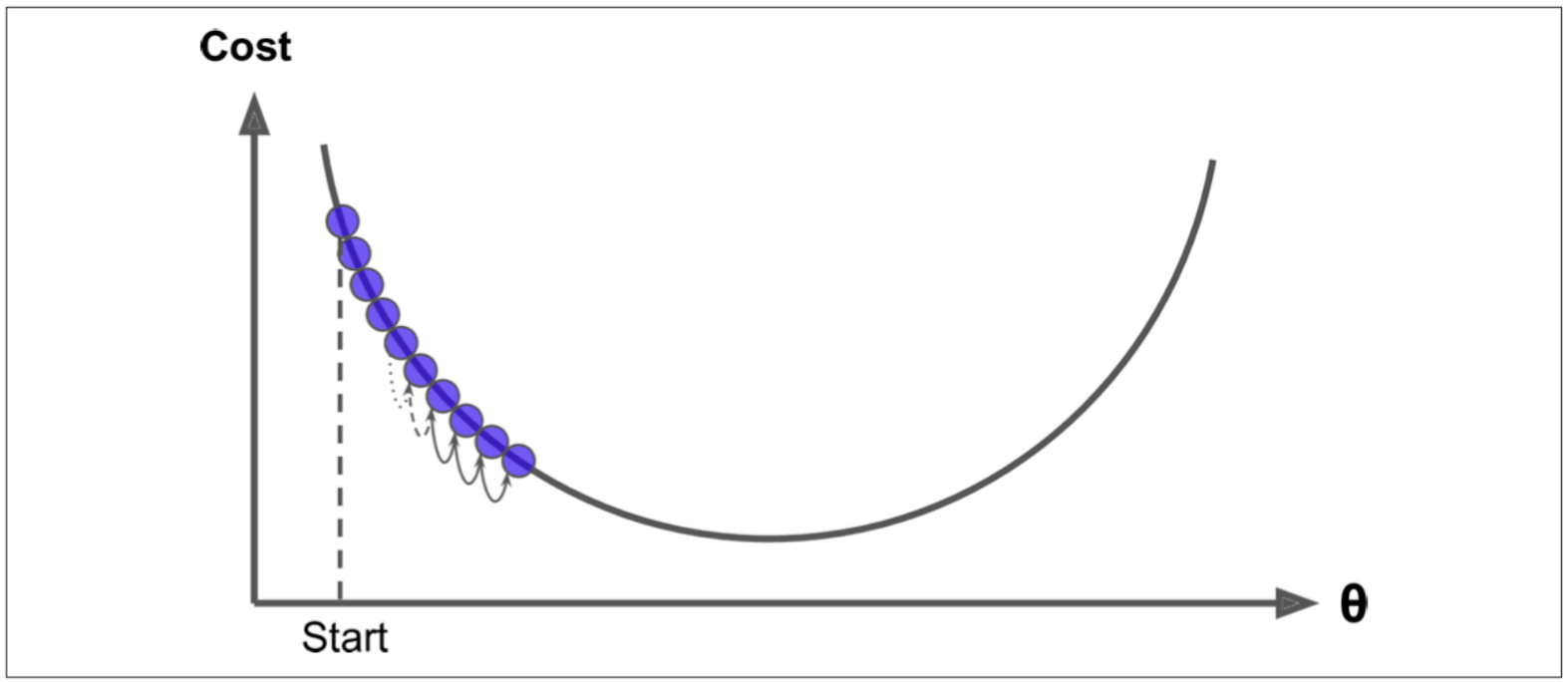
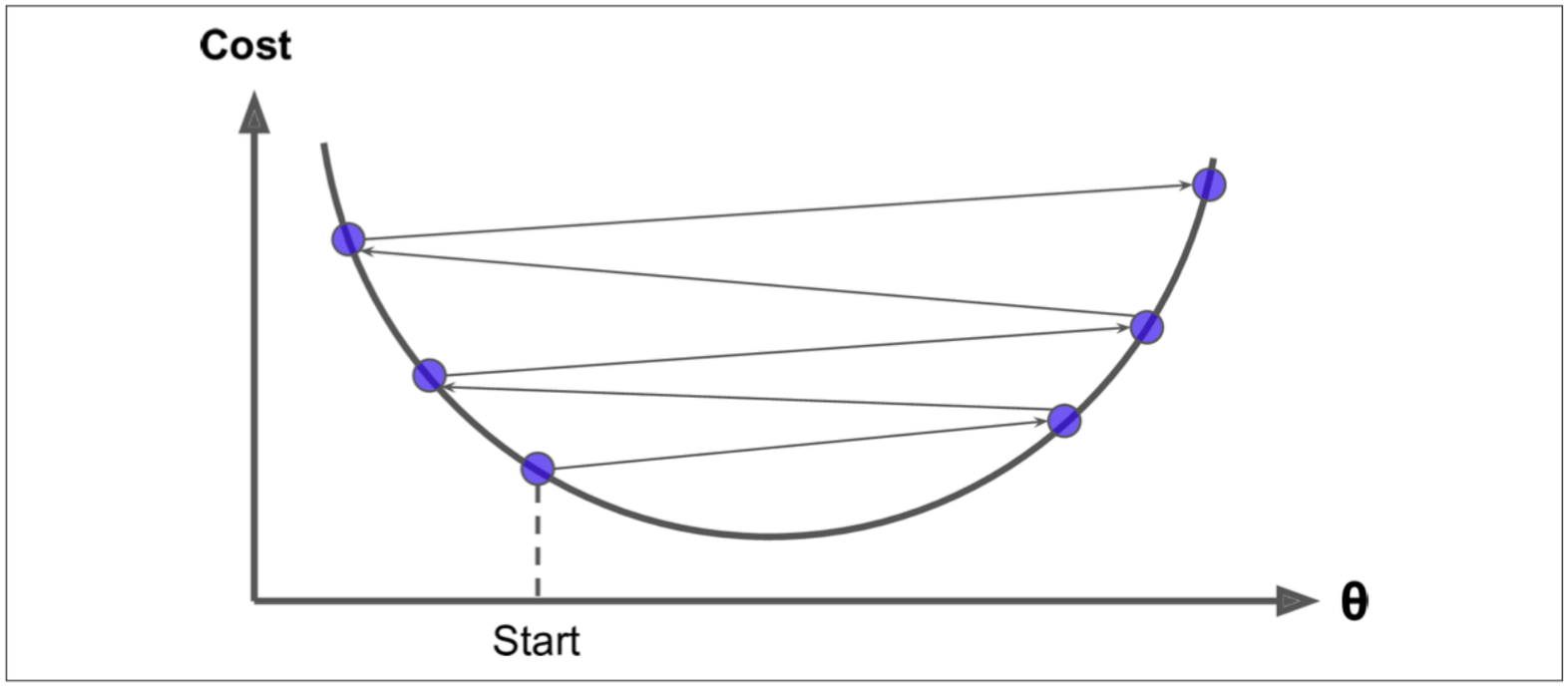


Figure 4-4. Learning rate too small



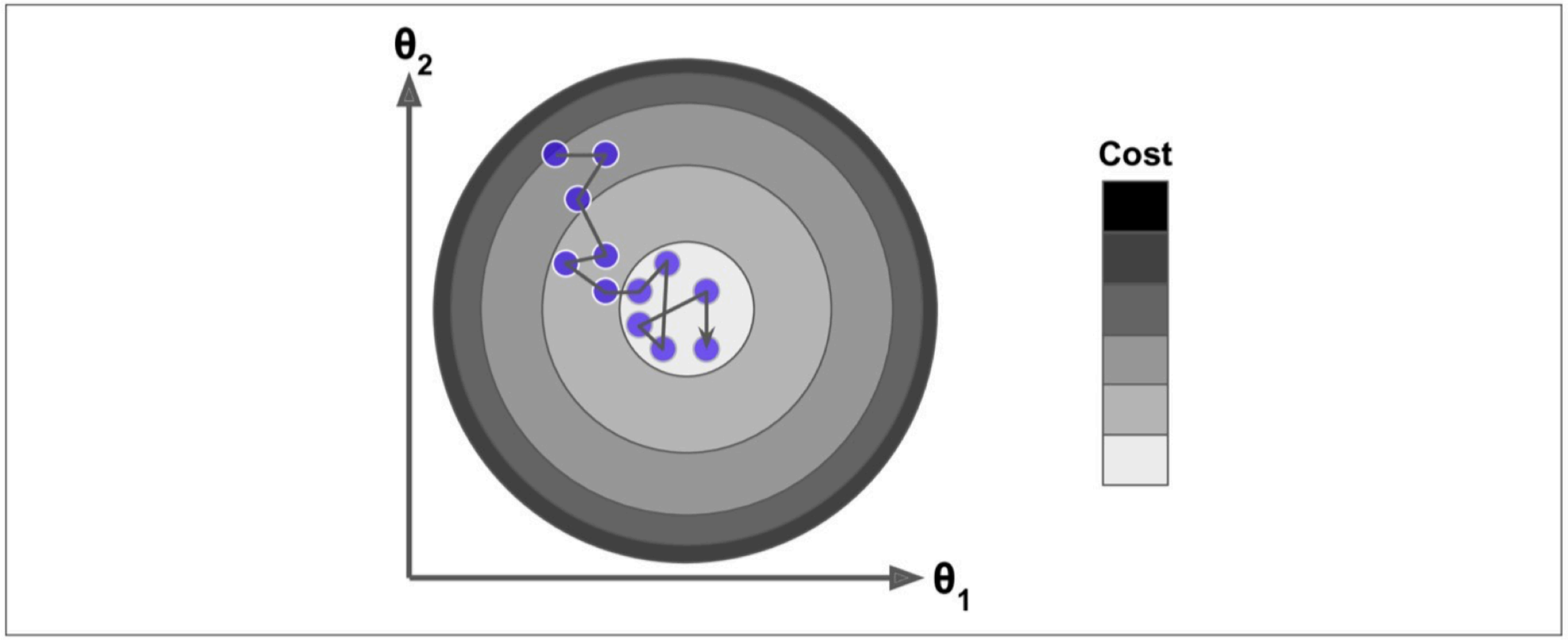
- *Figure 4-5. Learning rate too large*

Batch GD

- $\text{MSE}(\boldsymbol{X}, \boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \left(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)} \right)^2$
- $\frac{\partial}{\partial \theta_j} \text{MSE}(\boldsymbol{\theta}) = \frac{2}{m} \sum_{i=1}^{m} \left(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)} \right) x_j^{(i)}$
- $$\nabla_\theta \text{MSE}(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial}{\partial \theta_0} \text{MSE}(\boldsymbol{\theta}) \\ \frac{\partial}{\partial \theta_1} \text{MSE}(\boldsymbol{\theta}) \\ \vdots \\ \frac{\partial}{\partial \theta_n} \text{MSE}(\boldsymbol{\theta}) \end{bmatrix} = \frac{2}{m} \mathbf{X}^T (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$
- $\widehat{\boldsymbol{\theta}}^{\text{next}} = \widehat{\boldsymbol{\theta}} - \alpha \nabla_{\boldsymbol{\theta}} \text{MSE}(\widehat{\boldsymbol{\theta}})$
- When updating EACH $\boldsymbol{\theta}$, we have looked at ALL m observations ($\mathbf{X}^T (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$), and taking average ($\frac{1}{m}$). In other words, we use the whole BATCH of training data at every step. SLOW if m is large.

Stochastic Gradient Descent

- Randomly pick one instance (observation) in the training set
- Compute the gradients based only on this single instance
- $$\nabla_\theta \text{MSE}(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial}{\partial \theta_0} \text{MSE}(\boldsymbol{\theta}) \\ \frac{\partial}{\partial \theta_1} \text{MSE}(\boldsymbol{\theta}) \\ \vdots \\ \frac{\partial}{\partial \theta_n} \text{MSE}(\boldsymbol{\theta}) \end{bmatrix} = 2\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}\right)\boldsymbol{x}^{(i)}$$
- Update $\widehat{\boldsymbol{\theta}}$


θ2
θ1
Cost

Mini-batch Gradient Descent

- Each round of updating, look at a small random set of instances called Mini-Batch.
- Some notations that might be a bit confusing:
    - iteration: each time you update your coefficient estimates
    - batch-size: the size of the instances you choose to look at before updating
    - epoch: One epoch means you have looked at all samples once. But it does not necessarily equal to m, it could be a number defined by yourself.
- Example: 1000 instances
    - Divide into 10 batches
    - batch-size = 100
    - 1 iteration: update $\widehat{\boldsymbol{\theta}}$ once, i.e. uses 100 instances
    - 1 epoch: when you have trained the model on all 10 batches

Models

- No Free Lunch Theorem
    - model: a simplified version of descriptions
    - the simplifications are meant to discard superfluous details that are unlikely to generalize to new instances
        - to decide what to be kept, you make assumptions
- Wolpert (1996): if you make absolutely no assumption about the data, then there is no model dominating other models.
    - the only way to know for sure which model is best is to test them all.
    - But this is impossible.
    - So you have to make assumptions, and evaluate those models that seems to fit these assumptions

Popular models

- State of the art:
    - Linear models: good for sparse high dimensional data
    - Tree-based models: very powerful, often to be the default method for tabular data
    - Ensembling
    - Image, video, language: Neural Networks (Deep Learning)

Model Regularization

- A good way to fight against overfitting: constrain the model in some way.

- Linear Model Examples:

- Ridge Regression: $J(\boldsymbol{\theta}) = \mathrm{MSE}(\boldsymbol{\theta}) + \alpha \frac{1}{2} \sum_{i=1}^{n} \theta_i^2$. Use this loss function during training. Afterwards, use the original linear model(The loss and the evaluation metrics are different). α: hyperparameter

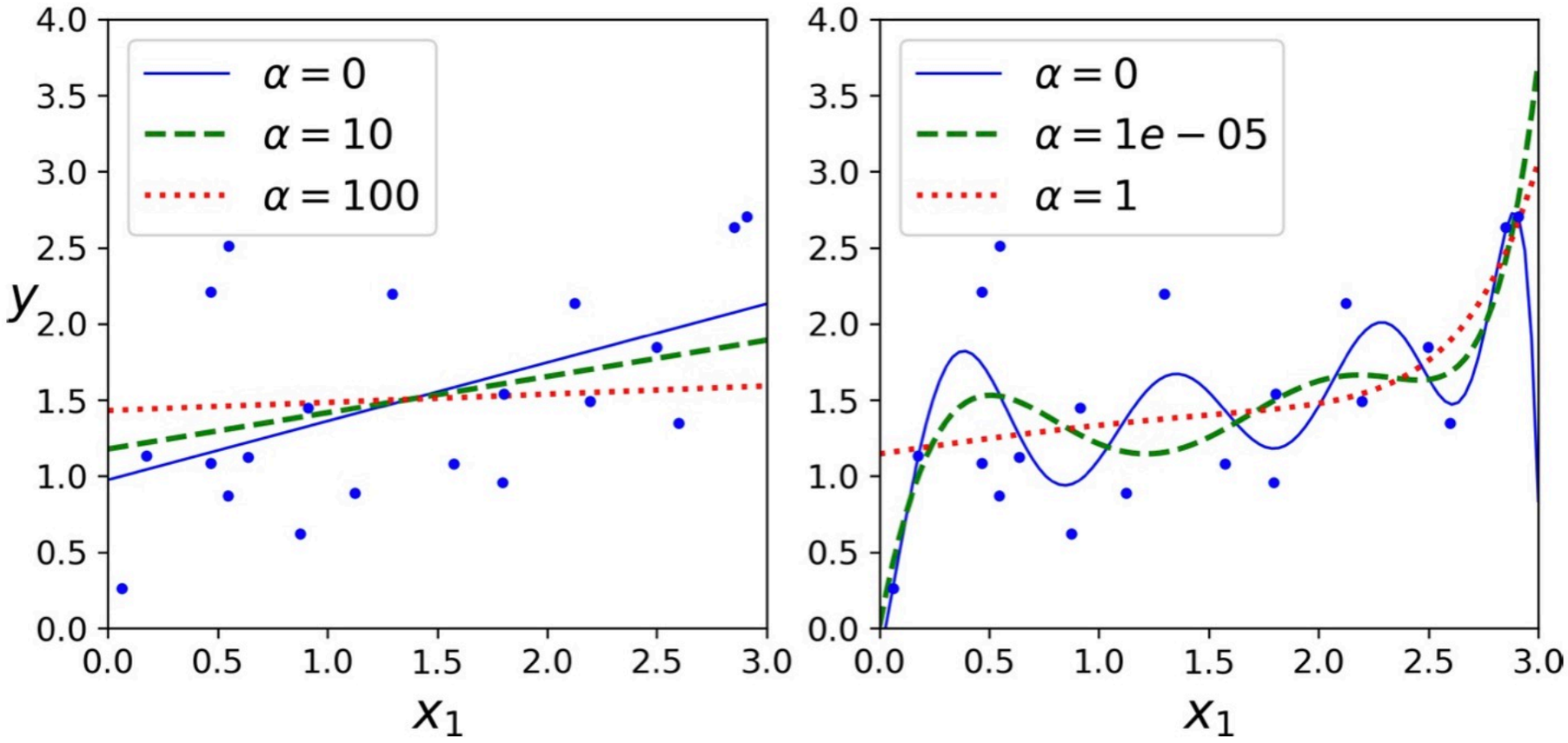


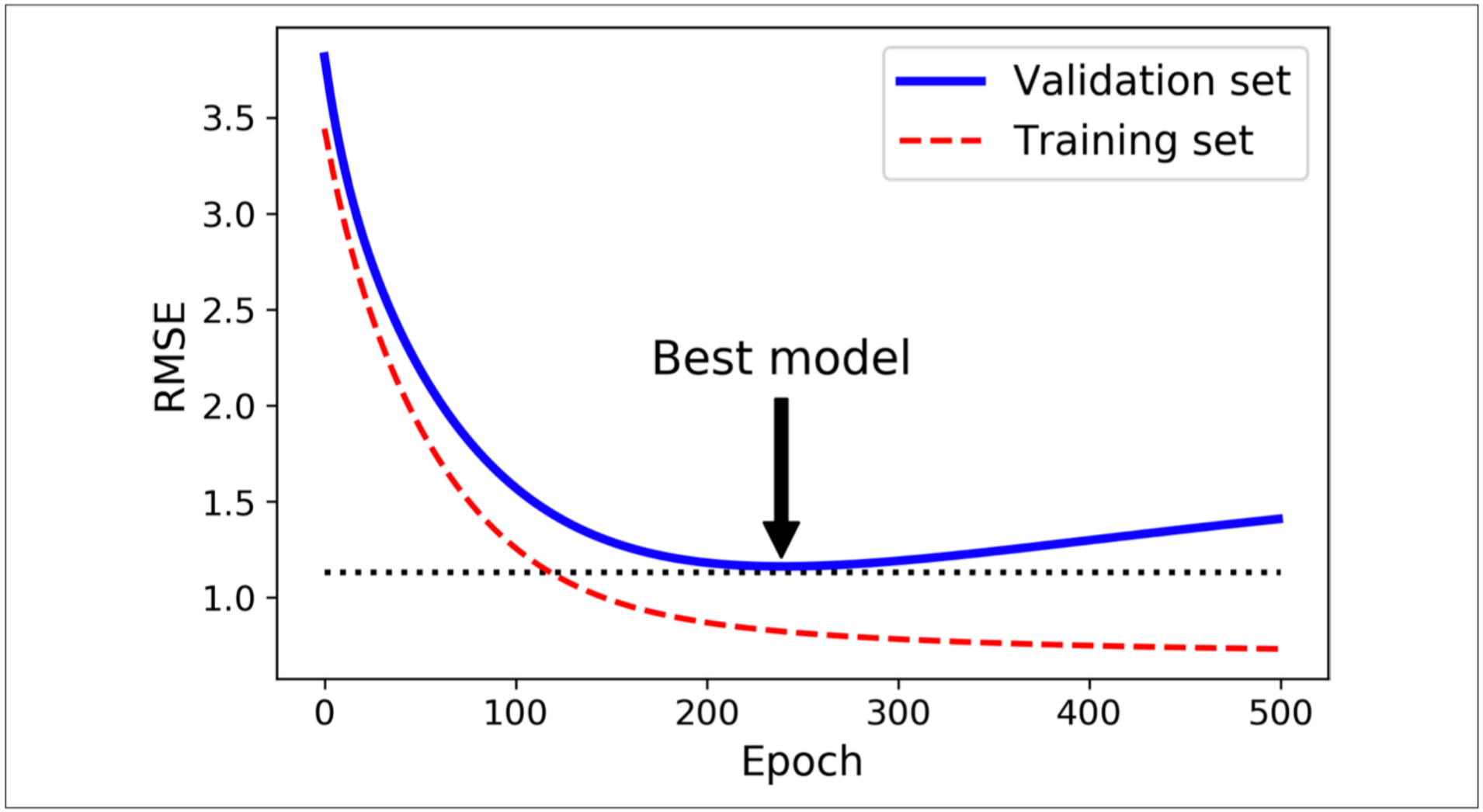
Figure 4-17. A linear model (left) and a polynomial model (right), both with various levels of Ridge regularization

- Least Absolute Shrinkage and Selection Operator (Lasso) Regression: $J(\boldsymbol{\theta}) = \text{MSE}(\boldsymbol{\theta}) + \alpha\Sigma_{i=1}^{n} \left|\theta_i\right|$
    - Lasso tends to completely eliminate the weights of the least important features. A kind of feature selection.

- Elastic Net:

$$J(\boldsymbol{\theta}) = \text{MSE}(\boldsymbol{\theta}) + r\alpha \sum_{i=1}^{n} \left|\theta_i\right| + \frac{1-r}{2}\alpha\Sigma_{i=1}^{n}\theta_i^2$$

- Early stopping



Classification problems: Logistic regression in ML

- Loss function (log-likelihood function):

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

- $\hat{p} = \sigma\left(\mathbf{x}^T\boldsymbol{\theta}\right), \sigma(t) = \dfrac{1}{1+\exp(-t)}$

- Training: gradient descent.

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

- Multiclass classification: Softmax Regression
  - Softmax score for class k: $s_k(\mathbf{x}) = \mathbf{x}^T \boldsymbol{\theta}^{(k)}$
  - The estimated probability of $\boldsymbol{x}$ belonging to class k:

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

- prediction: $\hat{y} = \underset{k}{\operatorname{argmax}}\, \sigma(\mathbf{s}(\mathbf{x}))_k$
- The loss function: $J(\boldsymbol{\Theta}) = -\frac{1}{m} \Sigma_{i=1}^{m} \Sigma_{k=1}^{K} y_k^{(i)} \log\left(\hat{p}_k^{(i)}\right)$. This is called Cross Entropy Loss Function, or simply log loss

A short note on the log loss function

- Shannon (1948). The beginning of Information Theory.
- Bit = 0 or 1.
- Bit = Uncertainty divided by 2.
- Entropy: $\mathrm{H(p)} = -\Sigma_k p_k \log_2 \left(p_k\right)$
- Cross Entropy: $\mathrm{H(p, q)} = -\Sigma_\mathrm{k} p_k \log_2 \left(q_k\right)$
- Cross Entropy = Entropy + KL Divergence

A recent working paper of mine: Direction is more Important than Speed.
https://ssrn.com/abstract=5176925

In a typical setting, a predictive model of asset returns can be written as $r_{i,t+1} = \mathrm{E}_t(r_{i,t+1}) + \varepsilon_{i,t+1}$, where $\mathrm{E}_t(r_{i,t+1}) = f(x_{i,t})$ and $f(\cdot)$ is some functional form that encapsulates some set of predictors $x_{i,t}$, generally constructed through economic theory. In this model, the realized excess return $r_{i,t+1}$ is a continuous variable and the determination of the sign of $r_{i,t+1}$ is not considered a separate issue. However, the performance of such predictive models is often poor, particularly out-of-sample, as famously argued by Welch and Goyal (2008) and Goyal, Welch, and Zafirov (2024), sparking widespread debate about stock return predictability. A common approach to improve model performance, proposed by Campbell and Thompson (2008), is to set predictions of a negative equity premium to zero, since the expected equity premium, as compensation for risk, should theoretically be positive. This practice, however, effectively eliminates the possibility of sign prediction.[2] Yet, the *expectation* that the equity premium should be positive is not entirely synonymous with *always* predicting the *next period*'s equity premium to be positive, especially when "expected" is understood statistically as the average (and its corresponding "expectation" in the mathematical sense). From a practical perspective, the monthly realized excess return of the S&P 500 is negative approximately 40% of the time, leading us to view sign prediction as a meaningful and important topic that does not contradict financial theory.

An example of the interplay of direction and value prediction is the dividend-price channel from the Campbell-Shiller identity (Campbell and Shiller, 1988a), as articulated by Cochrane (2008): a rising $\log(Dividend_t/Price_t)$ (logDP, or $d_t - p_t$) signals higher returns at $t+1$, while a falling logDP indicates lower returns, i.e.

$$r_{t+1} = a + b(d_t - p_t) + \epsilon_{t+1}, \quad b \approx 0.10 \text{ with annual data,} \quad \text{(Equation (1) of Cochrane (2008)).}$$

Cochrane (2008) shows that the primary variation in the log dividend-price ratio ($d_t - p_t$) stems from changes in price (P) rather than dividends (D). Noting that price fluctuations are directly linked to the positivity or negativity of returns, it is evident that a less ambitious inference can be drawn: logDP has a positive relationship with the sign of returns, such that an increase (decrease) in logDP is more likely to result in a positive (negative) return:

$$\Pr_t(r_{t+1} > 0) \propto \text{logDP}_t. \tag{1}$$

This example illustrates that sign prediction can be seen as a simplified version of value prediction grounded in economic theory. Similarly, other valuation-ratio-related indicators, such as the Dividend Yield (logDY), Earnings Price Ratio (logEP), Dividend Payout Ratio (logDE), and Book-to-Market Ratio (BM), among others, can be discussed within the same framework.[3]

Another intuitively compelling example, viewed from a statistical perspective, is provided by Christoffersen and Diebold (2006):

$$\begin{aligned}
\Pr_t\left(r_{t+1} > 0\right) &= 1 - \Pr_t\left(r_{t+1} \leq 0\right) \\
&= 1 - \Pr\left(\frac{r_{t+1} - \mu}{\sigma_{t+1|t}} \leq \frac{-\mu}{\sigma_{t+1|t}}\right) \\
&= \Phi\left(\frac{\mu}{\sigma_{t+1|t}}\right), \\
r_{t+1} \mid \Omega_t &\sim N\left(\mu, \sigma^2_{t+1|t}\right).
\end{aligned} \tag{2}$$

Figure 1: Signs and Sign Dependence of S&P 500 Excess Returns during Typical Periods

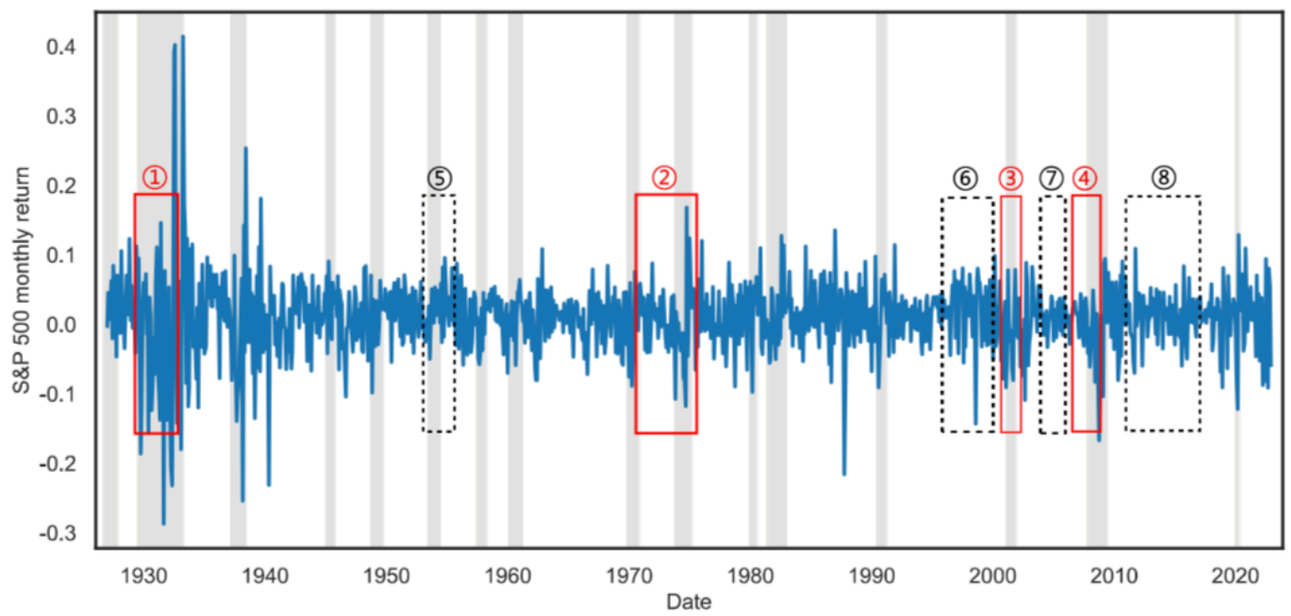


Table 4: Out-of-sample Forecasting Results of Direction and Value Prediction

Models	Accuracy (%)	Pos-Acc (%)	Neg-Acc (%)	F-score	R^2_{OOS} (%)
Panel A: Direction Prediction					
Logistic	**54.91**	69.17	34.07	45.65	-
PCA-Logistic	**58.33**	79.70	27.11	40.46	-
Lasso	58.78	82.21	24.54	**37.80**	-
Enet	58.93	82.21	24.91	**38.23**	-
GBDT	**61.01**	90.48	17.95	**29.96**	-
RF	**61.31**	86.22	24.91	**38.65**	-
NN	55.80	75.44	27.11	**39.89**	-
Avg(uni)	**59.38**	100.00	0.00	0.00	-
Avg(all)	**59.23**	83.21	24.18	**37.47**	-
Avg(linear)	**59.08**	79.45	29.30	**42.81**	-
Avg(nonlinear)	**59.67**	86.97	19.78	**32.23**	-
Avg(nonlin/NN)	**61.61**	88.47	22.34	**35.67**	-
Panel B: Value Prediction					
LR	52.08	62.41	37.00	46.46	-12.81
PCA-LR	55.95	70.68	34.43	46.30	-2.56
Lasso	59.38	99.50	0.73	1.45	$0.46^{*}(1.60)$
Enet	59.23	99.75	0.00	0.00	$0.41^{*}(1.60)$
GBRT	57.29	85.46	16.12	27.12	-0.45
RF	58.78	82.71	23.81	36.98	$0.06^{**}(1.90)$
NN	59.38	100.00	0.00	0.00	$0.16(1.22)$
Avg(uni)	59.23	99.75	0.00	0.00	$0.21(1.17)$
Avg(all)	58.48	89.22	13.55	23.53	$0.55^{**}(1.67)$
Avg(linear)	55.36	81.45	17.22	28.43	-0.09
Avg(nonlinear)	58.63	91.98	9.89	17.86	$0.53^{**}(1.65)$
Avg(nonlin/NN)	58.48	84.96	19.78	32.09	-0.01
Panel C: Benchmarks					
Historical Mean (HM)	59.38	100.00	0.00	0.00	0.00
Naive All Positive (NAP)	59.38	100.00	0.00	0.00	-

Table 7: Pairwise Comparison of Utility Gains between Direction and Value Strategies

	Direction Strategy 1				Direction Strategy 2			
	$\Delta_{(\text{value})}(\%)$		$\Delta_{(\text{value})}$ growth(%)		$\Delta_{(\text{value})}(\%)$		$\Delta_{(\text{value})}$ growth(%)	
	No trans.	50bps trans.	No trans.	50bps trans.	No trans.	50bps trans.	No trans.	50bps trans.
Logistic	1.87	1.22	32.60	23.74	2.30	0.76	39.96	14.74
PCA-Logistic	1.17	1.09	16.18	16.85	1.60	0.87	22.19	13.45
Lasso	1.38	1.38	20.42	22.53	1.60	1.10	23.74	18.02
Enet	2.31	2.26	33.92	36.19	2.63	2.00	38.61	32.09
GBDT	0.35	1.80	4.38	31.73	0.27	1.29	3.44	22.68
RF	0.14	1.45	1.57	22.57	0.34	0.92	3.90	14.28
NN	0.93	0.52	16.10	9.96	0.75	−0.14	12.98	−2.75
Avg(uni)	0.71	2.09	10.35	38.07	0.71	2.09	10.35	38.07
Avg(all)	−0.43	0.65	−4.86	9.23	−0.70	−0.08	−7.80	−1.16
Avg(linear)	0.53	1.98	6.07	30.64	0.75	1.64	8.64	25.30
Avg(nonlinear)	0.28	1.36	3.60	23.20	0.34	0.93	4.41	15.76
Avg(nonlin/NN)	1.14	2.46	14.82	44.20	1.35	2.26	17.59	40.57

Note: The utility gain $\Delta_{(\text{value})}$ (%) is the difference of the annualized certainty equivalent return gain between direction-based strategy and value-based strategy for the same model. The $\Delta_{(\text{value})}$ growth (%) is the percentage change of $\Delta_{(\text{value})}$.

Table 8: Pairwise Comparison of Utility Gains from the Direction-constrained Value Strategy

Constrained models	Benchmark: Value Strategy				Benchmark: Direction Strategy 2			
	$\Delta_{(\text{value})}(\%)$		$\Delta_{(\text{value})}$ growth(%)		$\Delta_{(\text{direction2})}(\%)$		$\Delta_{(\text{direction2})}$ growth(%)	
	No trans.	50bps trans.	No trans.	50bps trans.	No trans.	50bps trans.	No trans.	50bps trans.
LR	0.19	0.74	2.36	14.23	0.24	0.09	3.03	1.49
PCA-LR	0.96	0.90	13.97	18.45	−0.96	−1.55	−10.93	−21.13
Lasso	0.85	0.34	12.55	5.50	−0.76	−0.76	−9.04	−10.61
Enet	2.09	1.59	30.76	25.52	−0.53	−0.41	−5.66	−4.97
GBRT	−0.11	0.07	−1.43	1.15	−0.39	−1.22	−4.70	−17.55
RF	1.08	1.04	12.61	16.07	0.75	0.11	8.38	1.56
NN	−0.21	−1.24	−3.57	−24.05	−0.96	−1.10	−14.65	−21.90
Avg(uni)	0.05	0.05	0.70	0.85	−0.33	−1.06	−4.35	−13.94
Avg(all)	−0.42	−0.46	−4.72	−6.51	0.27	−0.38	3.34	−5.41
Avg(linear)	1.07	1.32	12.21	20.36	0.31	−0.32	3.29	−3.94
Avg(nonlinear)	0.96	0.90	12.48	15.25	0.62	−0.03	7.73	−0.44
Avg(nonlin/NN)	1.12	1.19	14.65	21.38	−0.23	−1.07	−2.50	−13.65

Table 10: Out-of-sample Forecasting Results of Direction and Value Prediction with a Subset of Factors

Models with Limited Variables	Accuracy (%)	Pos-Acc (%)	Neg-Acc (%)	F-score	R^2_{OOS} (%)
Panel A: Direction Prediction					
Logistic	**55.80**	81.45	18.32	14.92	-
PCA-Logistic	**58.63**	97.24	2.20	2.14	-
Lasso	56.55	85.96	13.55	**11.65**	-
Enet	55.36	83.21	14.65	**12.19**	-
GBDT	**60.12**	93.48	11.36	**10.62**	-
RF	**62.65**	89.97	22.71	**20.43**	-
NN	54.17	75.19	23.44	**17.63**	-
Avg(uni)	59.38	100.00	0.00	0.00	-
Avg(all)	57.14	88.97	10.62	**9.45**	-
Avg(linear)	56.99	87.47	12.45	**10.89**	-
Avg(nonlinear)	**58.04**	87.72	14.65	**12.85**	-
Avg(nonlin/NN)	**61.31**	92.23	16.12	**14.87**	-
Panel B: Value Prediction					
LR	52.23	68.92	27.84	19.19	−9.84
PCA-LR	55.21	77.19	23.08	17.81	−0.49
Lasso	59.38	99.25	1.10	1.09	−0.97
Enet	59.38	99.25	1.10	1.09	−0.84
GBRT	56.25	88.47	9.16	8.10	−0.60
RF	56.10	83.46	16.12	13.45	−0.74
NN	59.38	99.75	0.37	0.37	0.23**(1.65)
Avg(uni)	59.38	99.75	0.37	0.37	−0.31
Avg(all)	57.29	92.23	6.23	5.74	−0.39
Avg(linear)	57.89	89.97	10.99	9.89	−1.23
Avg(nonlinear)	56.40	93.48	2.20	2.05	0.08(1.22)
Avg(nonlin/NN)	55.21	85.21	11.36	9.68	−0.56
Panel C: Benchmarks					
Historical Mean (HM)	59.38	100.00	0.00	0.00	0.00
Naive All Positive (NAP)	59.38	100.00	0.00	0.00	-

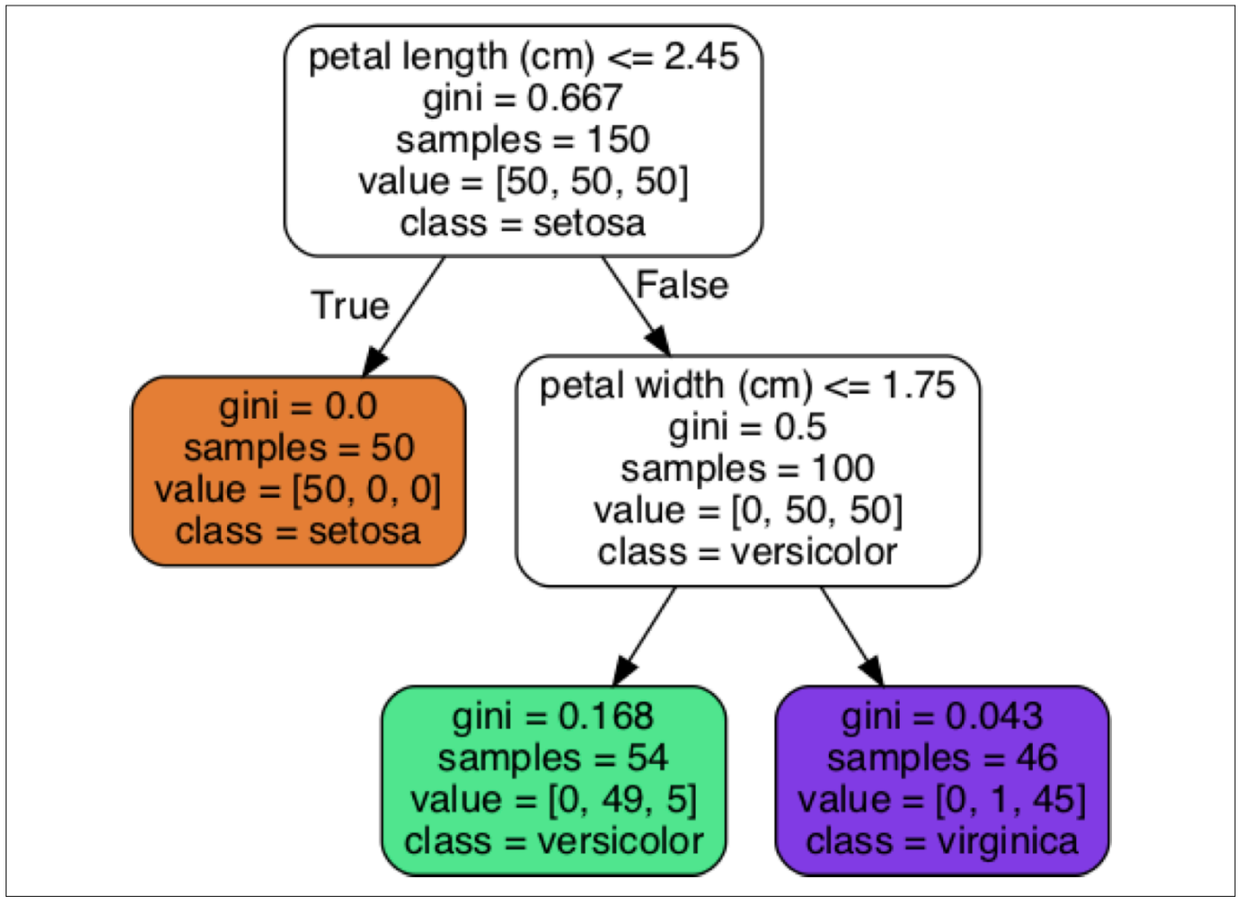
Note: Model name abbreviations are the same as those in Table 4. Clark and West (2007) MSFE-adjusted statistics are included in the parentheses. *, ** and *** stand for significance at the 10%, 5%, and 1% levels, respectively. Bold numbers in Panel A indicate values that surpass those of the same model under value prediction in Panel B. Numbers with underlining indicate that they are smaller than the values at the same positions in Table 4.

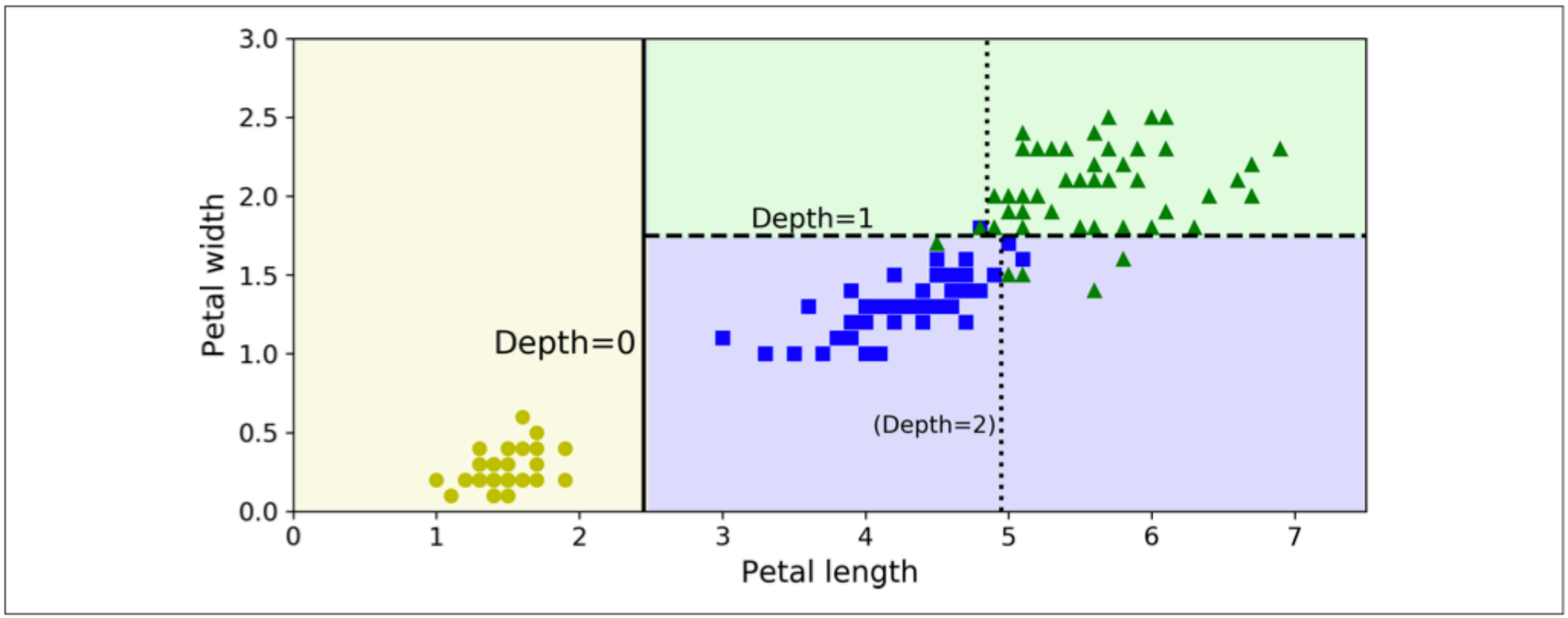
Table 11: Pairwise Comparison of Direction and Value Prediction Models with a Subset of Factors: Utility Gains

Models with Limited Variables	Direction Strategy 1				Direction Strategy 2			
	$\Delta_{(\text{value})}(\%)$		$\Delta_{(\text{value})}$ growth(%)		$\Delta_{(\text{value})}(\%)$		$\Delta_{(\text{value})}$ growth(%)	
	No trans.	50bps trans.	No trans.	50bps trans.	No trans.	50bps trans.	No trans.	50bps trans.
Panel A: All Models with Limited Factors								
Logistic	1.74	3.77	29.79	125.16	1.49	3.73	25.49	123.77
PCA-Logistic	0.14	1.90	1.89	36.75	0.60	2.45	8.33	47.41
Lasso	1.48	1.22	24.42	21.13	1.74	1.63	28.78	28.14
Enet	1.71	1.40	29.54	25.83	2.03	1.97	35.17	36.26
GBDT	1.00	2.61	14.60	54.84	0.74	2.55	10.75	53.59
RF	1.23	2.74	15.78	47.49	−0.17	1.09	−2.17	18.84
NN	0.15	−0.59	2.23	−10.32	0.33	−0.60	5.06	−10.54
Avg(uni)	1.95	2.64	34.74	53.51	1.95	2.64	34.74	53.51
Avg(all)	0.91	1.84	13.94	36.66	0.62	1.93	9.51	38.29
Avg(linear)	1.41	2.69	22.30	60.19	1.02	2.51	16.13	56.14
Avg(nonlinear)	0.63	1.56	9.28	30.80	0.89	2.07	13.21	40.79
Avg(nonlin/NN)	1.11	2.56	15.26	48.34	0.25	1.82	3.47	34.28
Panel B: Value Prediction with Full Set of Variables; Direction Prediction with Limited Factors								
Logistic	−0.50	1.55	−6.13	29.63	−0.75	1.51	−9.24	28.83
PCA-Logistic	0.46	2.20	6.63	45.01	0.92	2.75	13.37	56.32
Lasso	0.79	0.91	11.63	14.87	1.05	1.31	15.53	21.52
Enet	0.69	0.58	10.11	9.36	1.01	1.15	14.89	18.42
GBDT	−0.08	1.70	−1.02	29.91	−0.34	1.64	−4.35	28.86
RF	0.42	2.05	4.87	31.82	−0.98	0.40	−11.38	6.21
NN	0.84	−0.07	14.39	−1.36	1.02	−0.08	17.55	−1.61
Avg(uni)	0.38	1.11	5.28	17.18	0.38	1.11	5.28	17.18
Avg(all)	−1.46	−0.15	−16.40	−2.11	−1.75	−0.07	−19.65	−0.95
Avg(linear)	−1.02	0.69	−11.73	10.59	−1.41	0.50	−16.19	7.79
Avg(nonlinear)	−0.28	0.76	−3.63	12.95	−0.01	1.27	−0.16	21.58
Avg(nonlin/NN)	0.74	2.30	9.66	41.25	−0.12	1.55	−1.56	27.87

Decision Trees (DT)

- Ask questions sequentially.
-



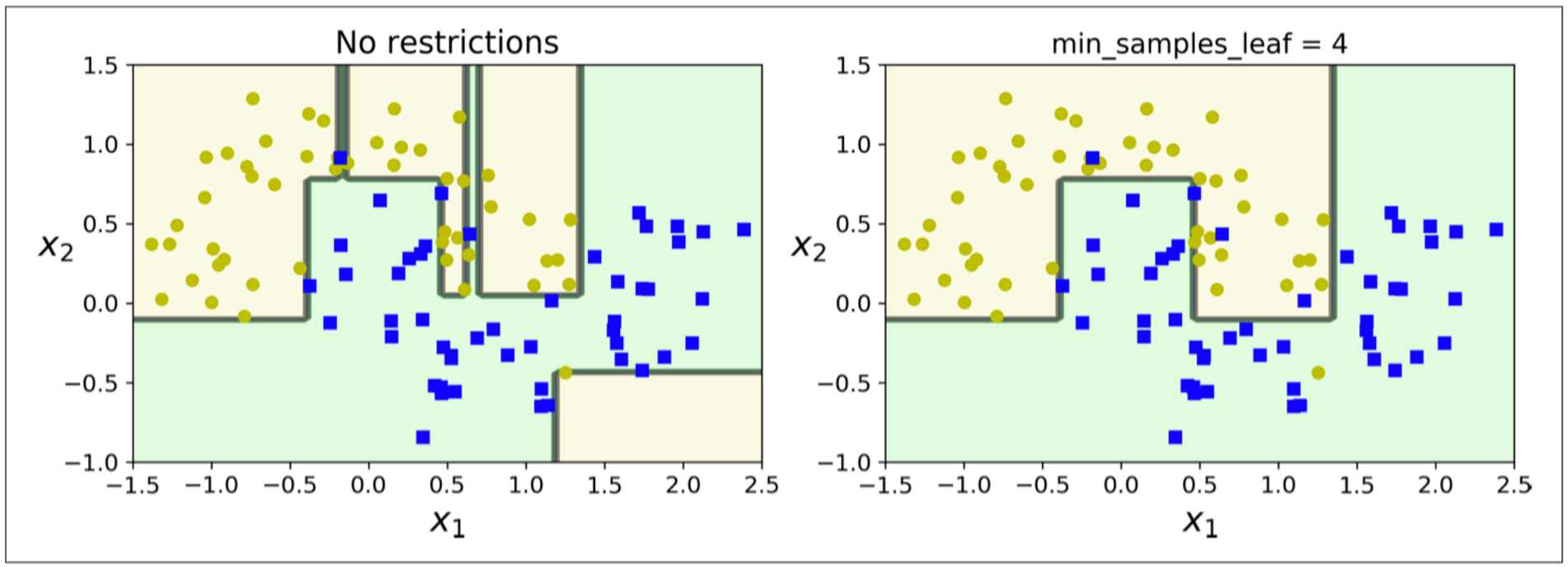

3.0
2.5
2.0
1.5
1.0
0.5
0.0
Petal width
0
1
2
3
4
5
6
7
Petal length
Depth=0
Depth=1
(Depth=2)

DT: the CART training algorithm

- First, split the samples in 2 subsets using a single feature k and a threshold t_k.
  - The algorithm searches k and t_k that produces the purest subsets (weighted by their size): $J\left(k, t_k\right) = \frac{m_{\text{left}}}{m} G_{\text{left}} + \frac{m_{\text{right}}}{m} G_{\text{right}}$
    - G: the impurity.
    - m: number of samples in the left/right subset
- Once it separates the samples into two subsets, it continues to search in the subsets.
- It stops when either it reaches the maximum depth, or if it cannot find a split that will reduce impurity.
- The optimal tree is not guaranteed.

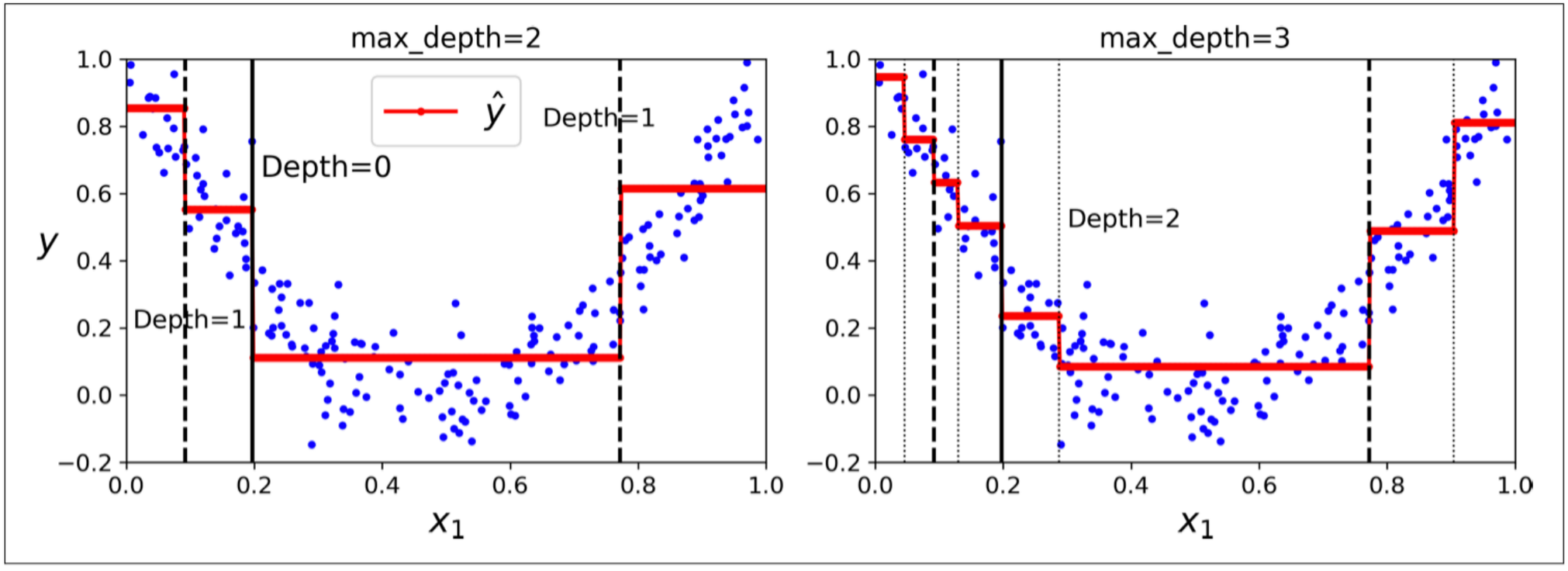
DT: Regularization

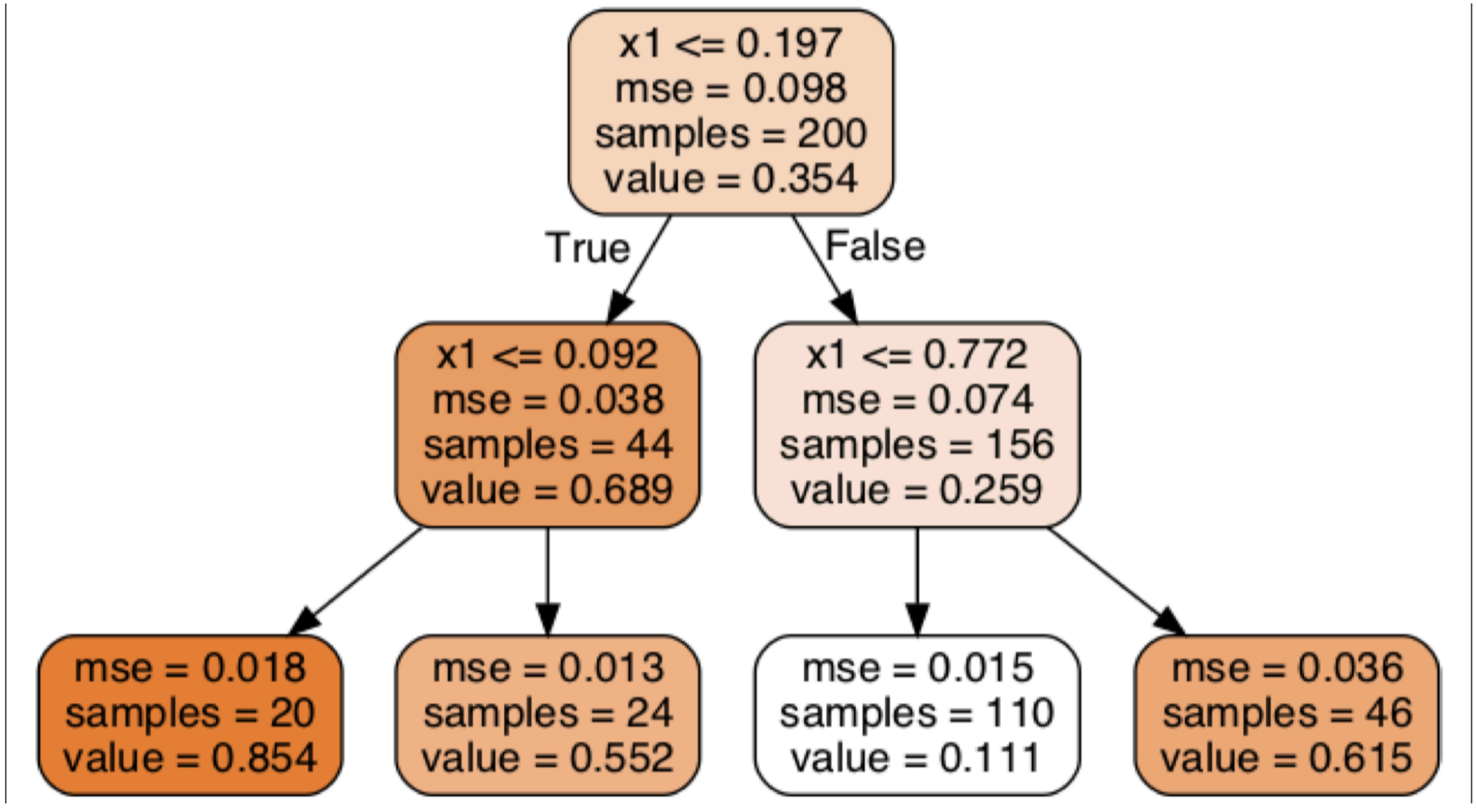
- DT is a nonparametric model.
- Nonparametric models may tend to overfit, i.e. fitting very close to the training data as little (or no) presumptions are made.
- To fight against overfitting, use regularization.
    - max_depth: the maximum level of the tree
    - min_samples_split: the minimum number of samples a node must have before it can be split
    - min_samples_leaf: the minimum number of samples a leaf node must have
    - max_leaf_nodes: maximum number of leaf nodes
    - max_features: maximum number of features that are evaluated for splitting at each node.
- pruning: delete leaf nodes if the impurity improvement it provides is not statistically significant


No restrictions
min_samples_leaf = 4
1.5
1.0
0.5
0.0
−0.5
−1.0
x_2
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
x_1

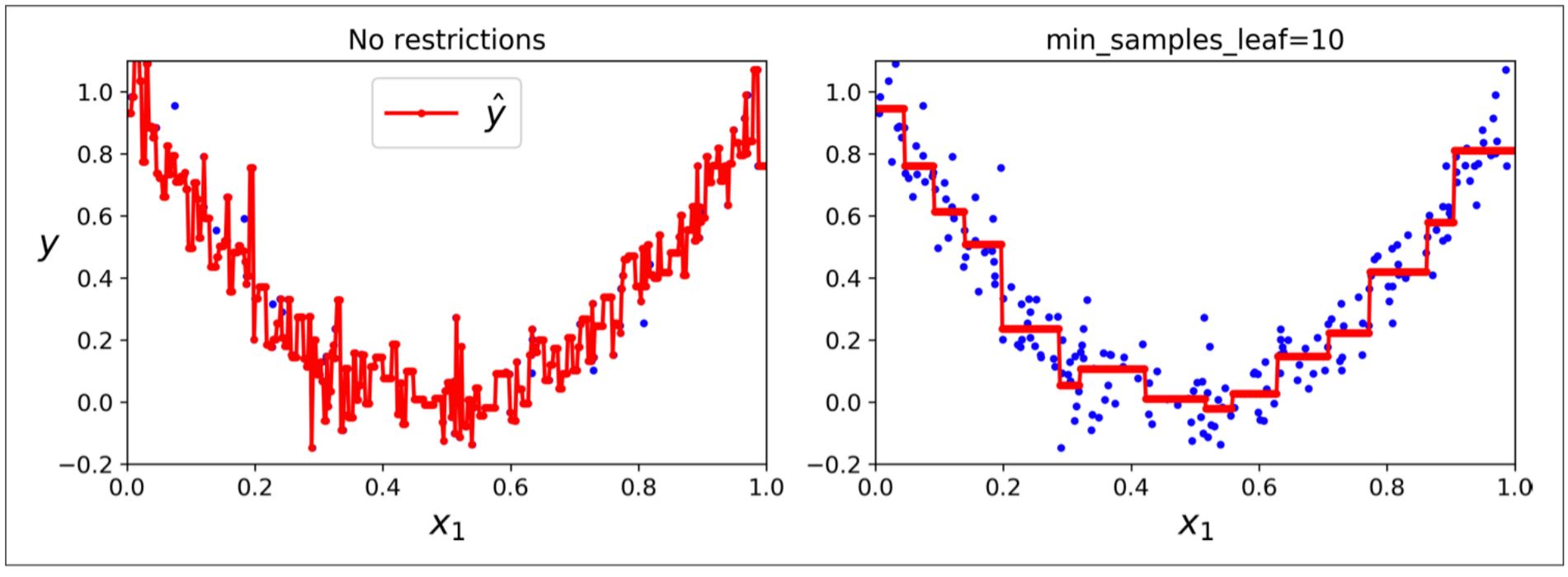
DT: regression

- $J\left(k, t_k\right) = \frac{m_{\text{left}}}{m}\text{MSE}_{\text{left}} + \frac{m_{\text{right}}}{m}\text{MSE}_{\text{right}}$



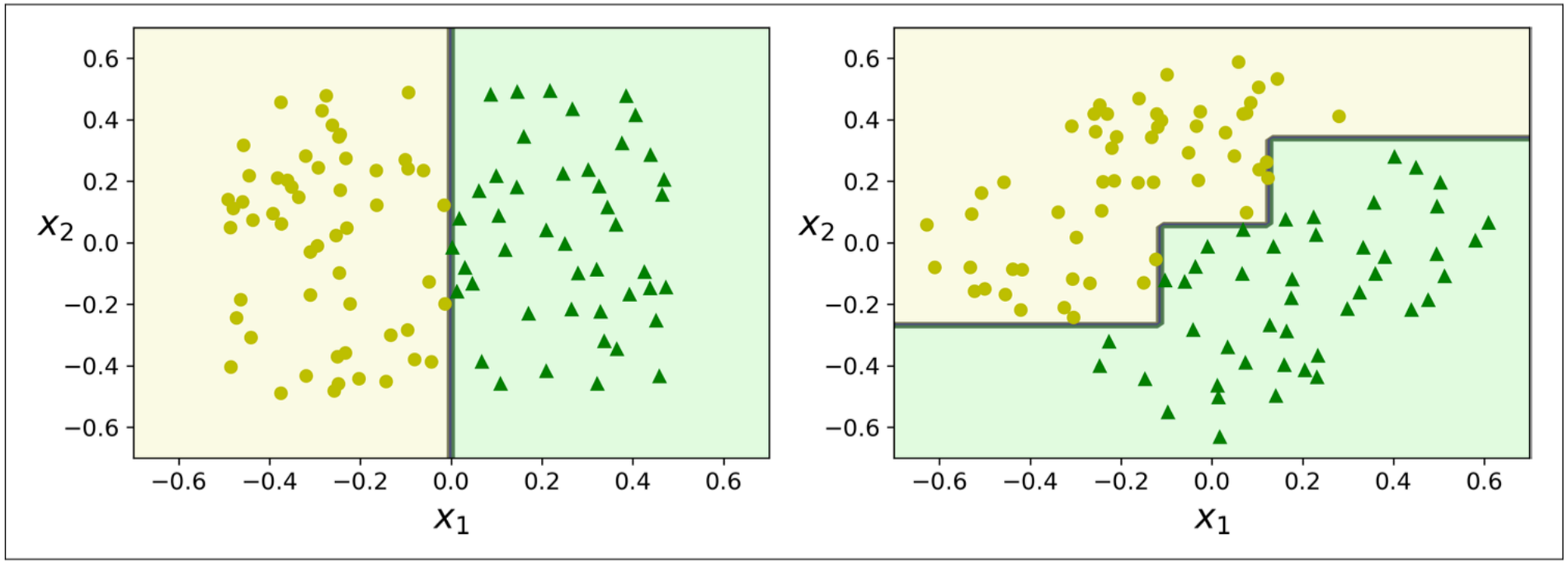

x1 <= 0.197
mse = 0.098
samples = 200
value = 0.354
True
False
x1 <= 0.092
mse = 0.038
samples = 44
value = 0.689
x1 <= 0.772
mse = 0.074
samples = 156
value = 0.259
mse = 0.018
samples = 20
value = 0.854
mse = 0.013
samples = 24
value = 0.552
mse = 0.015
samples = 110
value = 0.111
mse = 0.036
samples = 46
value = 0.615

-

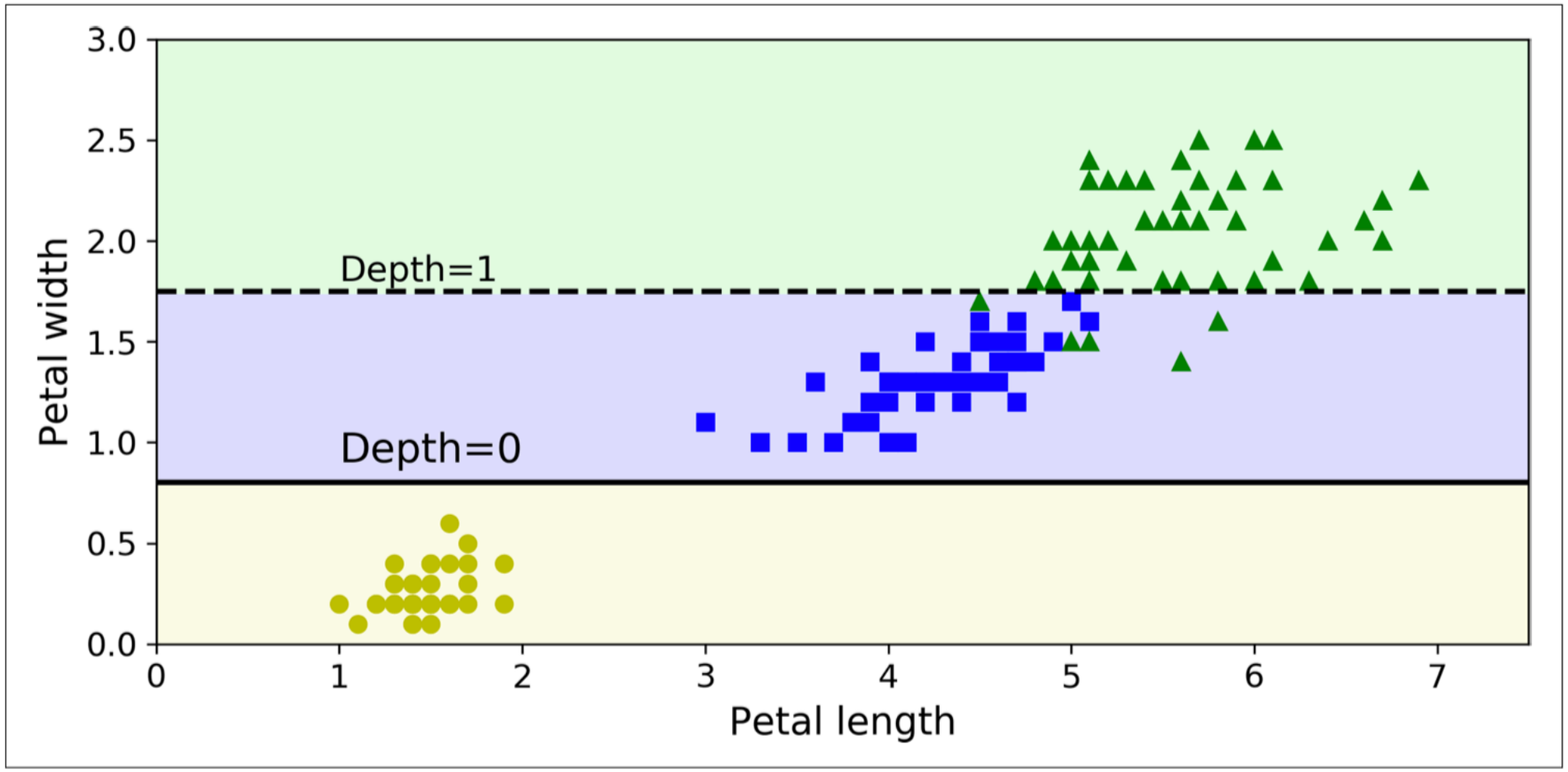
- 
No restrictions
min_samples_leaf=10
$\hat{y}$
y
x_1
x_1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0

DT: Instability

- Sensitive to small variations in data.
- rotation:



- removing just one sample from the iris data:

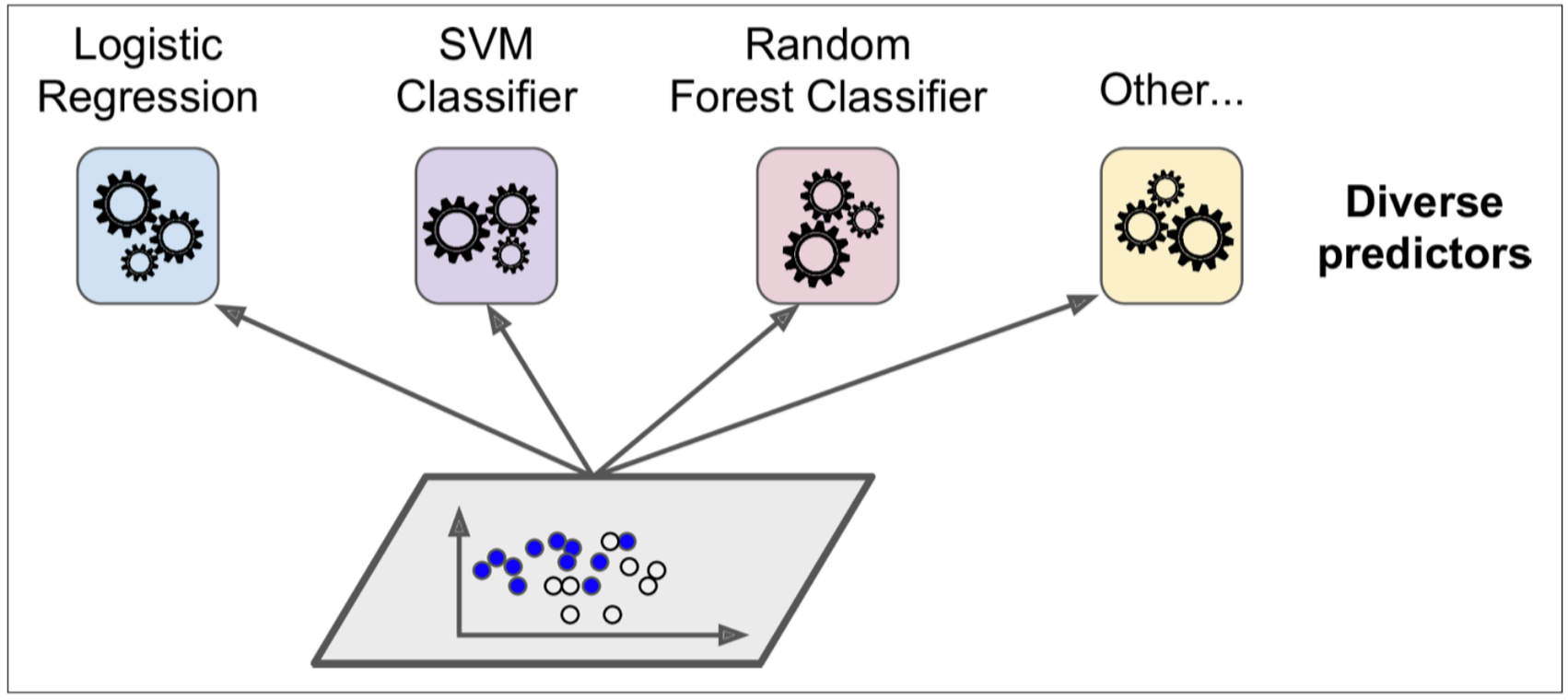


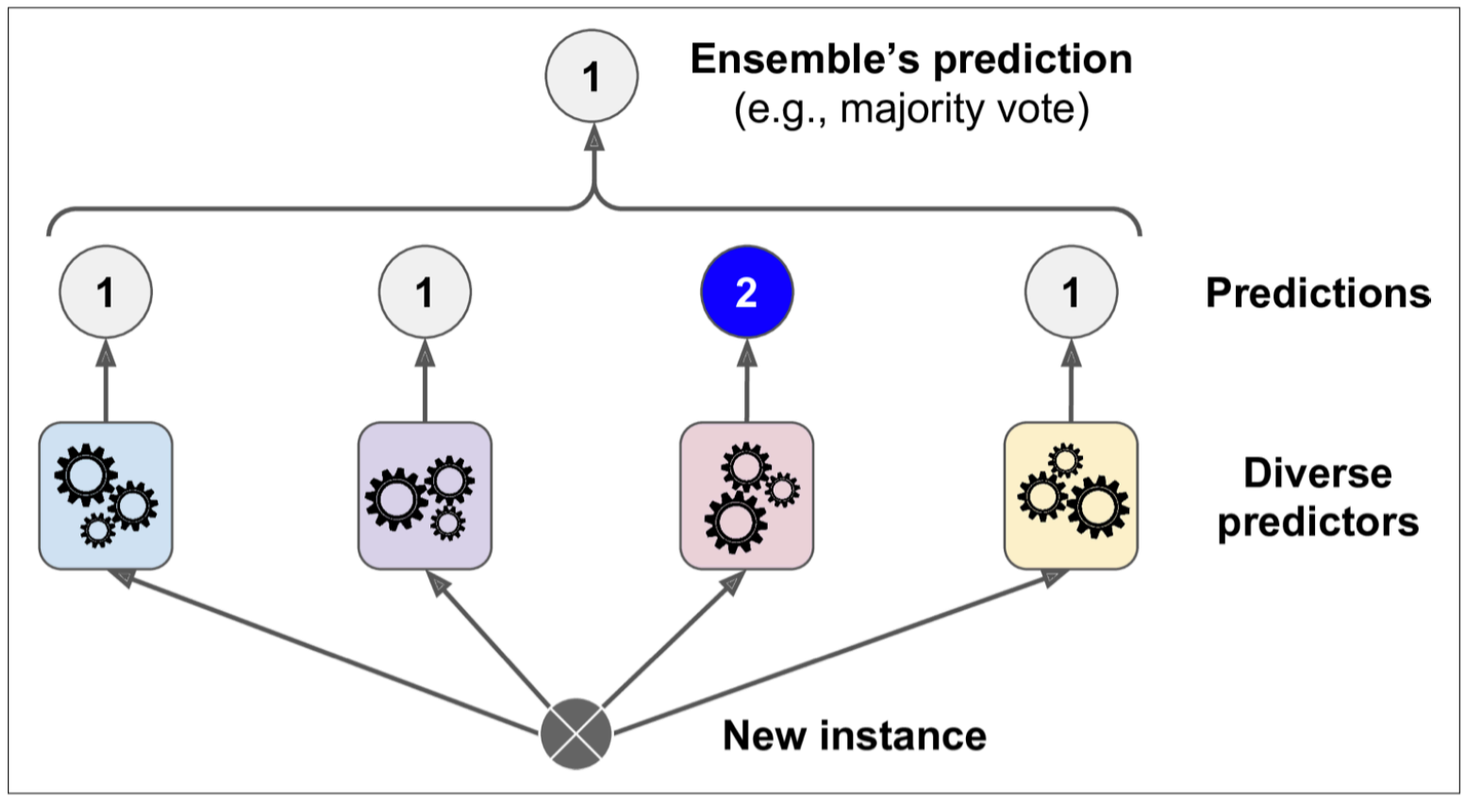
Ensemble methods

- Wisdom of the Crowd. A classical example by Francis Galton:
  - In 1908 Galton went to a country fair in Plymouth
  - The weight of an ox is being guessed by around 800 people. The person who guessed closest to the butchered weight of the ox won a prize.
  - After the contest Galton collected the tickets on which the people wrote their answers. He found that the average number is 1197 lbs, while the actual weight is 1198 lbs.
  - This answer is better than the winner’s answer, or any other cattle experts at the fair.

Ensemble: Voting

- Random Forest:
  - Train a group of Decision Trees on different subset of the training data.
  - Obtain the predictions from all individual trees, and aggregate them (perhaps by counting the votes for each class)
- One of the most powerful Machine Learning algorithms


Logistic Regression
SVM Classifier
Random Forest Classifier
Other...
Diverse predictors

- 
1
Ensemble's prediction
(e.g., majority vote)
1
1
2
1
Predictions
Diverse predictors
New instance

- Even each algorithm is weak (slightly better than random guess), ensembling can make it strong.

- Ensembling work best when the classifiers are as independent from one another as possible.

- Hard voting: average counting. Soft voting: average probability

- Soft voting is generally better, because probability is a weight of confidence.

Bagging and Pasting

- Use the same algorithm on different random subsets of the training set.
    - Sampling with replacement: Bagging, (Boostrap aggregating)
    - without replacement: Pasting
- After resampling, make predictions by aggregation function such as voting.
- This can be done in parallel, i.e. with multiple CPU cores or multiple servers.

Out-of-Bag Validation

- By default, resampling size is m for m training samples.
- This means that there are about 37% training instances that are not sampled. $(1 - \frac{1}{m})^m \approx 0.37$
- We can use these unseen samples as the validation set
- Note that these samples are not the same for different algorithms.

Random Patches and Random Subspaces

- Sampling features: Random Subspaces method

- Sampling both training instances and features: Random Patches method

Even more randomness

- Random Forest: CART but searching the best feature among a random set of features (instead of all features)
- Extra-Trees (Extremely Randomized Trees): random threshold for each feature instead of the best possible threshold.
- More randomness: bias vs. variance
- Also it means faster training.

Feature Importance

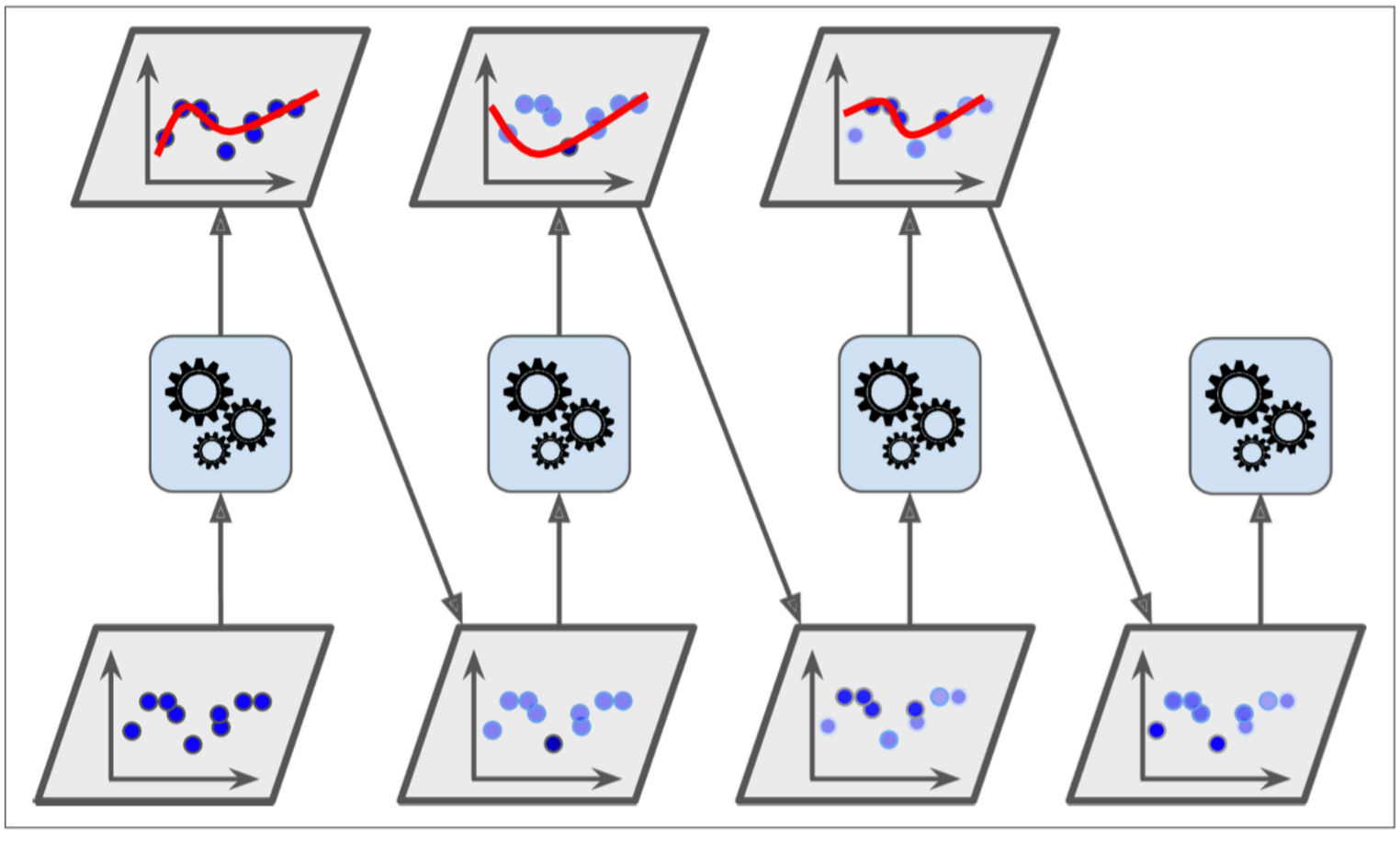
- Feature importance: how much the tree nodes that use that feature reduce impurity on average across all trees in the forest. Weighted by the number of training samples in the node.

- This is especially useful because you can quickly get an idea about what feature is valuable among all randomly chosen “experts” (trees)

Boosting

- Train predictors sequentially, each trying to improve based upon its predecessor's work

- AdaBoost: pay more attention to the training instances that the predecessor underfitted. Focus more and more on the hard cases.



Adaboost

- Each sample's weight $w^{(i)}$ is initially set to $\frac{1}{m}$
- $r_j = \frac{\sum_{i=1}^{m} w^{(i)}}{\hat{y}_j^{(i)} \neq y^{(i)}}$ where $\hat{y}_j^{(i)}$ is the j^{th} predictor's prediction for the i^{th} instance.
  - The higher r_j , the worse the predictor
- $\alpha_j = \eta \log \frac{1 - r_j}{r_j}$
  - α_j: the higher r_j, the lower α_j. The better the predictor, the higher α_j.

- Weight update rule:

$$\text{for } i = 1, 2, \cdots, m$$

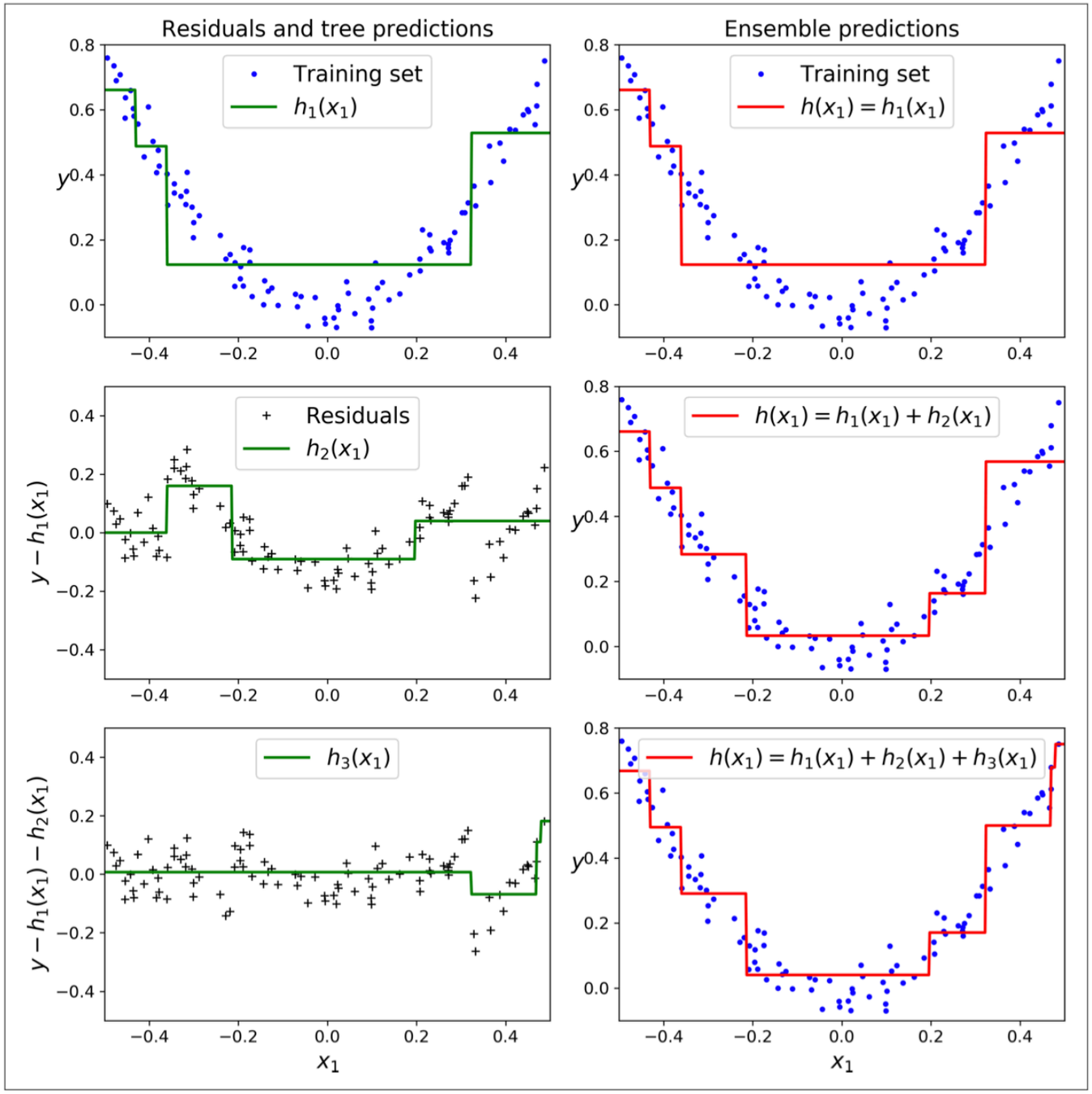
$$w^{(i)} \leftarrow \begin{cases} w^{(i)} & \text{if } \widehat{y_j}^{(i)} = y^{(i)} \\ w^{(i)} \exp\left(\alpha_j\right) & \text{if } \widehat{y_j}^{(i)} \neq y^{(i)} \end{cases}$$

- Finally,

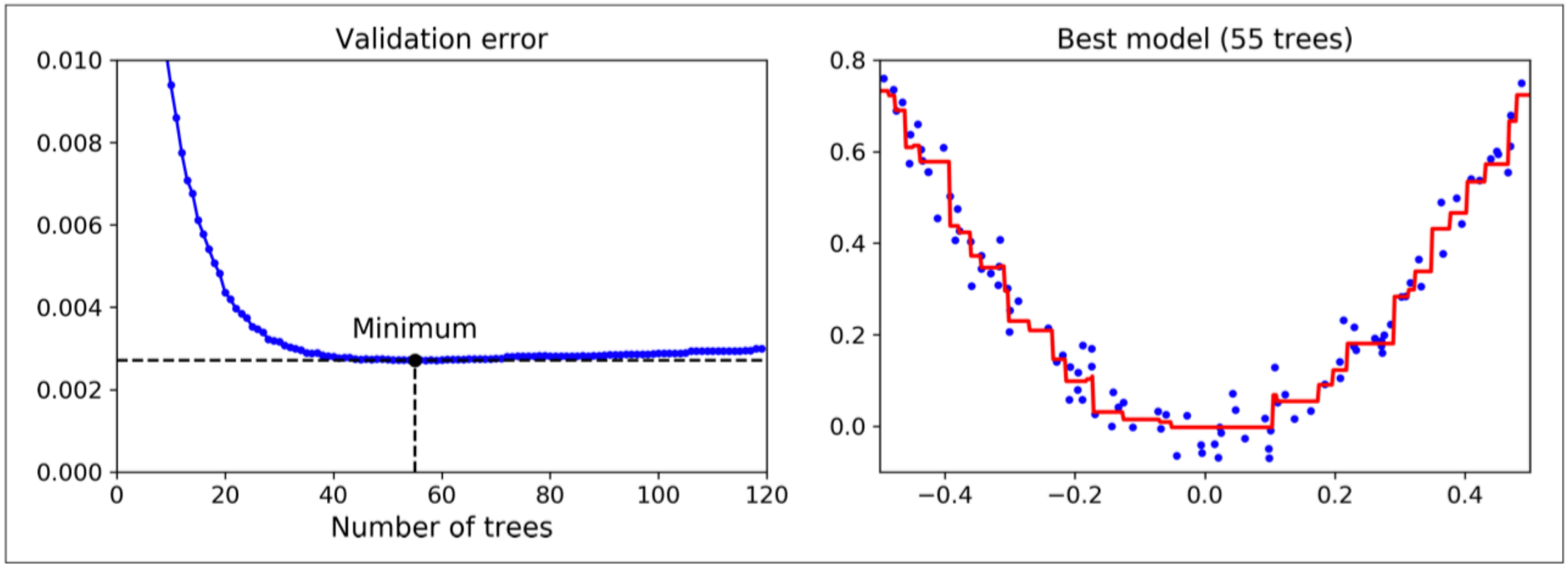
$\hat{y}(\mathbf{x}) = \underset{k}{\operatorname{argmax}} \sum_{\substack{j=1 \\ \hat{y}_j(\mathbf{x}) = k}}^{N} \alpha_j$ where N is the number of predictors.

Gradient Boosting

- Fit new predictors to the residual errors made by the previous predictor
    - tree_reg1 = DecisionTreeRegressor(max_depth=2)
      tree_reg1.fit(X, y)
      y2 = y - tree_reg1.predict(X)
    - tree_reg2 = DecisionTreeRegressor(max_depth=2)
      tree_reg2.fit(X, y2)
      y3 = y2 - tree_reg2.predict(X)
    - tree_reg3 = DecisionTreeRegressor(max_depth=2)
      tree_reg3.fit(X, y3)
    - y_pred = sum(tree.predict(X_new) for tree in (tree_reg1, tree_reg2, tree_reg3))
- Early stopping can be useful to determine how many trees are needed.


Residuals and tree predictions
Ensemble predictions
Training set
$h_1(x_1)$
$h(x_1) = h_1(x_1)$
Residuals
$h_2(x_1)$
$h(x_1) = h_1(x_1) + h_2(x_1)$
$h_3(x_1)$
$h(x_1) = h_1(x_1) + h_2(x_1) + h_3(x_1)$
y
$y - h_1(x_1)$
$y - h_1(x_1) - h_2(x_1)$
x_1

- Early stopping



- Stochastic Gradient Boosting: choose a random subset of instances for each tree.

- Check XGBoost

Stacking

- Instead of using simple rules when aggregating, such as voting, why not using a model to aggregate?

- Split the training data into two subsets.

- Train several models (the more different the better) on the first subsets. For example, train 3 models, one is linear regression, one is random forest, etc.

- Evaluate the models on the second subset. For each instance, we then have 3 predictions.

- Train a meta-model on this data (with 3 features)

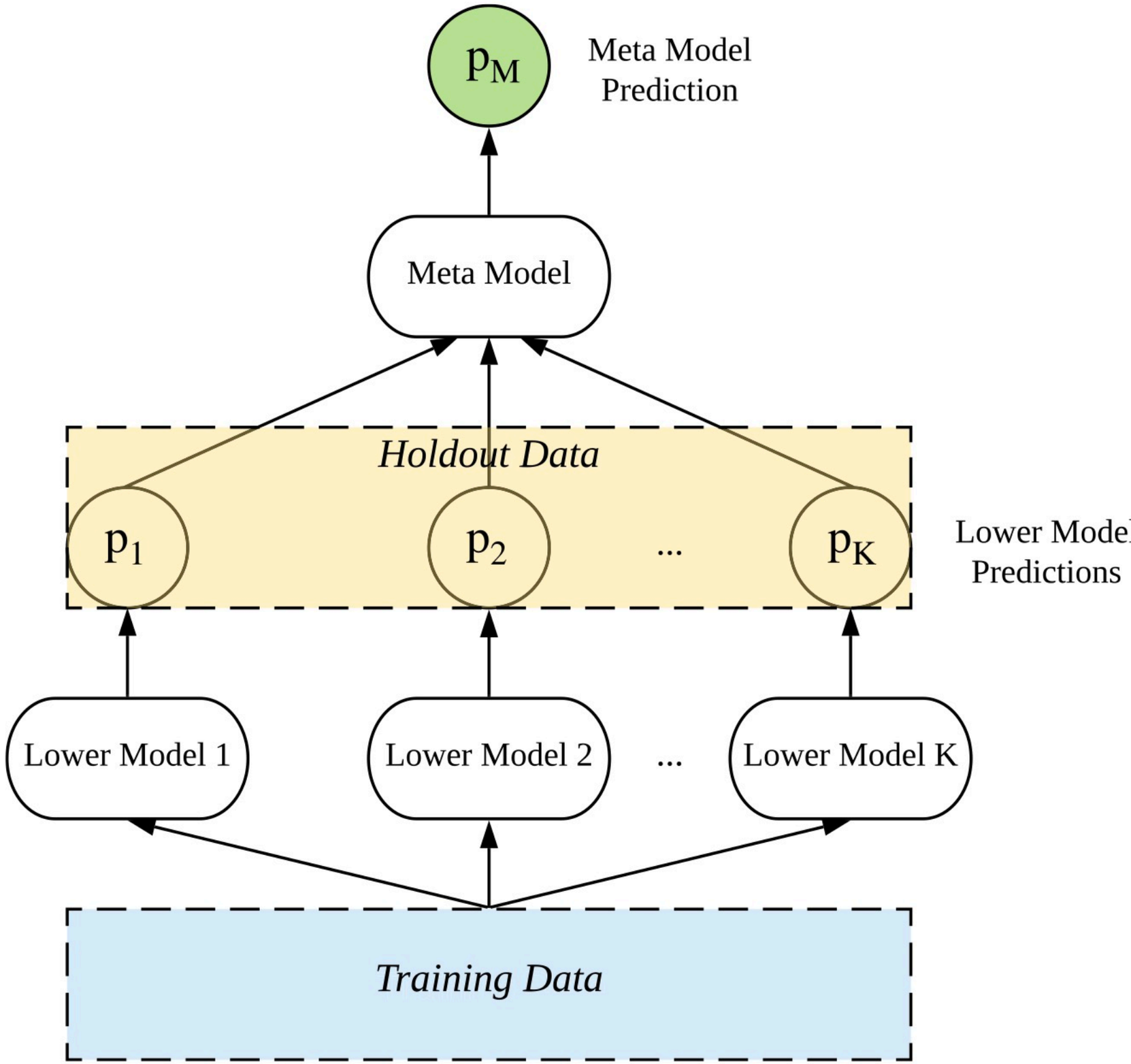
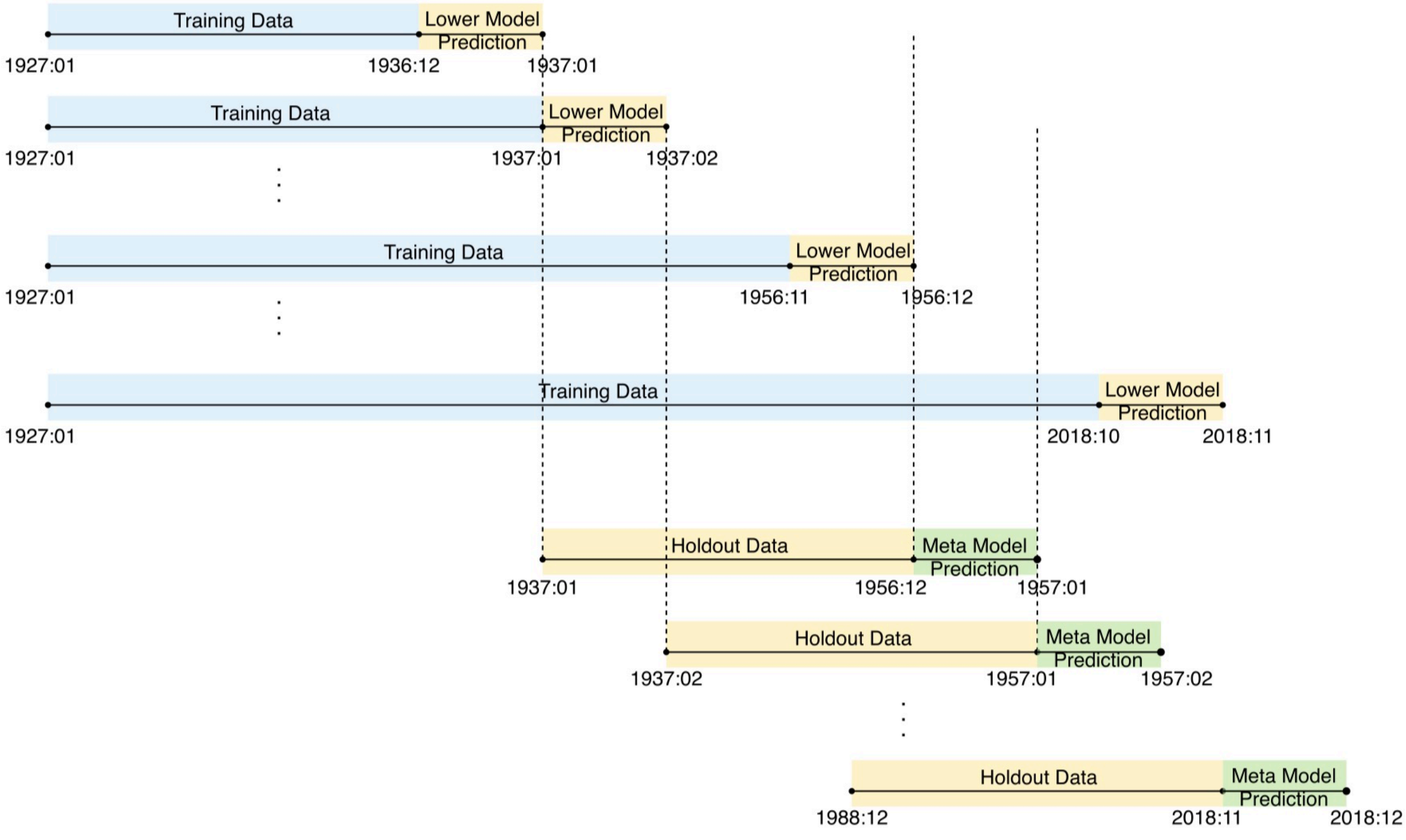


Fig. 1. Structure of stacking.

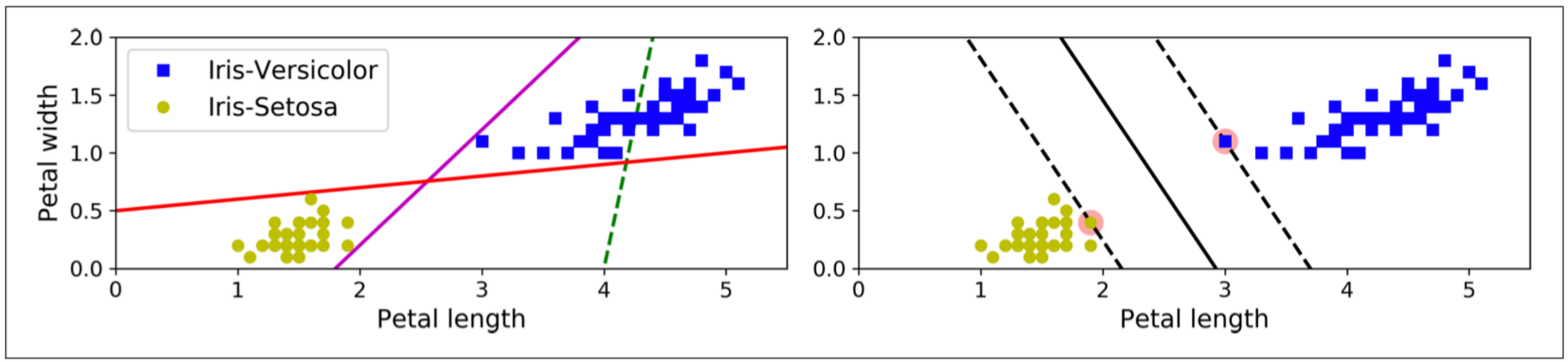
Panel A: The Training-Validation-Testing Framework



Support Vector Machines

- Particularly useful for classification of complex, but small or medium sized data

- Idea: separate classes with wide margin instead of narrow ones.

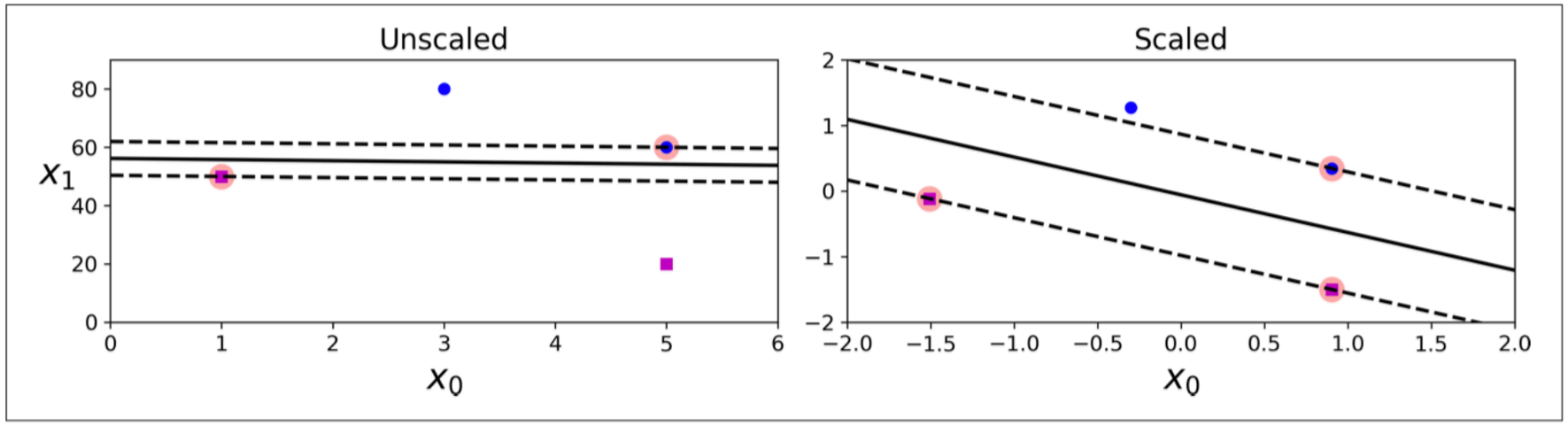


- *Figure 5-1. Large margin classification*

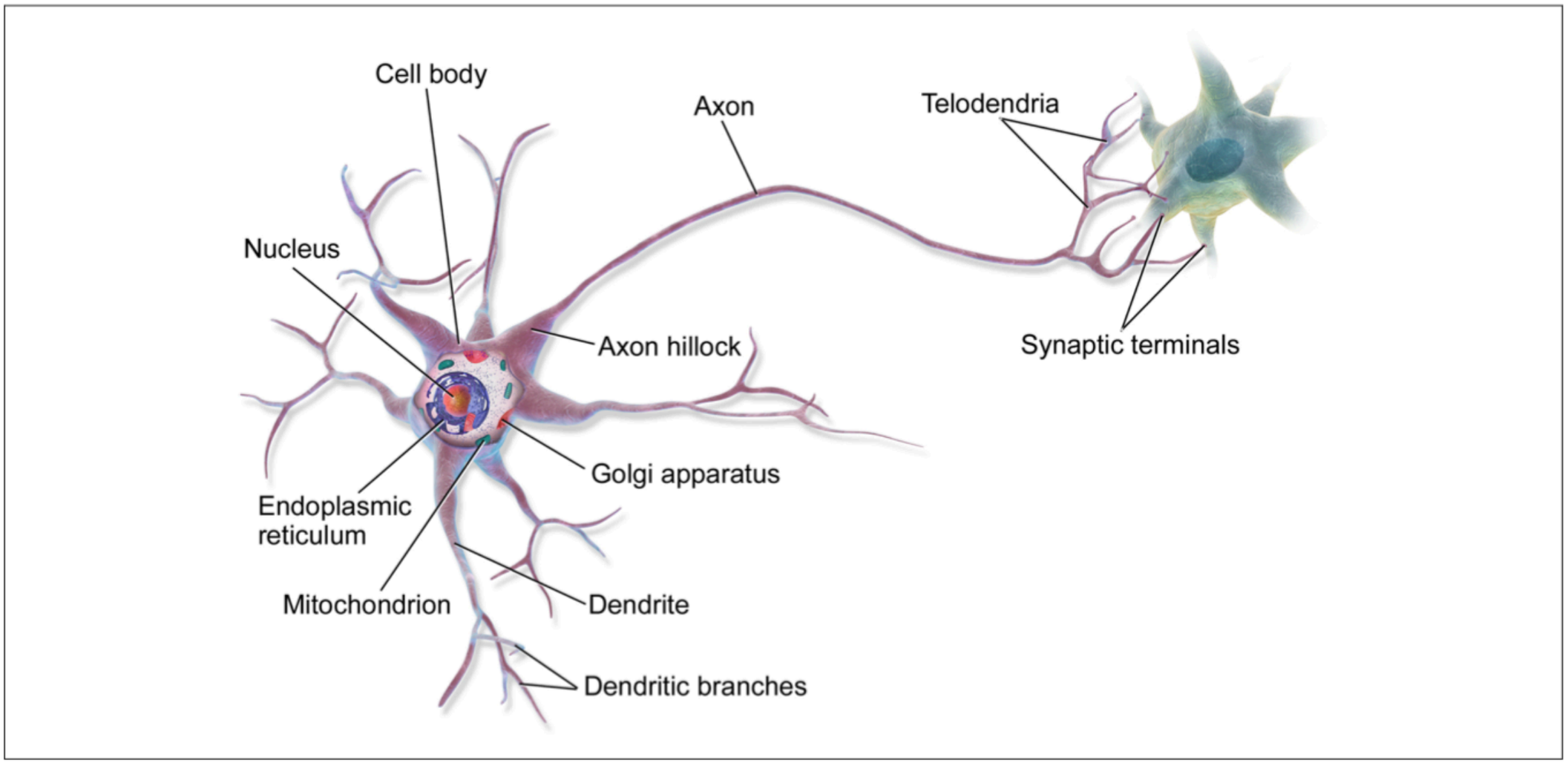
- adding new instances off the ‘street’ won’t help.

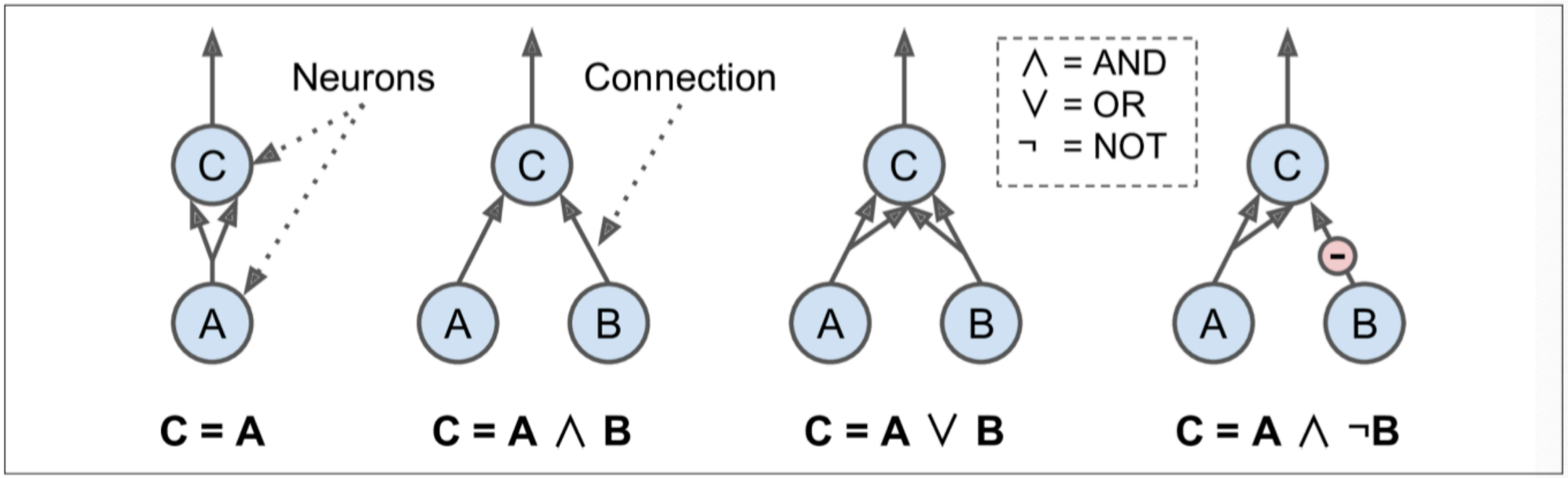
- The instances on the edge of the street are called ‘support vectors’.

- Sensitive to feature scales

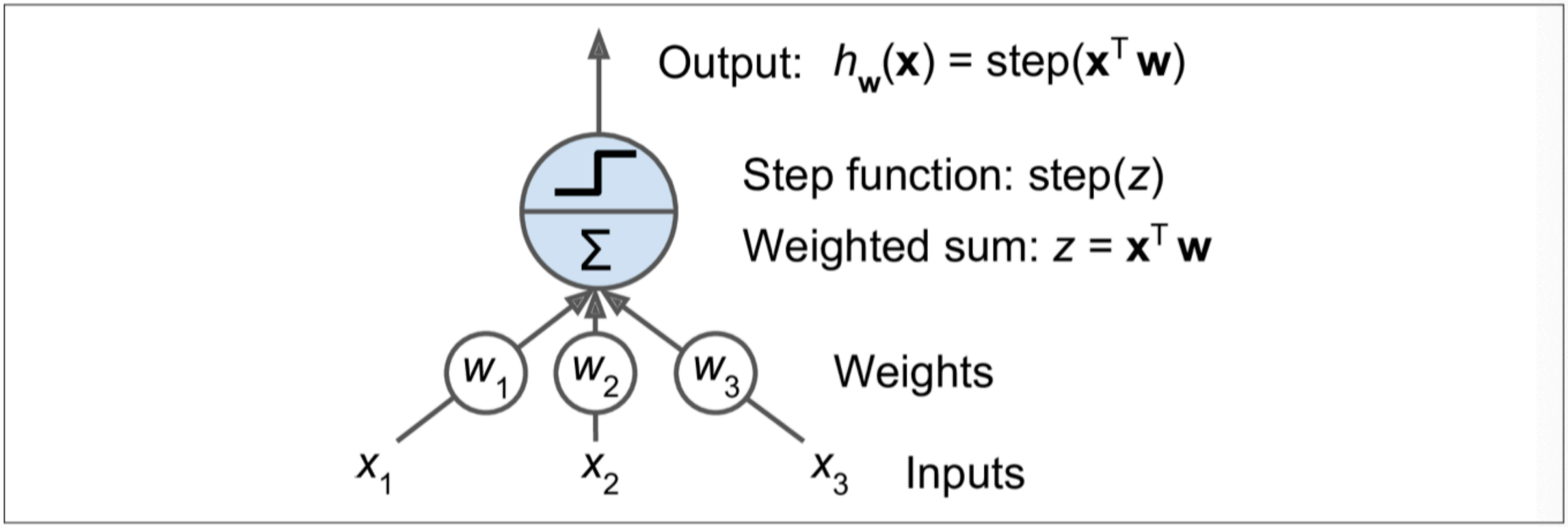


Neural Networks




Neurons
Connection
∧ = AND
∨ = OR
¬ = NOT
C
A
B
C = A
C = A ∧ B
C = A ∨ B
C = A ∧ ¬B

Perceptron

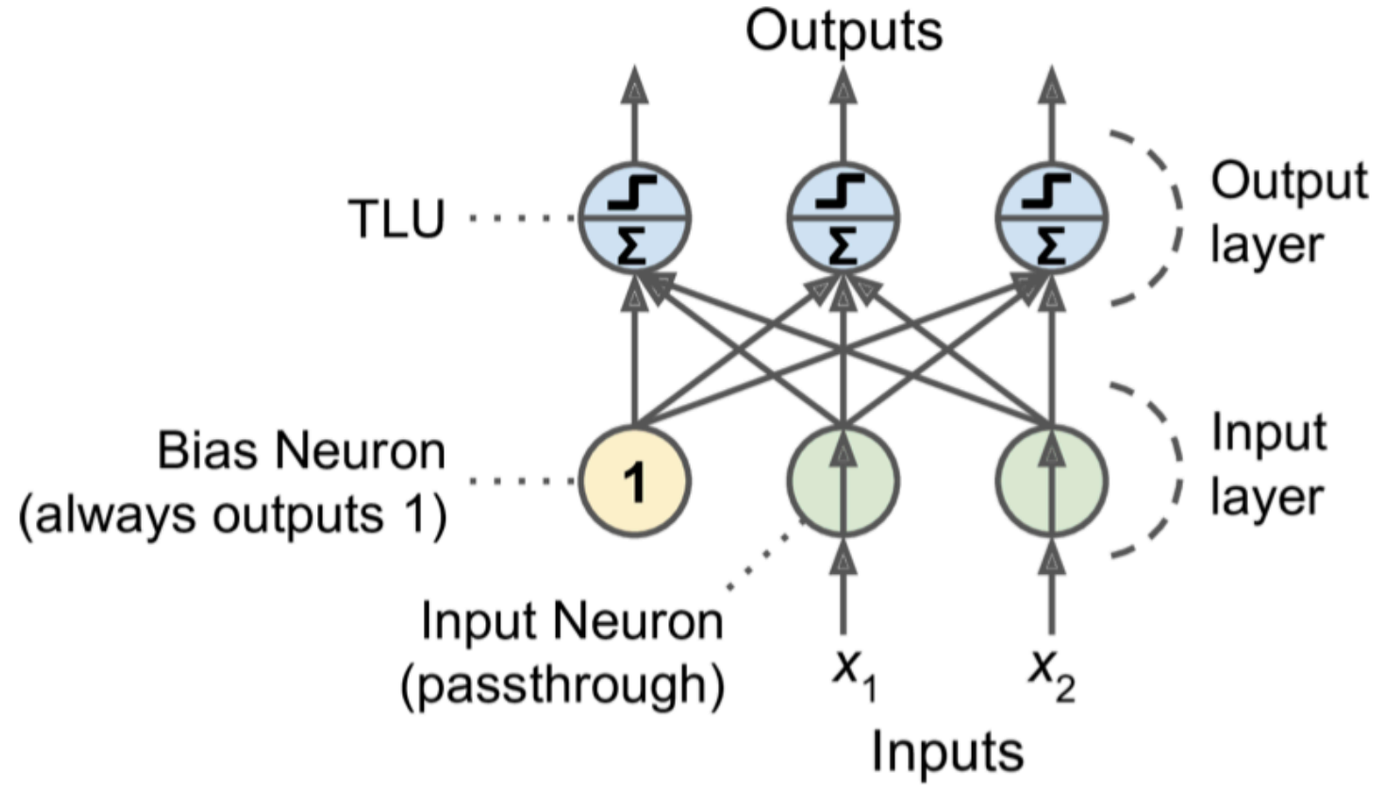

Output: $h_{\mathbf{w}}(\mathbf{x}) = \text{step}(\mathbf{x}^T \mathbf{w})$
Step function: step(z)
Weighted sum: $z = \mathbf{x}^T \mathbf{w}$
w_1
w_2
w_3
Weights
x_1
x_2
x_3
Inputs

- The step functions “activates” the neuron

- Common step functions:

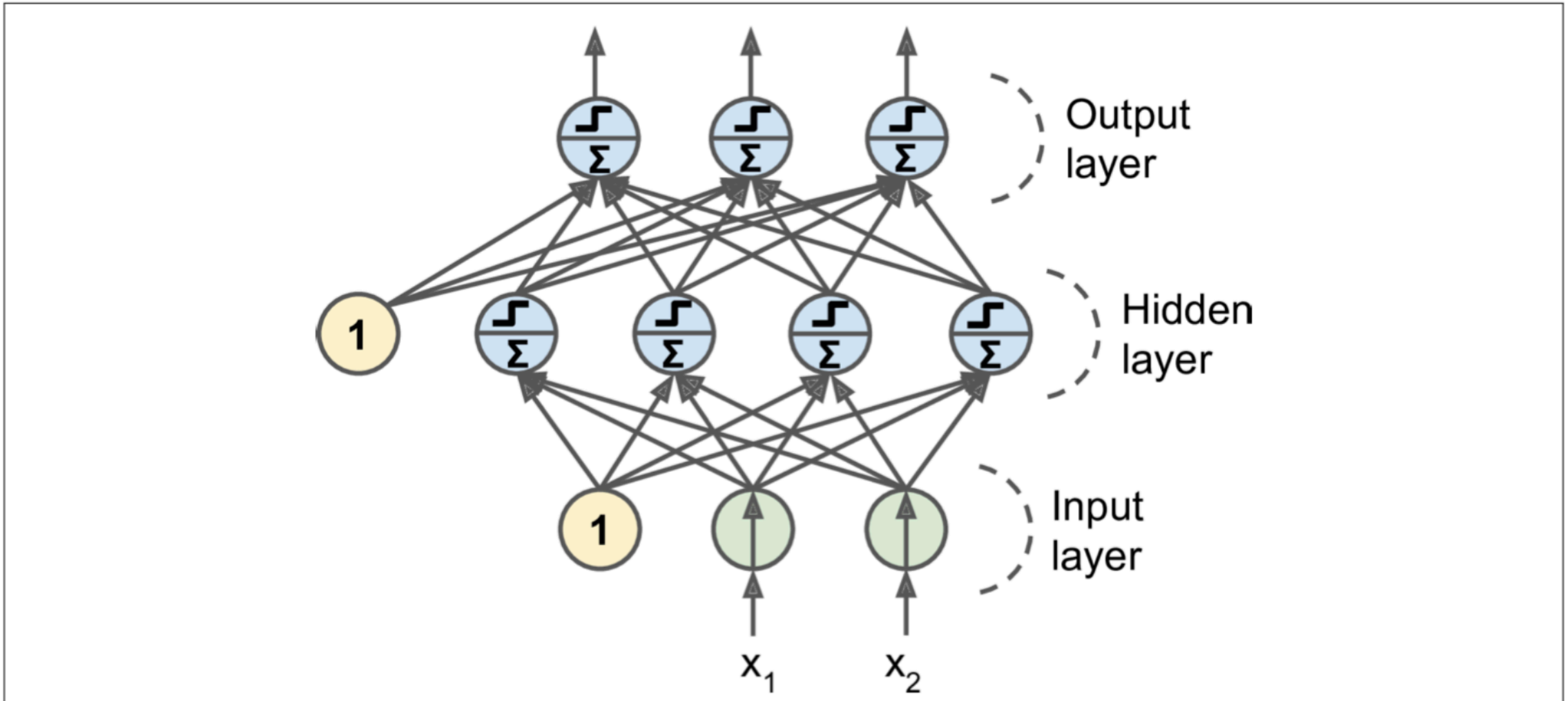
$$\text{heaviside}\,(z) = \begin{cases} 0 \text{ if } z < 0 \end{cases} \qquad \text{sgn}\,(z) = \begin{cases} -1 & \text{if } z < 0 \\ 0 & \text{if } z = 0 \\ +1 & \text{if } z > 0 \end{cases}$$

- Fully connected layer (dense layer):



- This simple structure of perceptrons has some weaknesses. Minsky and Papert (1969). It cannot solve the simple Exclusive OR (XOR) problem.

- By then people had put much faith in Neural Networks, this hurts them a lot. For a long time Neural Networks is neglected. People working in NN had no much respect among researchers in Artificial Intelligence.

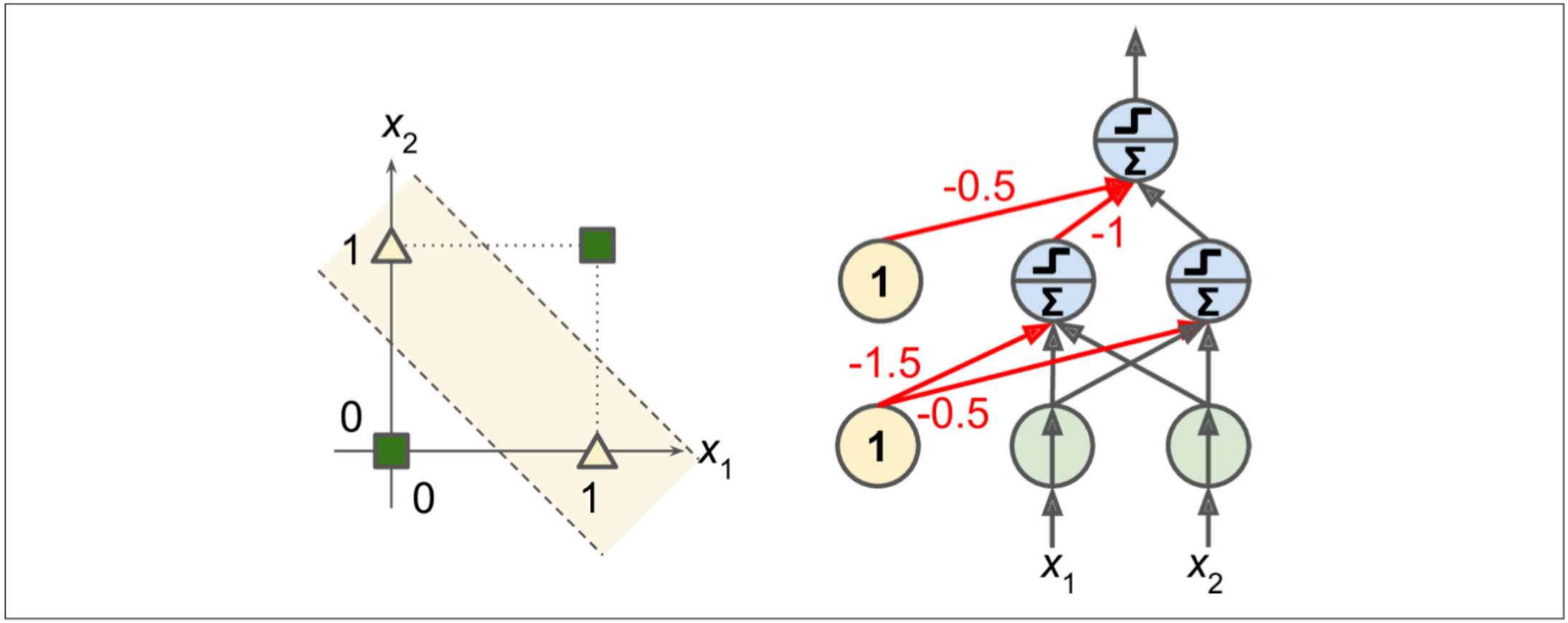


The **Universal Approximation Theorem** is a fundamental result in machine learning and neural networks. Roughly speaking, it says:

> A feedforward neural network with a **single hidden layer** containing a **finite number of neurons** can **approximate any continuous function** on a **compact subset** of $\mathbb{R}^n$, **as closely as desired**, **provided the activation function is sufficiently "nice"** (e.g., nonconstant, bounded, and continuous).

Multi-Layer Perceptron

- But stacking perceptrons can be helpful



- Cybenko (1989), "Approximation by superpositions of a sigmoidal function”
- Hornik (1991), "Approximation capabilities of multilayer feedforward networks”
- Lu et al. (2017), "Expressive Power of Neural Networks”
- Telgarsky (2016), "Benefits of Depth in Neural Networks”
- Eldan and Shamir (2016), "The Power of Depth for Feedforward Neural Networks”
- Raghu et al. (2017), "On the Expressive Power of Deep Neural Networks”

- KEY INNOVATION: The VERY HARD task of constructing X (features), finding the “correct” functional forms is transformed into —> constructing a neural networks; wait and see.

- But now the problem is: which neural networks? I.e., can we find a structure that can approximate EFFICIENTLY (less computational power, fast convergence, better performance)?

    - It becomes a Lego problem. Try fancy (but now it can be vague — intuitively this structure may capture some features of the data) ideas and wait and see.

- Still waiting for the next breakthrough: What is real intelligence? I.e., what makes a human intelligent? How different is human being different from animals?…

    - Neuroscience, biology, and more

Training in NN

- Especially difficult.
- In 1986, Rumelhart, Hinton, Williams (1986) introduced backpropagation.
- The idea:
    - For instances in each mini-batch, feed it into the network and computes the output of each neuron in each layer (just make predictions)
    - Measure the output error
    - evaluate the last layers' neuron's contribution to the error of each output neuron (gradient descent)
    - evaluate the previous hidden layers' neuron's contribution to these error contributions of the last layers' neurons.
    - update the weight of all connections using the error gradients
    - so on and so forth

- In order for the gradients to work, the authors replace the step functions with the logistic function,

$$\sigma(z) = 1/(1 + exp(-z))$$

- Other popular activation functions:
  - $tanh(z) = 2\sigma(2z) - 1$
  - ReLU: $ReLU(z) = max(z,0)$

- NN for regression: just keep one output neuron with (perhaps) the identity function as the activation

- NN for classification: for binary problems, use one output neuron with logistic function as the activation. For multiple classes, use several neurons.

- Image and videos (non-structural or non-tabular data): Convoluted Neural Networks

- Natural Language Processing (with a time series component): Recurrent Neural Networks

- Transformer

- Packages for doing NN: Tensorflow (keras), Pytorch.

What’s next

- Learning by doing

  - Check: scikit-learn, XGBoost, LightGBM, Tensorflow (keras), Pytorch

  - Start doing something immediately. For example, MNIST, IRIS, California housing prices, TITANIC, Image classification etc.

  - Apply to your own problems: predicting stock returns, stock volatilities, default probabilities, fraud detection, …

Resources

- www.kaggle.com
- Stanford’s courses on Computer Vision, Natural Language Processing, etc.
- Coursera
- …