CS3244 : Machine Learning

Semester 1 2023/24

Lecture 6 : Regularization Techniques

Xavier Bresson

https://twitter.com/xbresson

Department of Computer Science

National University of Singapore (NUS)

Material used for preparation

- Prof Kilian Weinberger, CS4780 Cornell, Machine Learning, 2018
  - https://www.cs.cornell.edu/courses/cs4780/2018fa
- Prof Min-Yen Kan, CS3244 NUS, Machine Learning, 2022
  - https://knmnyn.github.io/cs3244-2210
- Prof Mikhail Belkin, 2018
  - https://arxiv.org/pdf/1812.11118.pdf
- Prof Xavier Bresson, CS6208 NUS, Advanced Topics in Artificial Intelligence, 2023

Outline

- Fitting the training set
- Reducing over-fitting
- Loss regularization
- Cross-validation
- Early stopping
- Double descent
- Conclusion

Outline

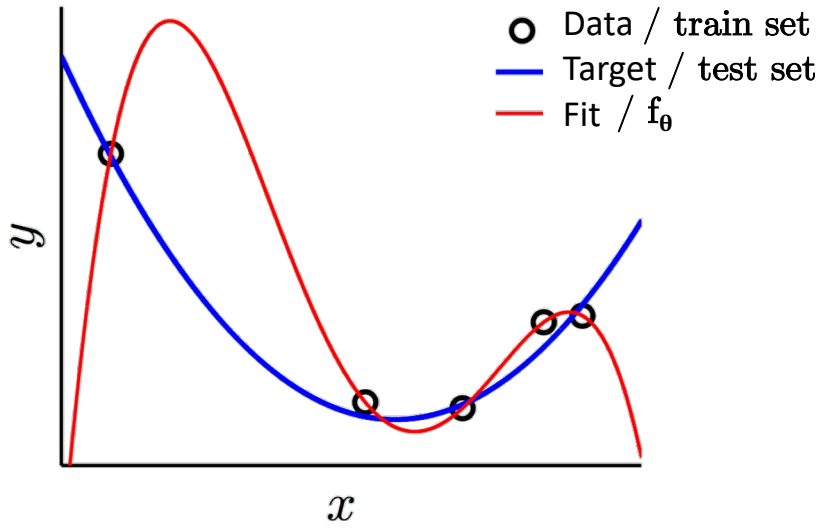
- Fitting the training set
- Reducing over-fitting
- Loss regularization
- Cross-validation
- Early stopping
- Double descent
- Conclusion

Fitting the training set

- Goal : Fitting perfectly the training set
- Example : Regression task
  - Training set : 5 data points sampled from data distribution (blue curve) + small noise
  - Model : 4^{th} order polynomial function, i.e. $f_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

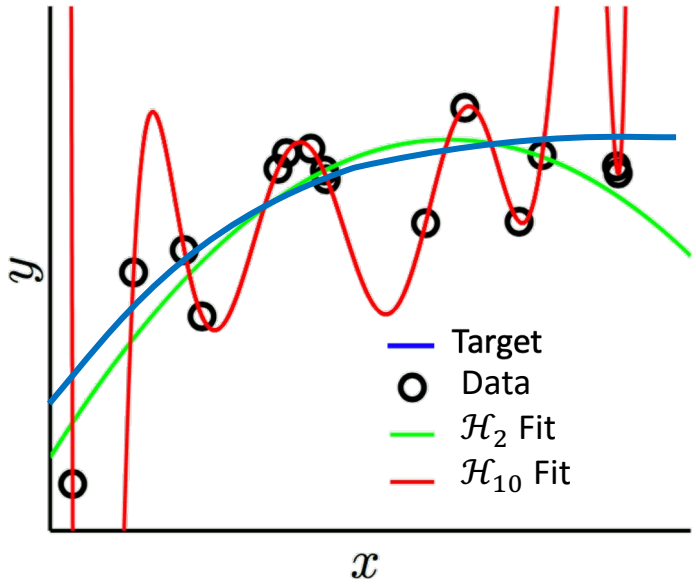
$$\text{Loss}(\ f(x \in S_{train})\ ) = 0\ /\ \text{☺}$$

$$\text{Loss}(\ f(x \in S_{test})\ ) = \text{large}\ /\ \text{☹}$$



Fitting the training set

- Another example



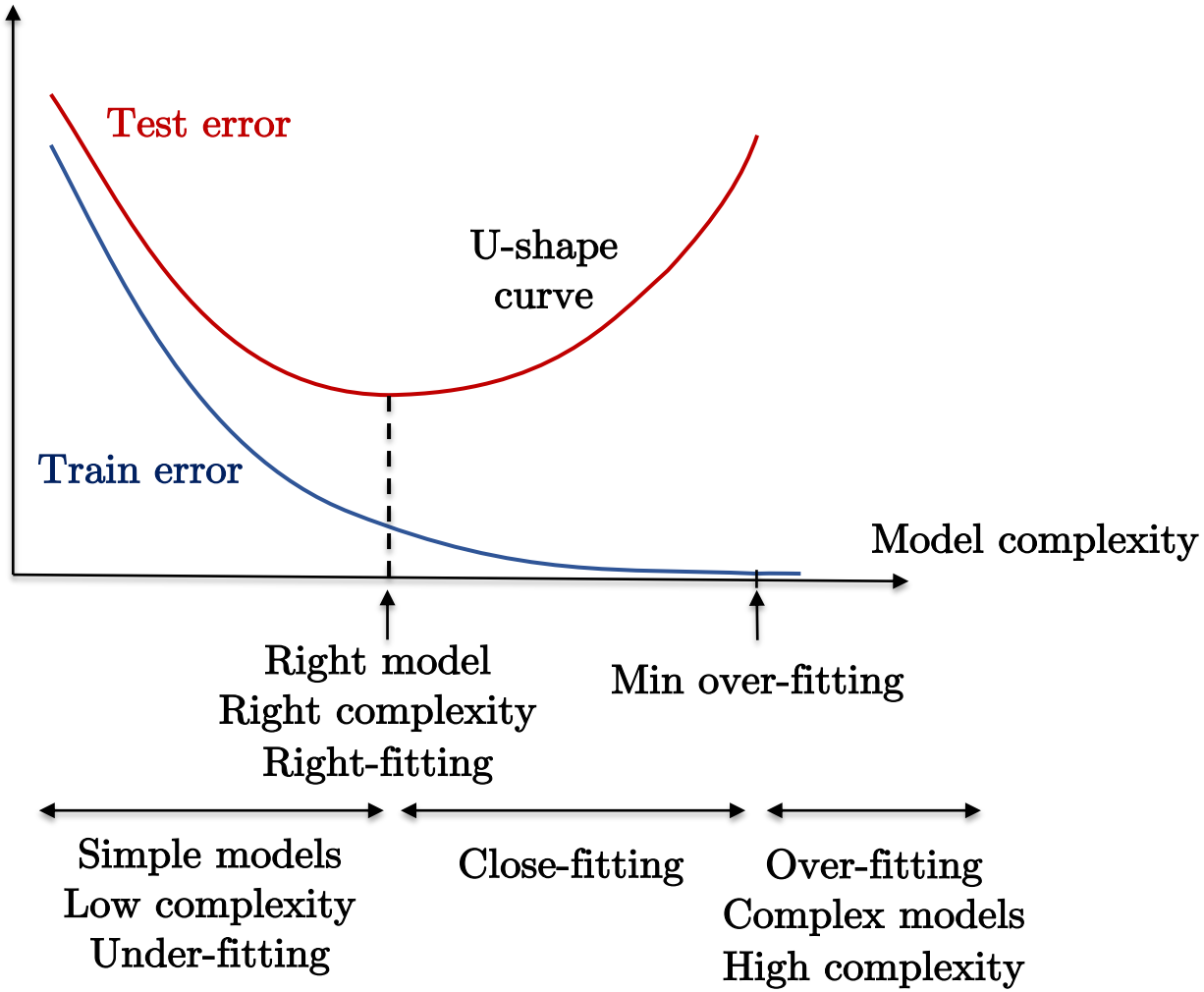
	2nd Order	10th Order
L_{train}	0.029	0.0001
L_{test}	0.120	7680.0
	$L(f(S_{train}))$ = large	$L(f(S_{train}))$ = 0
	$L(f(S_{test}))$ = small	$L(f(S_{test}))$ = large

Fitting the training set

- Under-fitting and over-fitting
  - Two problems encounter when training with a dataset.
  - These problems are related to the degree to which the training set is extrapolated to apply to unknown data.
  - Under-fitting : The learner is not expressive enough. It will make error on the provided training set, i.e. unable to benefit from all information present in the training data. In this case, both the training error and the test error will be high.
  - Over-fitting : The learner is too expressive and will become over-specialized of the training data, unable to extrapolate to unseen data because of high variance. In this situation, the training error will be small and the test error high.

Fitting the training set

- Under-fitting, over-fitting and right-fitting

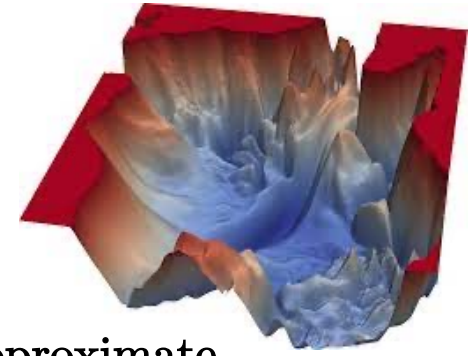


Outline

- Fitting the training set
- Reducing over-fitting
- Loss regularization
- Cross-validation
- Early stopping
- Double descent
- Conclusion

Reducing over-fitting

- How to avoid under-fitting? (easy)
  - Increase expressivity of the learner.
- How to avoid over-fitting? (difficult)
  - Use regularization loss with cross-validation to estimate the regularization parameter
  - Early stopping with a validation set
  - Regularization with stochastic gradient descent (SGD)
    - SGD not only speeds up gradient descent technique by computing an approximate gradient with a mini-batch of data points, it also regularizes the predictive function w.r.t. its parameters $\boldsymbol{\theta}$, allowing better generalization performance.
    - Theoretically, we should use mini-batch of a single data point for best generalization but it would be too slow. Using mini-batch of size e.g. 512 data points is the best trade-off speed and accuracy.



Outline

- Fitting the training set
- Reducing over-fitting
- Loss regularization
- Cross-validation
- Early stopping
- Double descent
- Conclusion

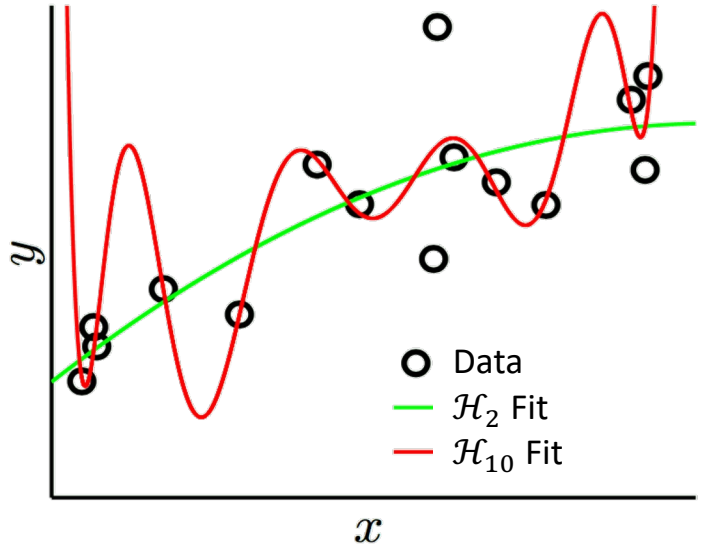
Loss regularization

- Can we reduce the hypothesis space $\mathcal{H}_{10}$ to $\mathcal{H}_2$?

$$\text{We have } \mathcal{H}_{10} = \{ f_{\boldsymbol{\theta}}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \cdots + \theta_{10} x^{10} \}$$

$$\text{and } \mathcal{H}_2 = \{ f_{\boldsymbol{\theta}}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 \}$$

$$\text{Then } \mathcal{H}_{10} = \mathcal{H}_2 \text{ when } \theta_3 = \theta_4 = \cdots = \theta_{10} = 0$$



Loss regularization

- Let us recall the MSE optimization problem for the regression task :

$$\min_{\boldsymbol{\theta}} L_{\mathcal{H}}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{j=1}^{n}(f_{\mathcal{H}}(\mathbf{x}^{(j)}) - y^{(j)})^2 \quad \text{unconstrained optimization}$$

- Equivalent optimization problems :

$$\min_{\boldsymbol{\theta}} L_{\mathcal{H}_2}(\boldsymbol{\theta}) \Leftrightarrow \min_{\boldsymbol{\theta}} L_{\mathcal{H}_{10}}(\boldsymbol{\theta}) \text{ such that } \theta_3 = \theta_4 = \cdots = \theta_{10} = 0$$

constrained optimization with hard constraints

Loss regularization

- Let us relax the hard constraints, $\theta_{j\geq 3} = 0$ to let the optimization select the best value of $\theta_{j\geq 3}$:

$$\min_{\boldsymbol{\theta}} L_{\mathcal{H}_{10}}(\boldsymbol{\theta}) \text{ such that } \sum_{j\geq 3}^{d} \theta_j^2 \leq C\,, C > 0$$

constrained optimization with soft constraints

- Hyper-parameter C controls the amount of non-zero for the parameters $\theta_{j\geq 3}$.
  - Small value C implies most $\theta_{j\geq 3}$ close to zero, i.e. $\mathcal{H}_{10} = \mathcal{H}_2$.
  - Large value C provides non-zero $\theta_{j\geq 3}$, i.e. $\mathcal{H}_{10} \gg \mathcal{H}_2$.

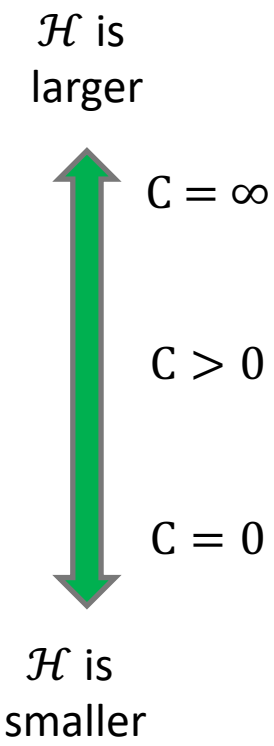
Loss regularization

- Relationship between hypothesis spaces :

$$\mathcal{H}_{10} = \{ f_{\boldsymbol{\theta}}(x) = \theta_0 + \theta_1 x + \cdots + \theta_{10} x^{10} \}$$

$$\mathcal{H}_C = \{ f_{\boldsymbol{\theta}}(x) = \theta_0 + \theta_1 x + \cdots + \theta_{10} x^{10} \} \text{ such that } \sum_{j=3}^{10} \theta_j^2 \leq C$$

$$\mathcal{H}_2 = \{ f_{\boldsymbol{\theta}}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 \}$$
$$= \{ f_{\boldsymbol{\theta}}(x) = \theta_0 + \theta_1 x + \cdots + \theta_{10} x^{10} \} \text{ such that } \theta_3 = \theta_4 = \cdots = \theta_{10} = 0$$



Loss regularization

- Let us consider the general constrained regularized problem :

$$\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) \text{ such that } \sum_{j=0}^{d} \theta_j^2 = \boldsymbol{\theta}^\top \boldsymbol{\theta} \leq C, C > 0 \qquad (1)$$

- There exists an equivalent unconstrained optimization problem (easier to solve) :

$$\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) + \lambda \, \boldsymbol{\theta}^\top \boldsymbol{\theta} \, , \lambda > 0 \qquad (2)$$

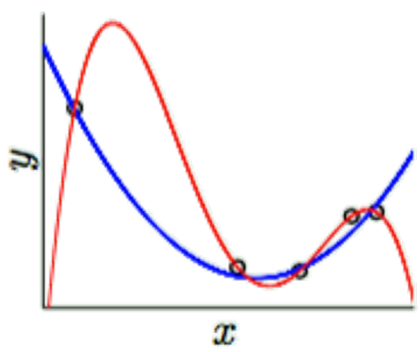
- For each value C, there exists a value λ such that (1) is equivalent to (2) (Lagrange multiplier).
  - Additionally, $C \propto 1/\lambda$

Loss regularization

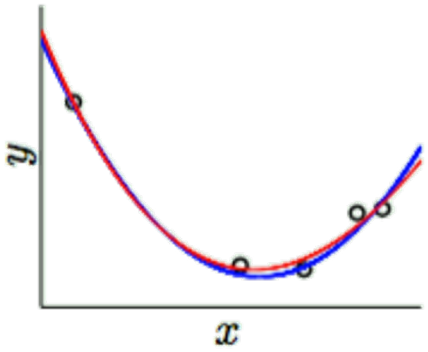
- Study the influence of the regularization parameter λ on the solution of

$$\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) + \lambda\, \boldsymbol{\theta}^\top \boldsymbol{\theta} \quad \Leftrightarrow \quad \min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) \text{ s.t. } \boldsymbol{\theta}^\top \boldsymbol{\theta} \leq C$$

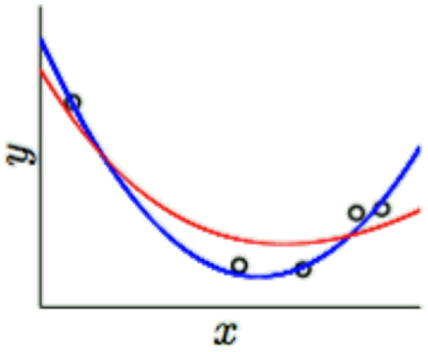
- Example : Regression task with 4^{th} order polynomial function



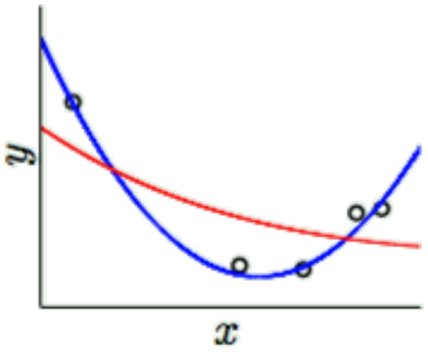
$\lambda = 0$
$C = \infty$
Over-fitting



$\lambda = 10$
$C = 0.1$
Right-fitting



$\lambda = 100$
$C = 0.01$
Under-fitting



$\lambda = 1{,}000$
$C = 0.001$
Under-fitting

Loss regularization

- Normal equations for linear regression with loss regularization :

Original MSE loss :

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

Set gradient of loss to zero :

$$\nabla L = \frac{\partial L}{\partial \boldsymbol{\theta}} = \mathbf{0} \Rightarrow \mathbf{X}^{\top}(\mathbf{X}\boldsymbol{\theta} - \mathbf{y}) = \mathbf{0} \Rightarrow \boldsymbol{\theta} = (\mathbf{X}^{\top}\mathbf{X})^{-1}\mathbf{X}^{\top}\mathbf{y}$$

Regularized MSE loss :

$$\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \frac{1}{n}\|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|^2 \text{ s.t. } \boldsymbol{\theta}^{\top}\boldsymbol{\theta} \leq C$$
$$\Leftrightarrow \min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \frac{1}{n}\|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|^2 + \lambda\, \boldsymbol{\theta}^{\top}\boldsymbol{\theta}$$

Set gradient of loss to zero :

$$\nabla L = \frac{\partial L}{\partial \boldsymbol{\theta}} = \mathbf{0} \Rightarrow \frac{1}{n}\mathbf{X}^{\top}(\mathbf{X}\boldsymbol{\theta} - \mathbf{y}) + \lambda\boldsymbol{\theta} = \mathbf{0} \Rightarrow \boldsymbol{\theta} = (\mathbf{X}^{\top}\mathbf{X} + \lambda n\mathbf{I})^{-1}\mathbf{X}^{\top}\mathbf{y}$$

Loss regularization

- Understanding the solution of normal equations :

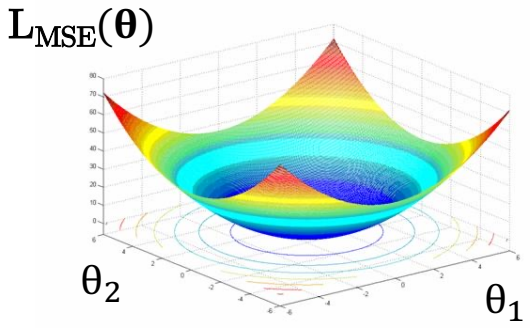
$$\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = L_{MSE}(\boldsymbol{\theta}) \text{ s.t. } L_{REG}(\boldsymbol{\theta}) \leq C$$

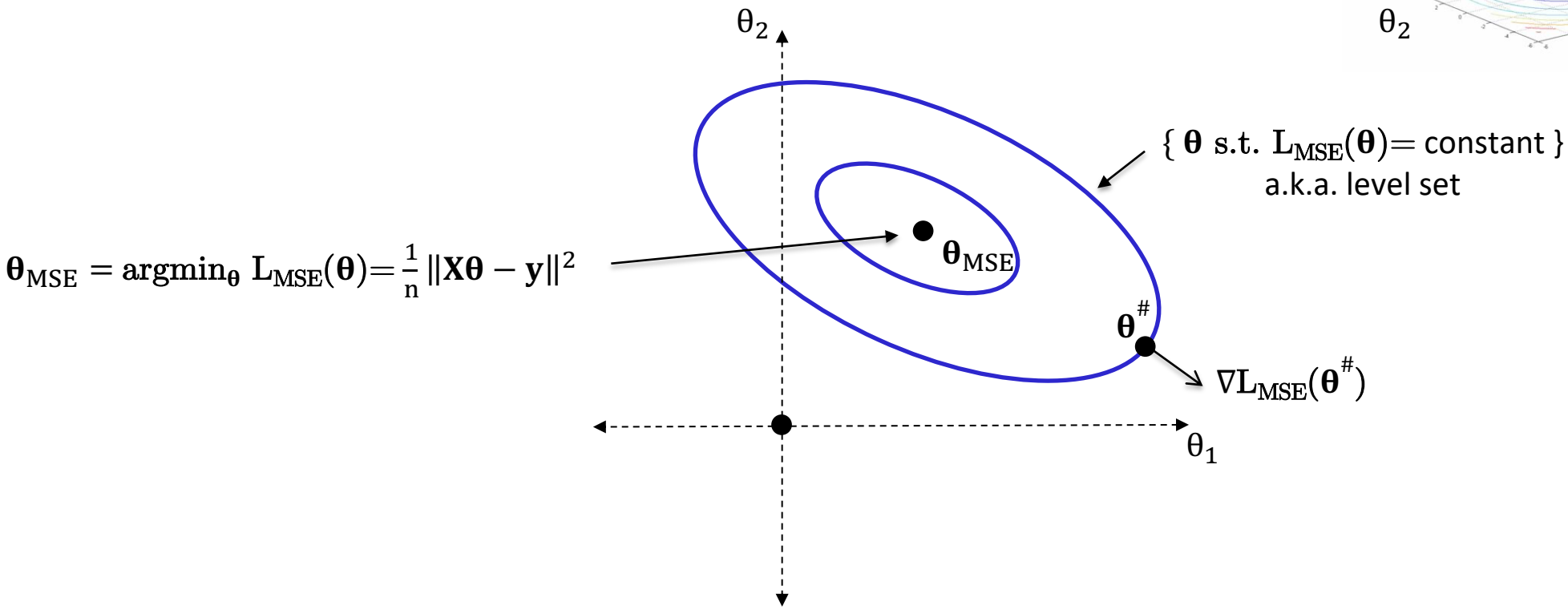
$$\text{with } L_{MSE}(\boldsymbol{\theta}) = \frac{1}{n} \|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|^2 \text{ and } L_{REG}(\boldsymbol{\theta}) = \boldsymbol{\theta}^\top \boldsymbol{\theta} = \|\boldsymbol{\theta}\|^2$$

- Let us plot the landscape of the L_{MSE} loss and the L_{REG} loss.

Loss regularization

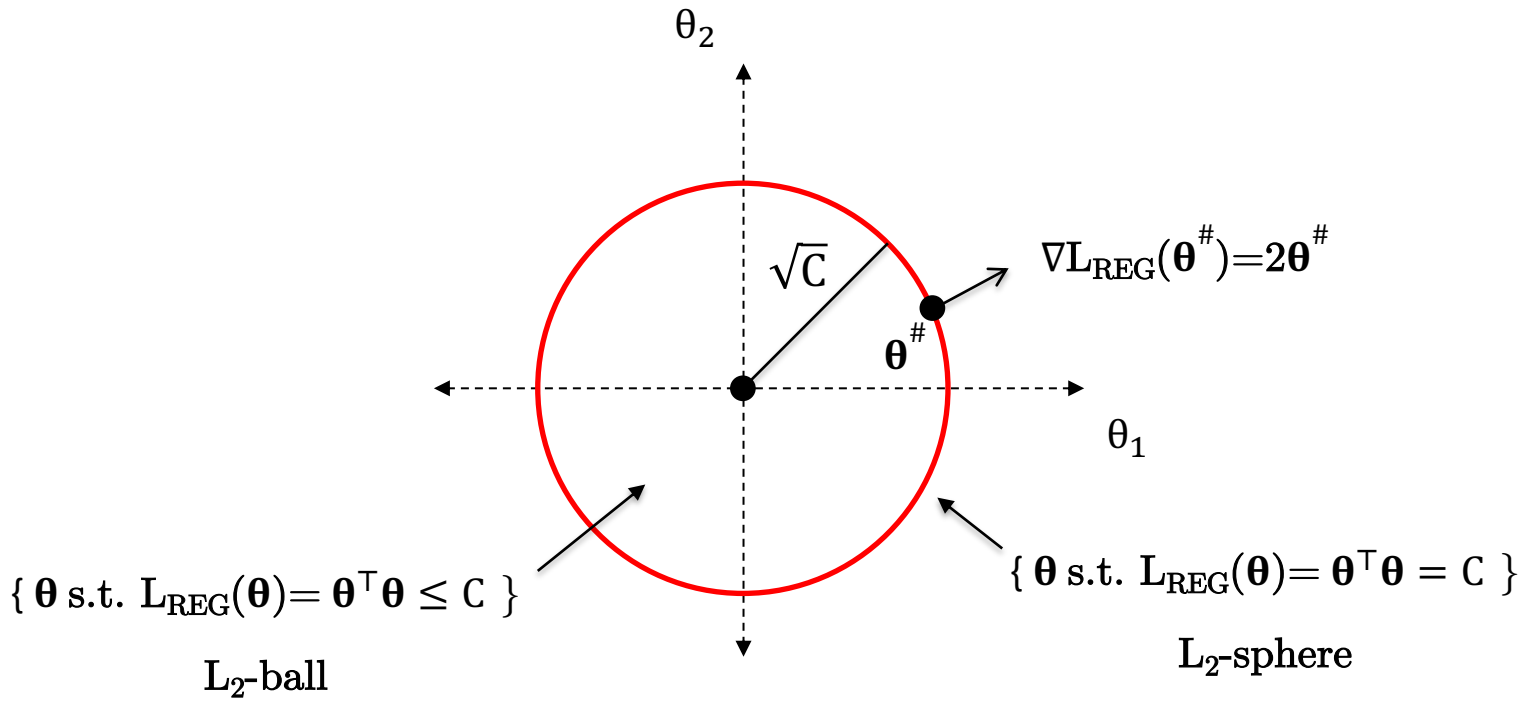
- Landscape of L_{MSE} loss : $L_{MSE}(\boldsymbol{\theta}) = \frac{1}{n}\|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|^2$
  - Quadratic and convex function.
  - Let us suppose we have two parameters, i.e. $\boldsymbol{\theta}=(\theta_1, \theta_2)$, for visualization.





Loss regularization

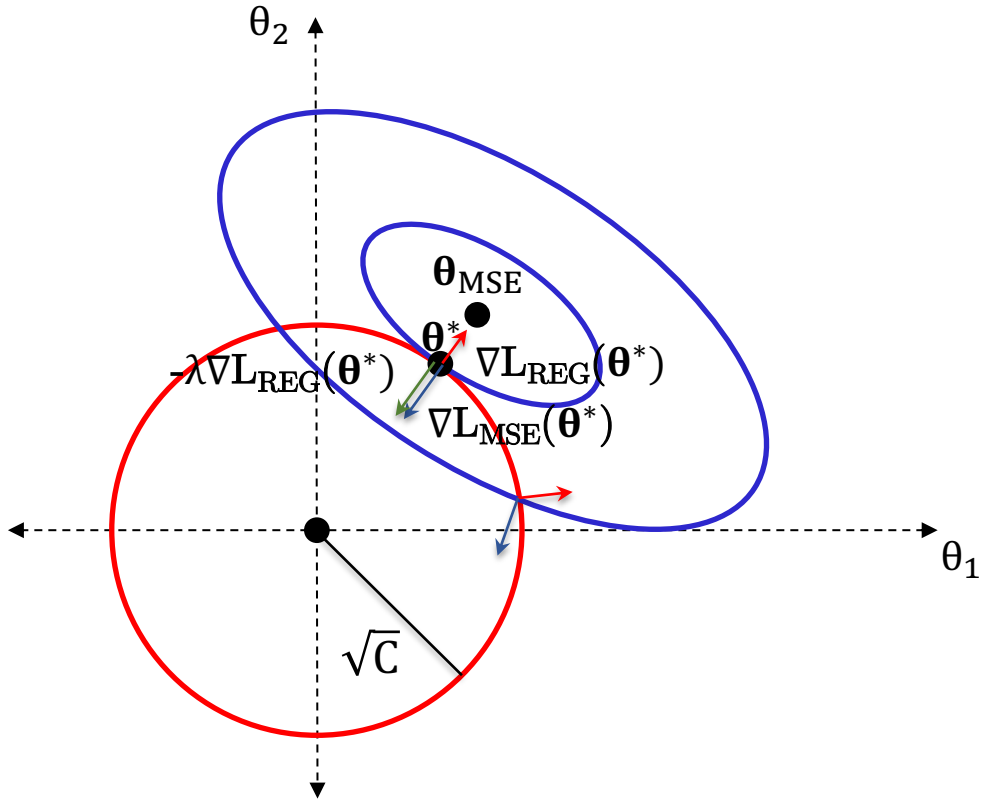
- Landscape of L_{REG} loss : $L_{REG}(\boldsymbol{\theta}) = \boldsymbol{\theta}^\top \boldsymbol{\theta} = \|\boldsymbol{\theta}\|^2$
  - Quadratic and convex function.



Loss regularization

- Minimizer of the total loss : $\boldsymbol{\theta}^* = \text{argmin}_{\boldsymbol{\theta}}\ L_{MSE}(\boldsymbol{\theta})$ s.t. $L_{REG}(\boldsymbol{\theta}) \leq C$

  $\Leftrightarrow\ \boldsymbol{\theta}^* = \text{argmin}_{\boldsymbol{\theta}}\ L_{MSE}(\boldsymbol{\theta}) + \lambda\, L_{REG}(\boldsymbol{\theta})$



Solution $\boldsymbol{\theta}^*$ is as close as the MSE solution $\boldsymbol{\theta}_{MSE}$ as allowed by the L2-ball constraint $\|\boldsymbol{\theta}\|^2 \leq C$.

Gradient of loss at $\boldsymbol{\theta}^*$:

$\nabla\,(L_{MSE}(\boldsymbol{\theta}^*) + \lambda\, L_{REG}(\boldsymbol{\theta}^*)) = 0$

$\Rightarrow \nabla L_{MSE}(\boldsymbol{\theta}^*) = -\lambda \nabla L_{REG}(\boldsymbol{\theta}^*)$

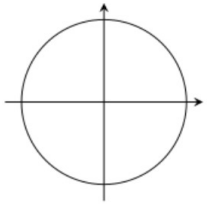
Gradients of L_{MSE} and L_{REG} are aligned (in the opposite direction) at the solution.

Loss regularization

- The L_2-ball regularization can be generalized to L_p-ball, $p \in [0, +\infty]$.

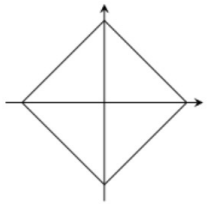
$$\min_{\boldsymbol{\theta}} L_{MSE}(\boldsymbol{\theta}) \text{ s.t. } \|\boldsymbol{\theta}\|_p^p \leq C \Leftrightarrow \min_{\boldsymbol{\theta}} L_{MSE}(\boldsymbol{\theta}) + \|\boldsymbol{\theta}\|_p^p \text{ where} \|\boldsymbol{\theta}\|_p = \left(\sum_{i=1}^{d} |\theta_i|^p\right)^{\frac{1}{p}}$$

- L_2-ball/L_2 regularization, a.k.a. weight decay
  - Advantages : Strictly convex, differentiable, fast optimization, robust w.r.t. perturbation.
  - Limitations : Although θ_i values are minimized, solutions are dense, i.e. $\theta_i > 0$.
    This means no feature selection in e.g. $f_{\boldsymbol{\theta}}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \cdots + \theta_{10} x^{10}$,
    as all data features are used for prediction.



$p = 2$

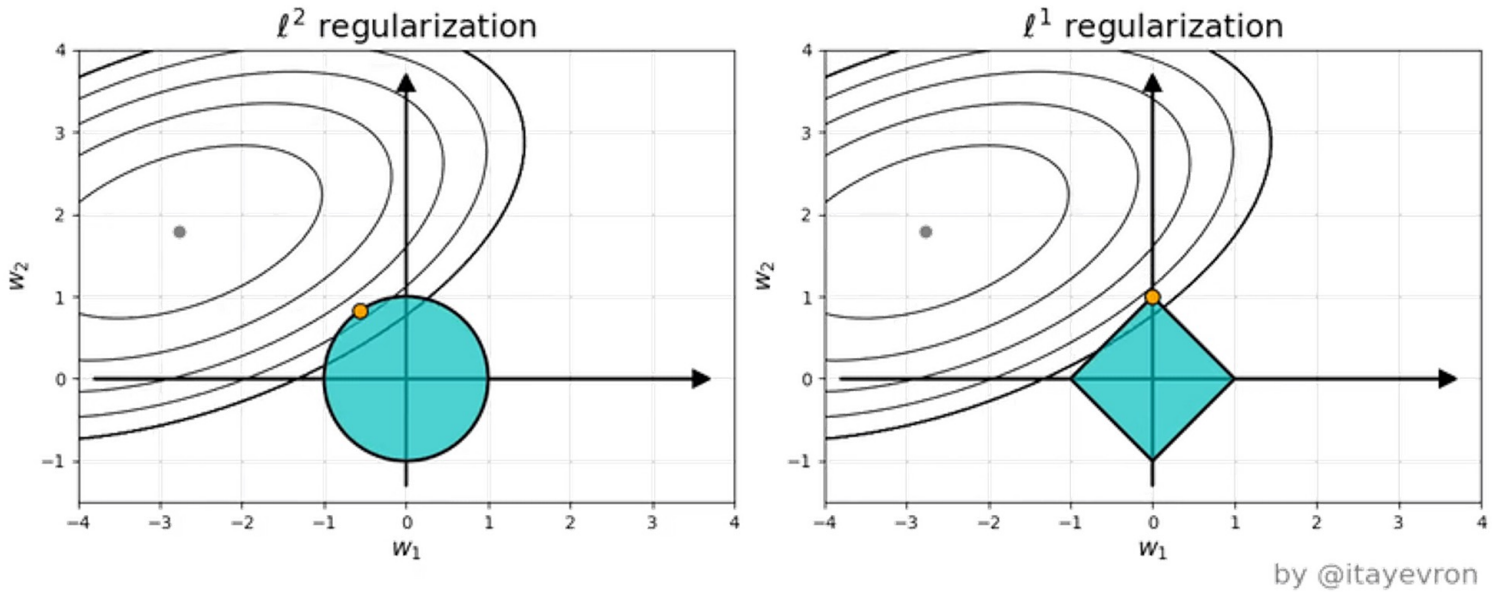
- L_1 regularization
  - Advantages : Convex (but not strictly), fast optimization algorithms exist, robust w.r.t. perturbation, solutions are guaranteed to be sparse meaning feature selection, as only a few data features are used for prediction.
  - Limitations : Not differentiable at the origin.



$p = 1$

Loss regularization

- MSE + L_2 regularization vs. L_1 regularization

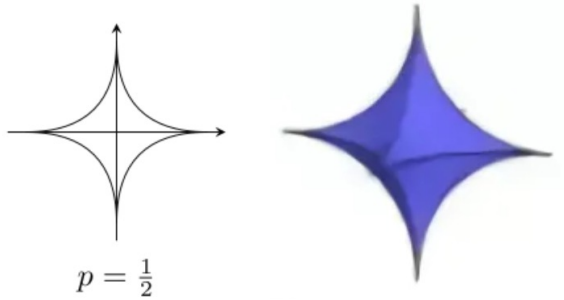


The probability to have a solution on the axes is almost zero, most solutions lie on the quadrants of the L_2 ball.

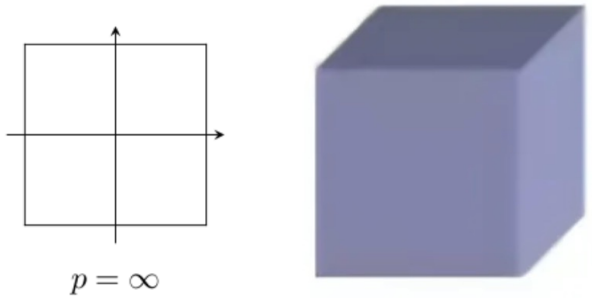
The probability to have a solution on diagonal edges is almost zero, most solutions lie on a tip of the L_1 ball.

Loss regularization

- L_p regularization, $0<p\leq 1$
  - Advantages : Very sparse solutions, better than L_1 regularization.
  - Limitations : Non-convex, non-differentiable, solution depends on initial condition.



- L_p regularization, $p = \infty$
  - Never used in practice (not stable)



Loss regularization

- L_p regularized loss for any predictive task :

$$\min_{\boldsymbol{\theta}} L_{Task}(\boldsymbol{\theta}) \text{ s.t.} \|\boldsymbol{\theta}\|_p^p \leq C \Leftrightarrow \min_{\boldsymbol{\theta}} L_{Task}(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_p^p$$

$$\text{where } L_{Task}(\boldsymbol{\theta}) = \frac{1}{n} \sum_{j=1}^{n} \ell_{Task}(f_{\boldsymbol{\theta}}(\mathbf{x}^{(j)}), y^{(j)})$$

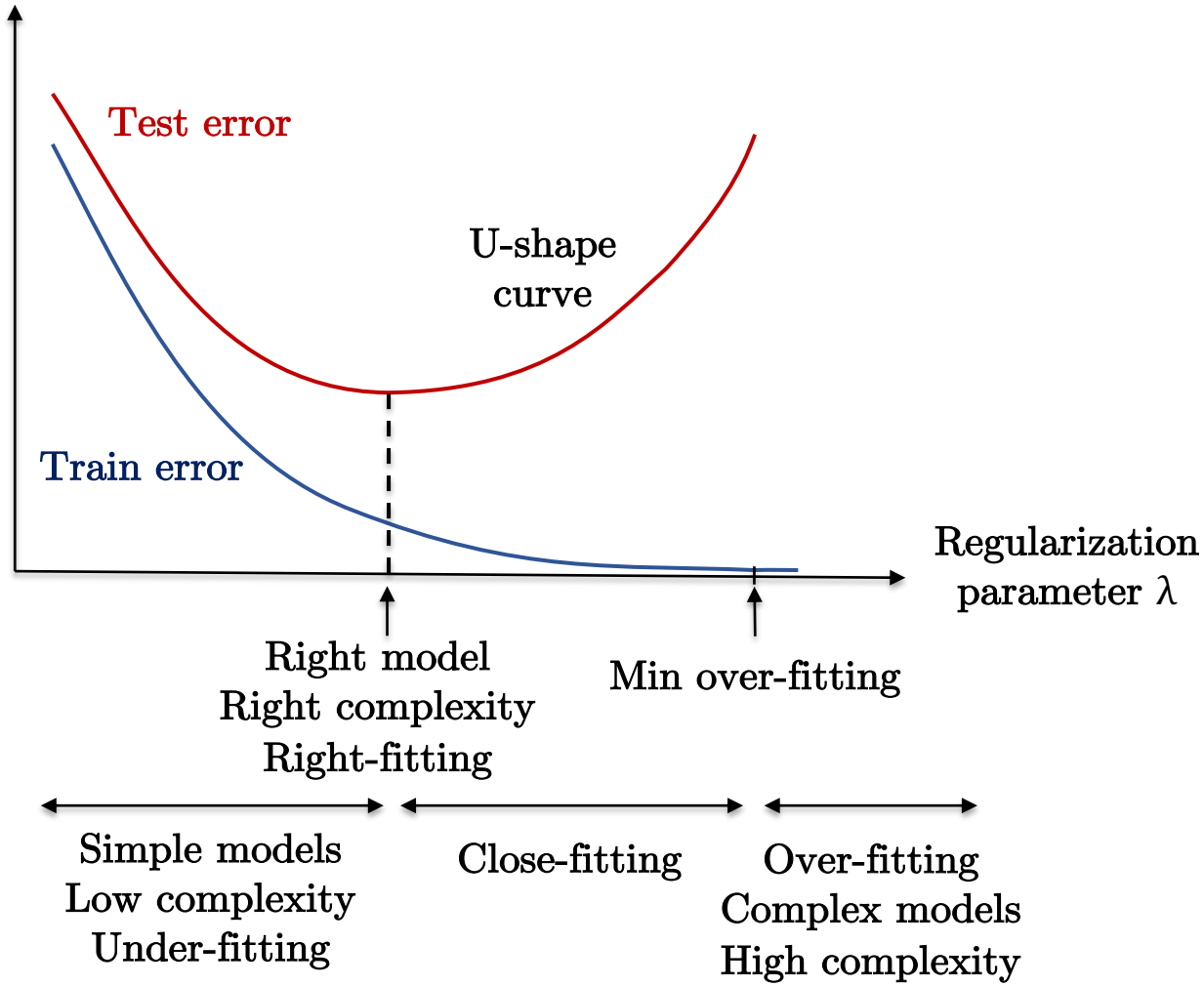
$$\text{and} \|\boldsymbol{\theta}\|_p^p = \sum_{i=1}^{d} |\theta_i|^p$$

Loss regularization

- Summary
  - Adding a regularization loss, a.k.a. regularizer, can reduce over-fitting.
  - With the right amount of regularization, controlled by the hyper-parameters λ (or C), the regularizer can decrease the complexity of the predictive model, i.e. its variance, without affecting the bias.
  - With no regularization, the model over-specializes to the training data, i.e. high variance.
  - With too much regularization, the model becomes too simple, i.e. high bias.

Loss regularization

- How to choose λ, i.e. the right amount of regularization?

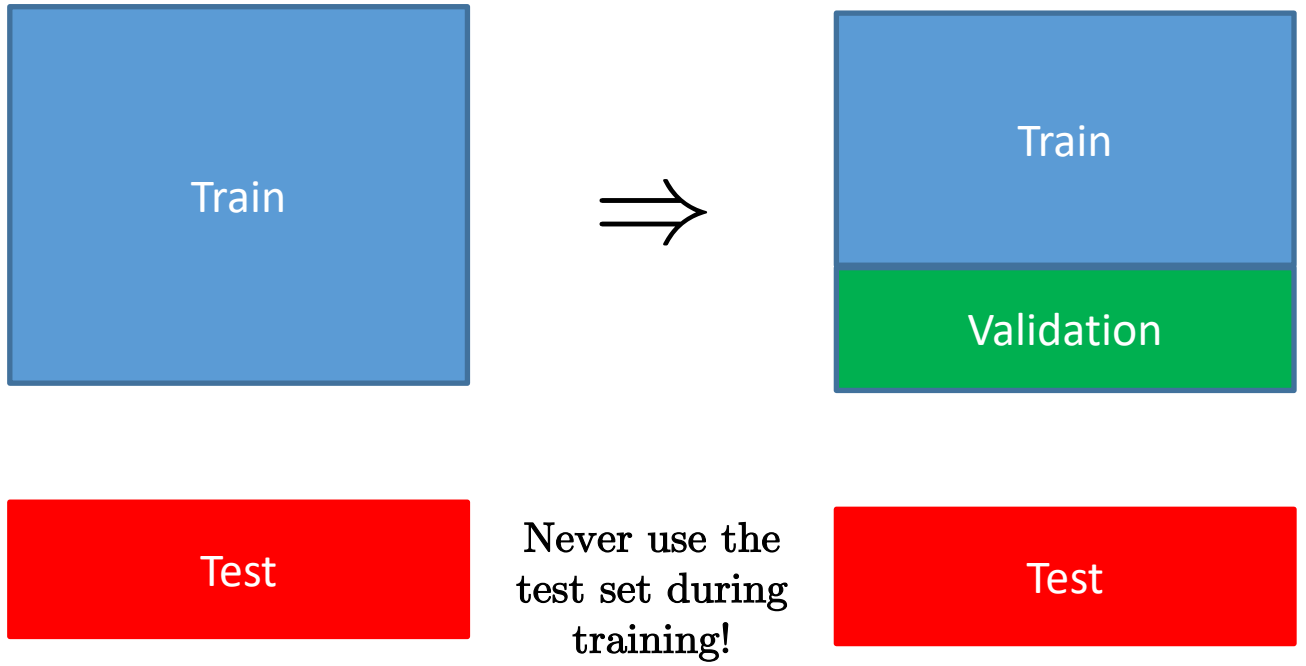


Outline

- Fitting the training set
- Reducing over-fitting
- Loss regularization
- Cross-validation
- Early stopping
- Double descent
- Conclusion

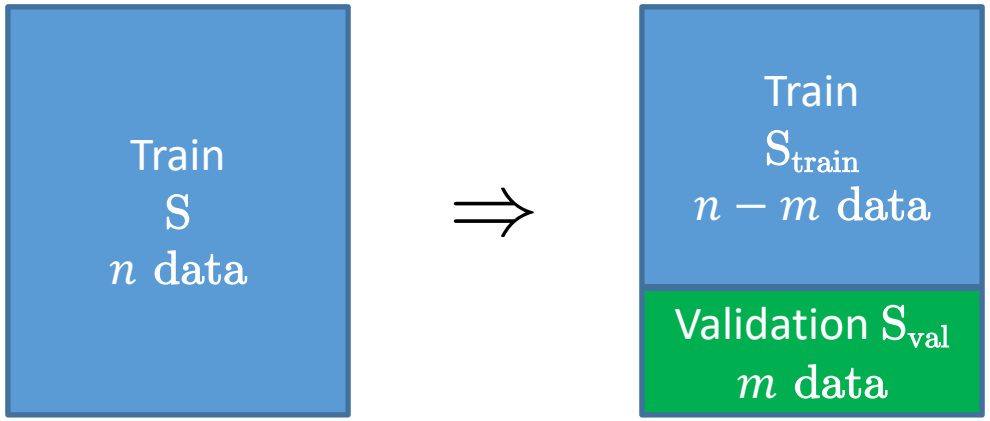
Cross-validation

- We need a surrogate of the test set to estimate the regularization parameter λ which identifies the right model complexity that minimizes the test error.
- The simplest operation is to split the training set into two datasets; a smaller training set and a validation set.



Cross-validation

- Give a training set S of n data points, S is split into
  - A smaller training set S_{train} of $n - m$ data points.
  - A validation set S_{val} of m data points.
- How to use S_{val} to estimate the regularization value λ?



Cross-validation

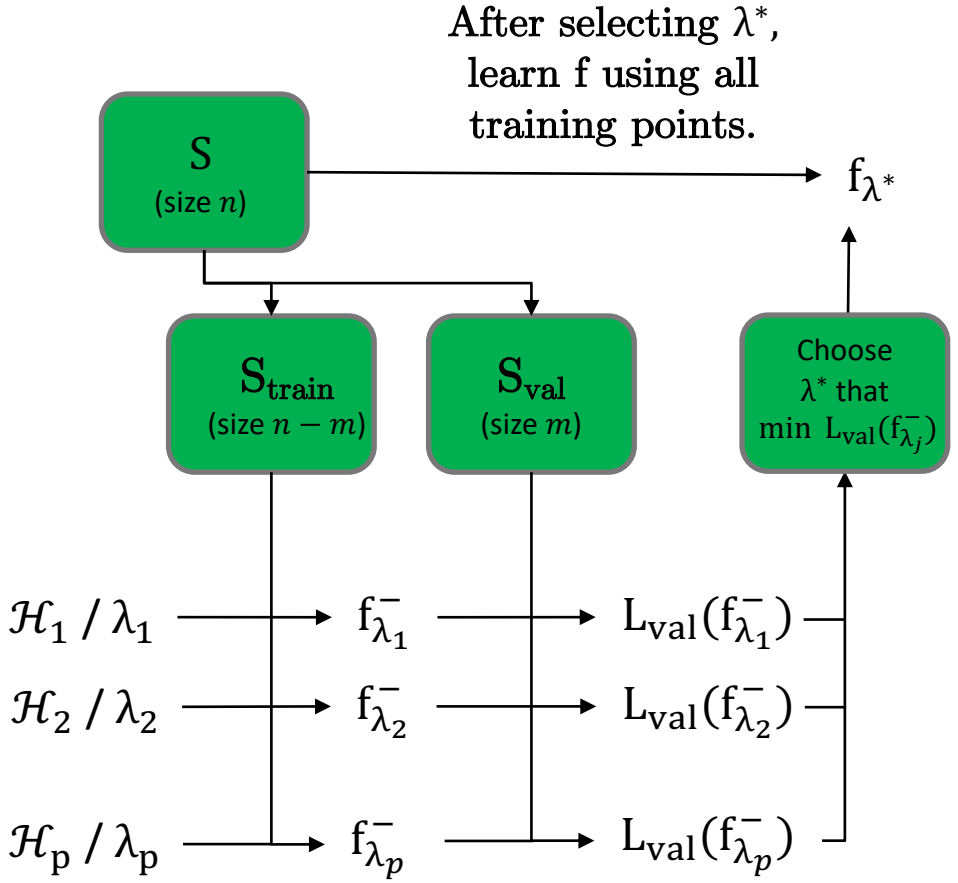
- Use p hypothesis/values to estimate λ :

We consider p values $\lambda_1, \ldots, \lambda_p$.

Use S_{train} to learn $f_{\lambda_j}^-$ for each λ value.

Evaluate $f_{\lambda_j}^-$ using S_{val} : $L_{val}\left(f_{\lambda_j}^-\right)$ for $j = 1, \ldots, p$

Select value $\lambda^* = \lambda_j$ with smallest L_{val}



Cross-validation

- How robust is the estimation of the validation loss L_{val} ?
- Suppose that
  - $\ell(f_\theta(x), y)$ is the loss value for the data point (x,y)
  - $\mathbb{E}_{(x,y)\sim U}[\ell(f_\theta(x), y)] = L_{test}(f_\theta)$, which is the mean error of the predictive function f_θ applied to U, the set of all unseen points by f_θ during training, i.e. S_{test} and S_{val}.
  - $Var_{(x,y)\sim U}[\ell(f_\theta(x), y)] = \sigma^2$, which corresponds to the variance of the prediction error.
  - $L_{val}(f_\theta) = \frac{1}{m}\sum_{k=1}^{m} \ell(f_\theta(x^{(k)}), y^{(k)})$ is the validation loss.

Cross-validation

- Then, we have
  - $\mathbb{E}_{(x,y)\sim U}\left[L_{val}(f_\theta) \right] = \frac{1}{m}\sum_{k=1}^{m} \mathbb{E}_{(x,y)\sim U}\left[\ell(f_\theta(x^{(k)}), y^{(k)}) \right] = L_{test}(f_\theta)$
  - $\mathrm{Var}_{(x,y)\sim U}\left[L_{val}(f_\theta) \right] = \frac{1}{m^2}\sum_{k=1}^{m} \mathrm{Var}_{(x,y)\sim U}\left[\ell(f_\theta(x^{(k)}), y^{(k)}) \right] = \frac{1}{m^2} \cdot m\,\sigma^2 = \frac{\sigma^2}{m} \Rightarrow \mathrm{Std} = O\left(\frac{1}{\sqrt{m}}\right)$
  - $L_{val} = L_{test} \pm O\left(\frac{1}{\sqrt{m}}\right)$
- Consequence : A small validation set does not provide a good estimate of the test error

Cross-validation

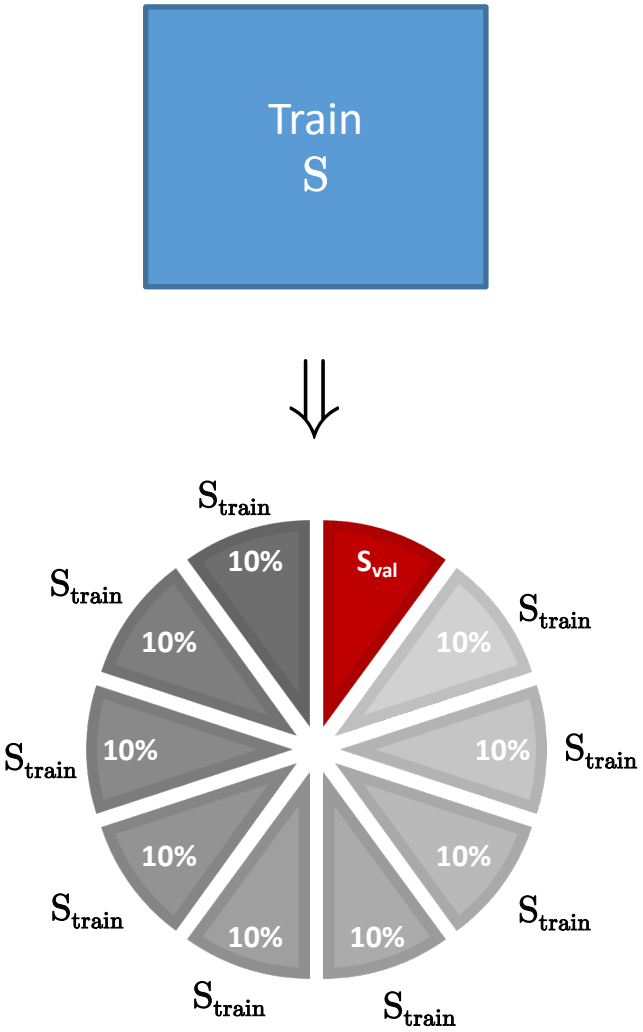
- In practice, we have two situations
  - Big datasets
    - Modern situation, training sets are large, e.g. millions of data points.
    - We can use a small fraction, e.g. m = 100,000 data points as validation set.
    - The validation set will approximate well the test set distribution.
  - Small datasets
    - Situation before 2012 or today for highly expensive or challenging datasets to collect (e.g. nuclear fusion) or for protected datasets (e.g. medical data).
    - For limited datasets, e.g. $n = 1{,}000$ data points, it is not possible to get simultaneously good estimates of the predictive function and the validation set.

Cross-validation

- For small datasets, we have 2 opposite cases
  - Recall : Model f is trained on the full training set of n data, and f^- is trained on the $n-m$ training set
  - Case #1 : Small number m of validation data / large number $n-m$ of training data
    - Advantage : $L_{test}(f) \approx L_{test}(f^-)$ as f^- is well estimated.
    - Limitation : $L_{test}(f^-) \neq L_{val}(f^-)$ as the validation set is too small.
  - Case #2 : Large number m of validation data / small number $n-m$ of training data
    - Advantage : $L_{test}(f^-) \approx L_{val}(f^-)$ as the validation set is large enough.
    - Limitation : $L_{test}(f) \neq L_{test}(f^-)$ as f^- is badly estimated.
- How to reconcile the two cases?

Cross-validation

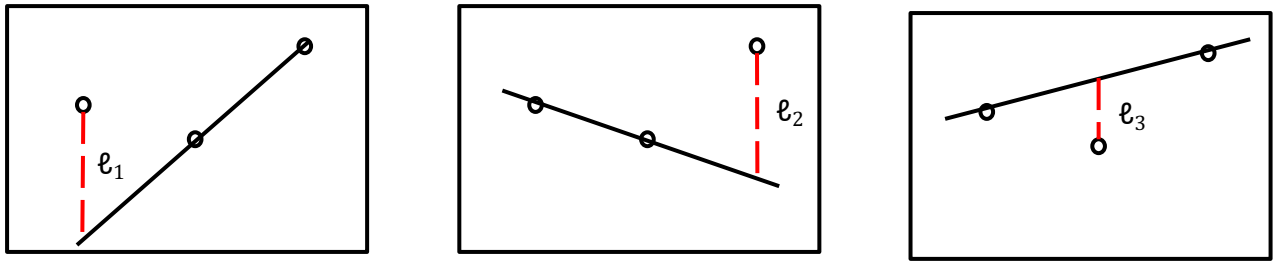
- k-fold cross-validation technique :
  - Split the original training set into k parts, i.e. each fold has n/k data points.
  - Repeat for all folds : Train on k-1 parts and leave one part out as validation set.
- Advantages
  - Each data in the original training set will be used as a validation data.
  - For each fold, we have a large training set to train a good learner, i.e. $L_{test}(f) \approx L_{test}(f^-)$.
  - We also have a good estimate of the validation error by averaging the validation error over all folds, i.e. $L_{test} \approx \text{mean}_{folds}\, L_{val}(f^-)$.



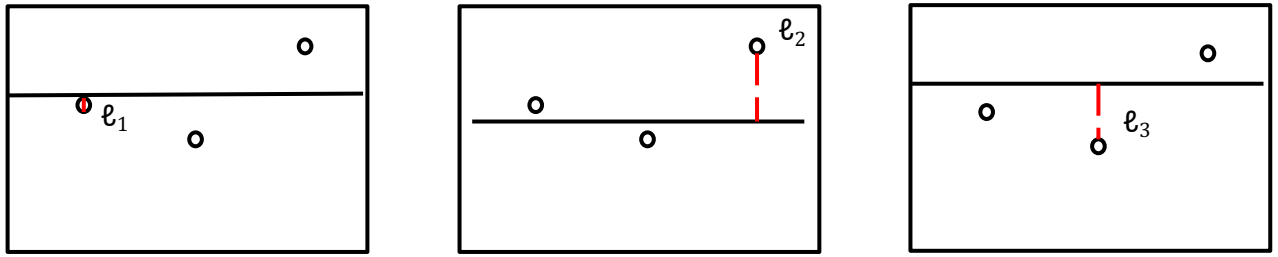
Cross-validation

- Example : Model selection using cross-validation
  - Models : Linear and constant models
  - Original training set S : $n = 3$ data points
  - Cross-validation value : $m = 1$ (validation set S_{val}), $n - m = 2$ (training set S_{train})

Linear model
$L_{CV} = 8.2$



Constant model
$L_{CV} = 4.3$



Cross-validation shows that the constant model is a better fit than the linear model for this dataset.

$$L_{CV} = \frac{1}{3}(\ell_1 + \ell_2 + \ell_3)$$

Cross-validation

- In practice
  - Hypotheses are an arbitrary set of choices, e.g. choices of predictive models $\{f_\theta\}$, choices of parameter values $\{\lambda\}$, etc.
  - Note that parameters s.a. the number of layers in neural networks is not differentiable, i.e. gradient descent cannot be used to select their optimal value.
  - For very small datasets, we cannot afford to leave out more than a single training data for validation, so we use $k = n$ folds (i.e. $m = 1$ validation point), a.k.a. Leave One Out Cross Validation (LOOCV).
  - Telescopic search is a standard two-step approach to determine parameter values.
    - First step : Find the best order of magnitude for λ, e.g. $\lambda = 0.01,0.1,1,10,100$.
    - Second step : Do a fine-grained search around the best λ found in first step. For example, if 10 is the best performing value from first step, then try out $\lambda=3,6,10,30,60,90$.

Cross-validation

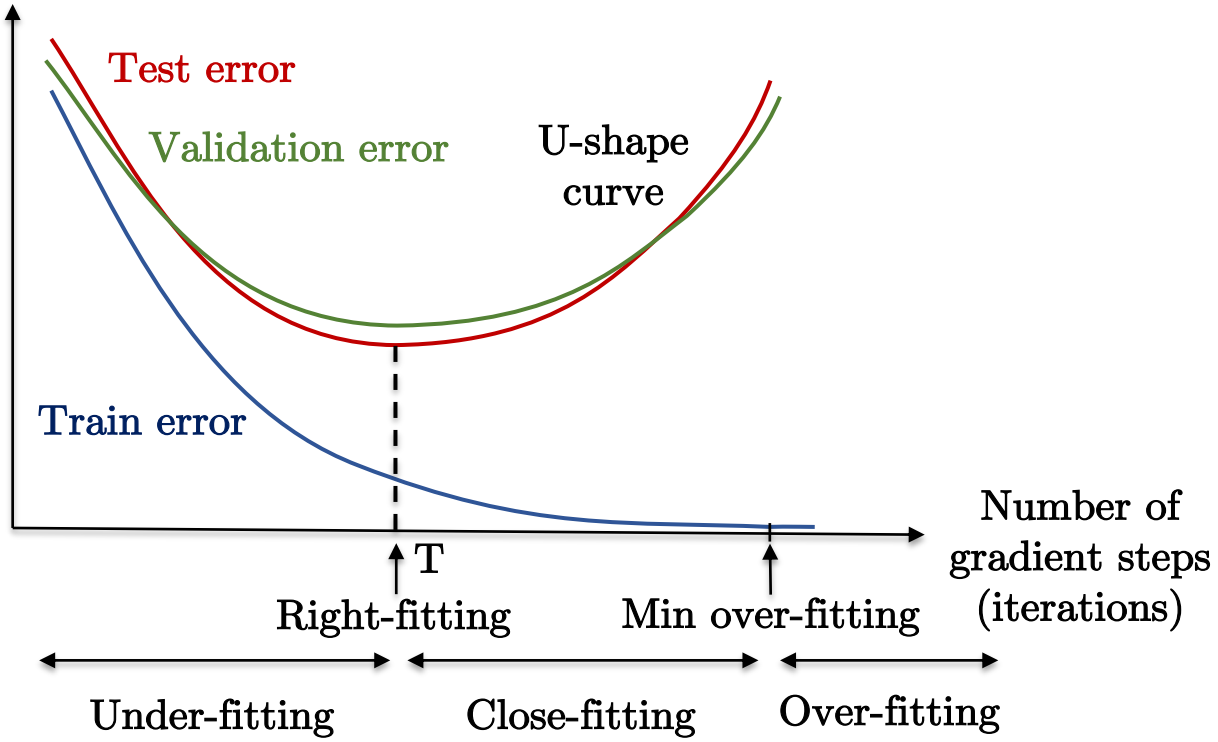
- Summary
  - Cross-validation is a sound technique, which works well in practice for different hypotheses and different sizes of training set.
  - The validation set is a surrogate of the test set but its capacity to represent well the test distribution depends on its size.
  - Best-case scenario : Training and validation sets are large enough.
  - Worst-case scenario : Either training or validation set is small.
    - Then k-fold cross-validation is required.
  - Even in the best-case scenario, it is still required to fully train the learner for each hypothesis, which can be time consuming, e.g. deep learning.
    - Can we develop a faster regularization technique to avoid over-fitting?

Outline

- Fitting the training set
- Reducing over-fitting
- Loss regularization
- Cross-validation
- Early stopping
- Double descent
- Conclusion

Early stopping

- The fastest regularization technique to avoid over-fitting.
- Stop optimization after T number of gradient steps, when the validation error starts increasing, even if optimization has not converged yet.
- Not really satisfying from an optimization theory perspective but it works well in practice.
- One of the most common regularization techniques in deep learning to control over-fitting.

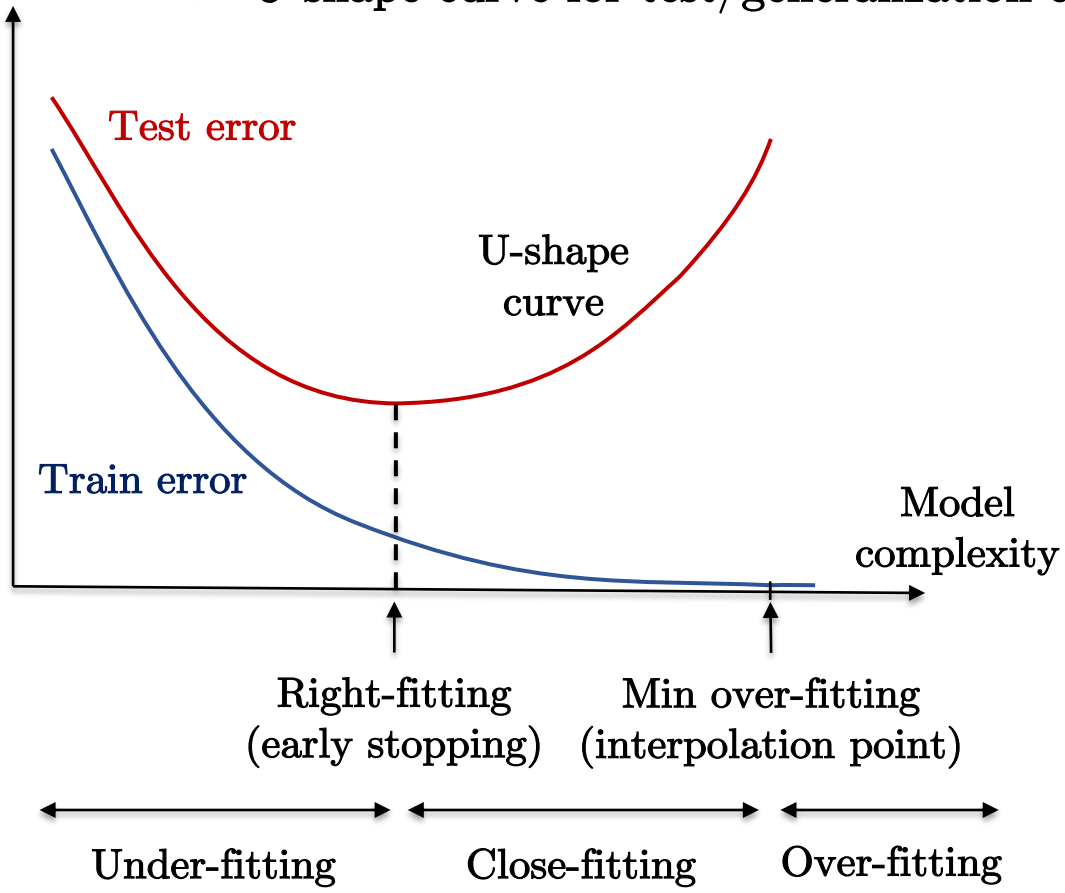


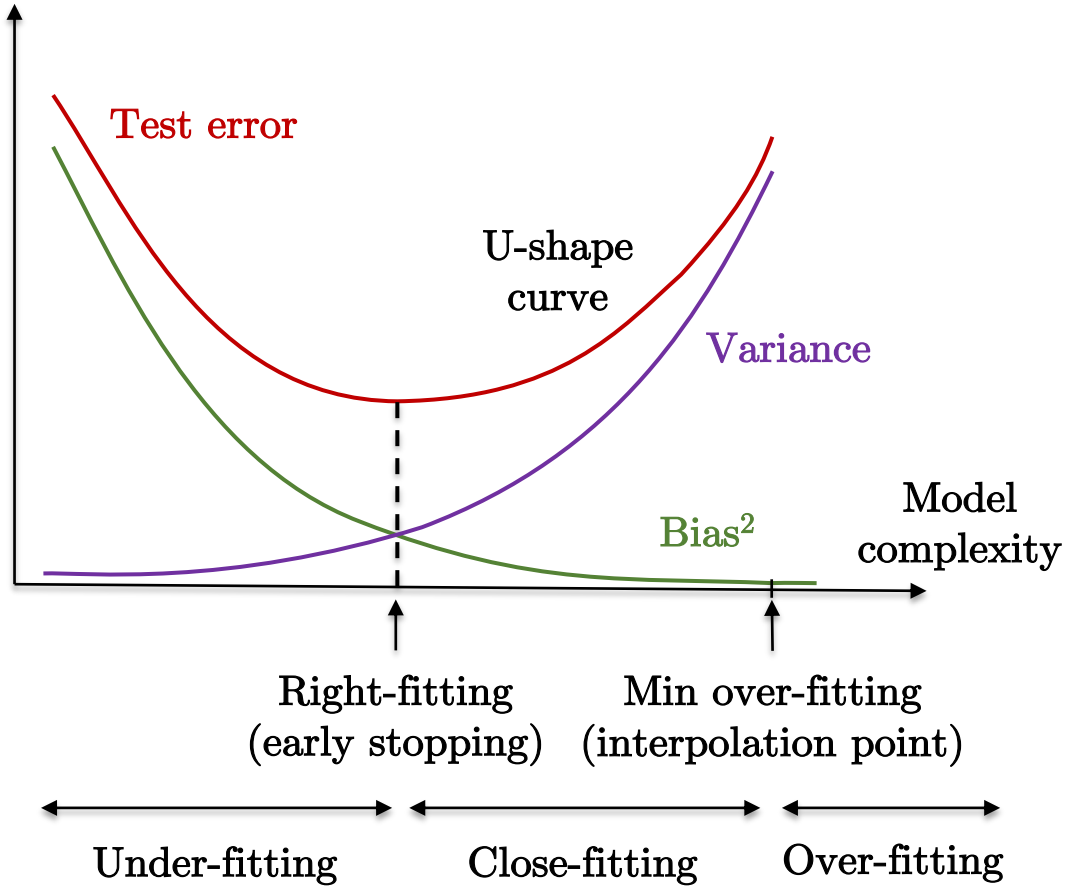
Outline

- Fitting the training set
- Reducing over-fitting
- Loss regularization
- Cross-validation
- Early stopping
- Double descent
- Conclusion

Double descent

- Classical machine learning (ML)
  - Bias-variance trade-off curve w.r.t. model complexity
  - U-shape curve for test/generalization error



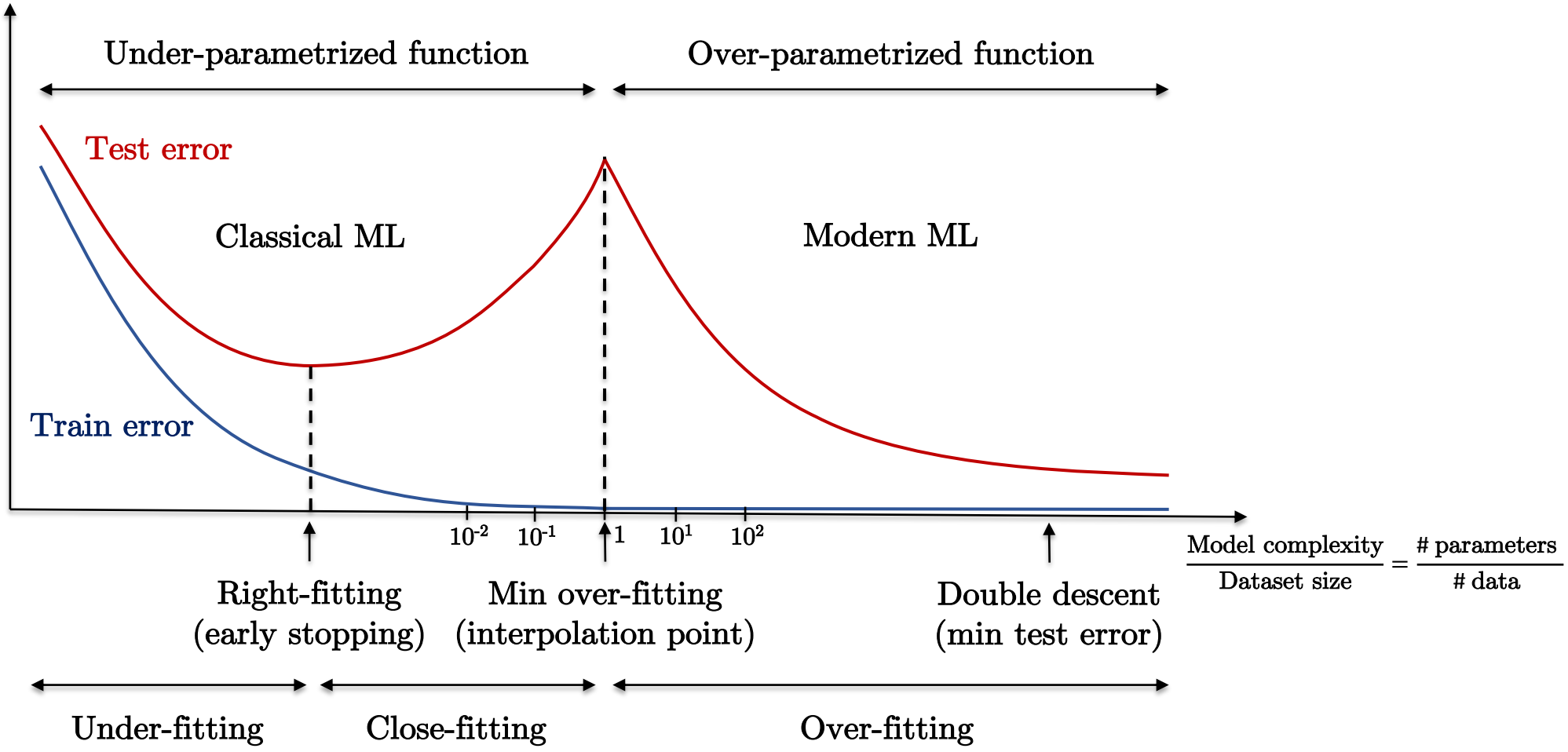


Double descent

- How to interpret the U-shape bias-variance trade-off curve?
  - In classical ML theory, when model complexity increases, variance and generalization error also increase.
- However, (deep learning) practitioners have observed an opposite phenomenon !
  - When model complexity increases, generalization error decreases 🙃
- This empirical result contradicts the conventional theory and shows a significant gap between theory and practice.
- To reconcile this inconsistency and better understand the properties of modern large ML models, a new learning mechanism known as “double descent” was introduced in 2018.

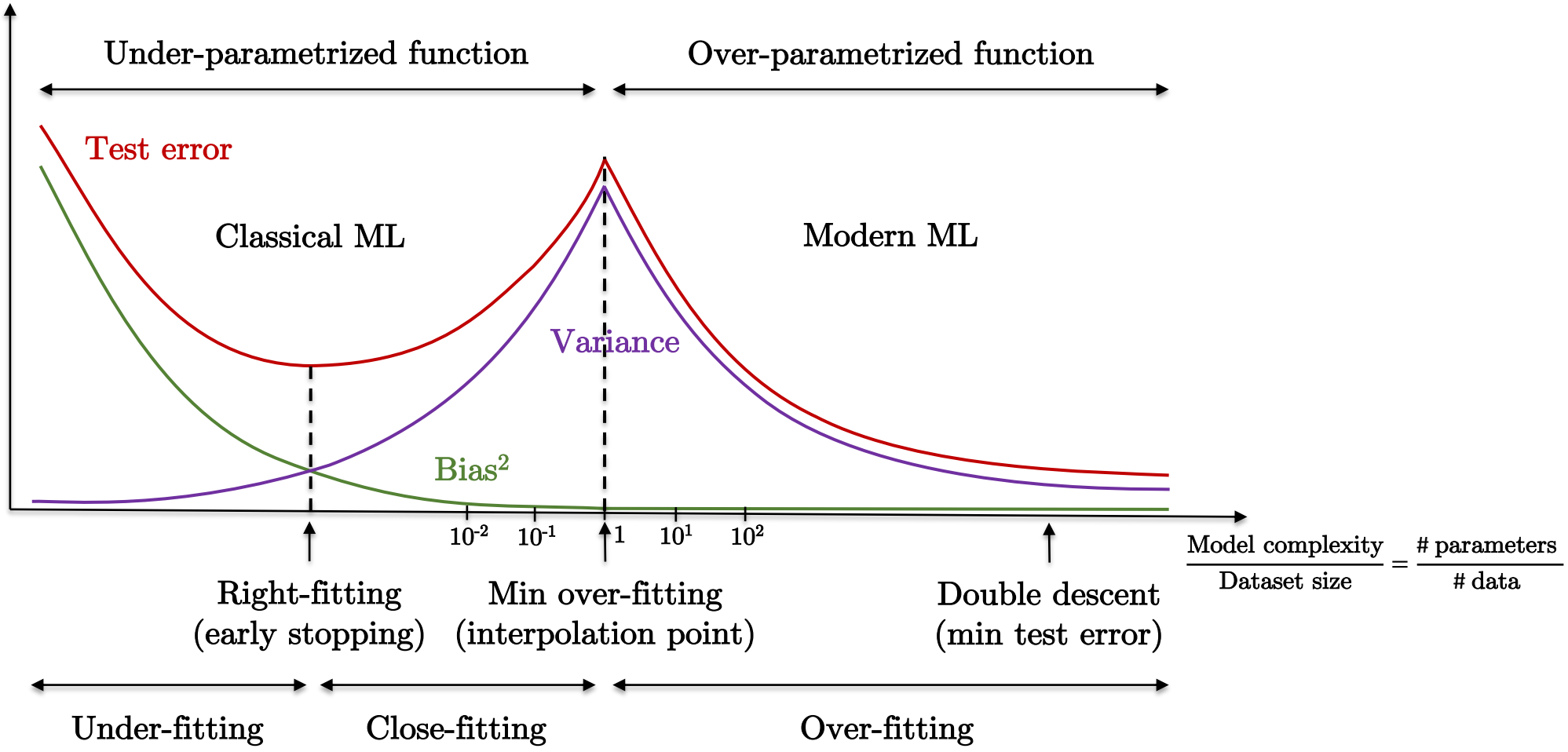
Double descent

- Double descent curve w.r.t. the ratio between the model complexity and the dataset size



Double descent

- Double descent curve for bias and variance

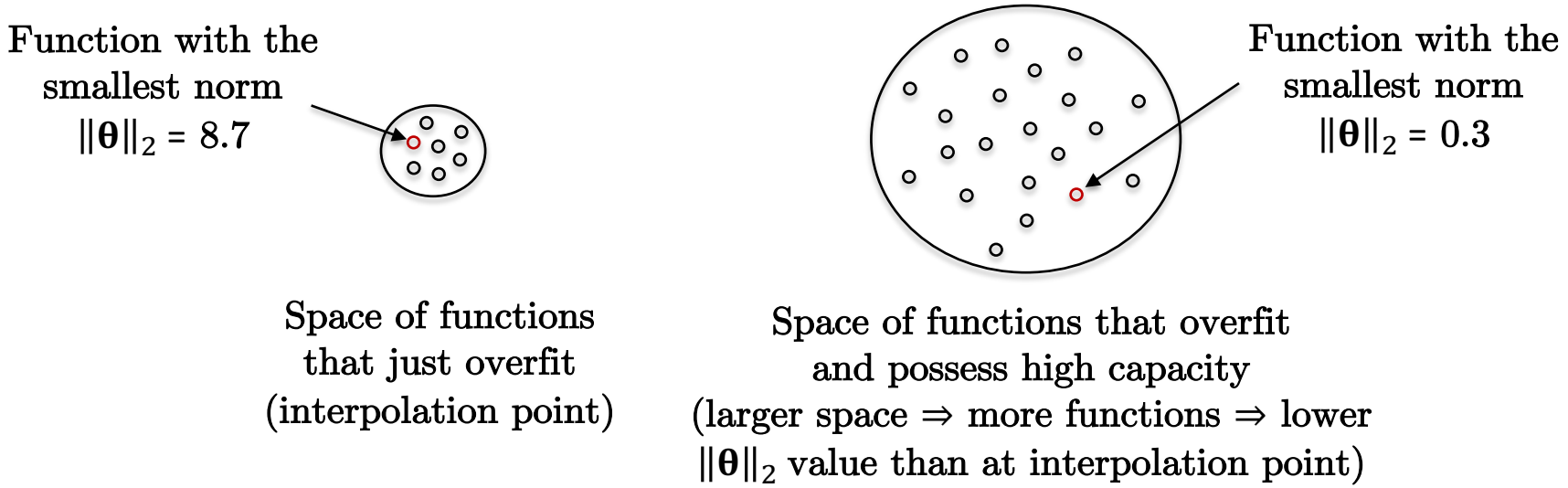


Double descent

- New learning curves
  - Let p be the number of parameters of the learner and n the number of training data points.
    - Under-parametrized functions are defined by $p \ll n$ (classical ML)
    - Over-parametrized functions are defined by $p \gg n$ (modern ML)
  - We also introduce the interpolation point, i.e. $p = n$, the minimal capacity needed to overfit the training set.
- In the classical ML paradigm, the optimal test error is at the minimum of the bias-variance trade-off and is captured in practice with early stopping using a validation set.
  - Classical ML establishes the existence of a right balance between under-fitting and over-fitting. Beyond this balance point i.e. over-fitting, generalization fails.
- In the modern ML regime, over-fitting is actually considered beneficial, and over-parametrized functions with high model complexity lead to successful generalization.

Double descent

- Understanding the second descent (new learning mechanism)
  - When p=n, the model possesses just enough parameters to over-fit all the training data. However, it also exhibits a significant variance, making it unable to generalize (standard result).
  - When p≫n, the model has much greater parameters than the number of training data. In this regime, the learner $f_\theta(x)$ continues to over-fit but critically, the L_2 norm of its parameters $\|\theta\|_2$ is significantly minimized by SGD, effectively reducing the model capacity (regularization effect).

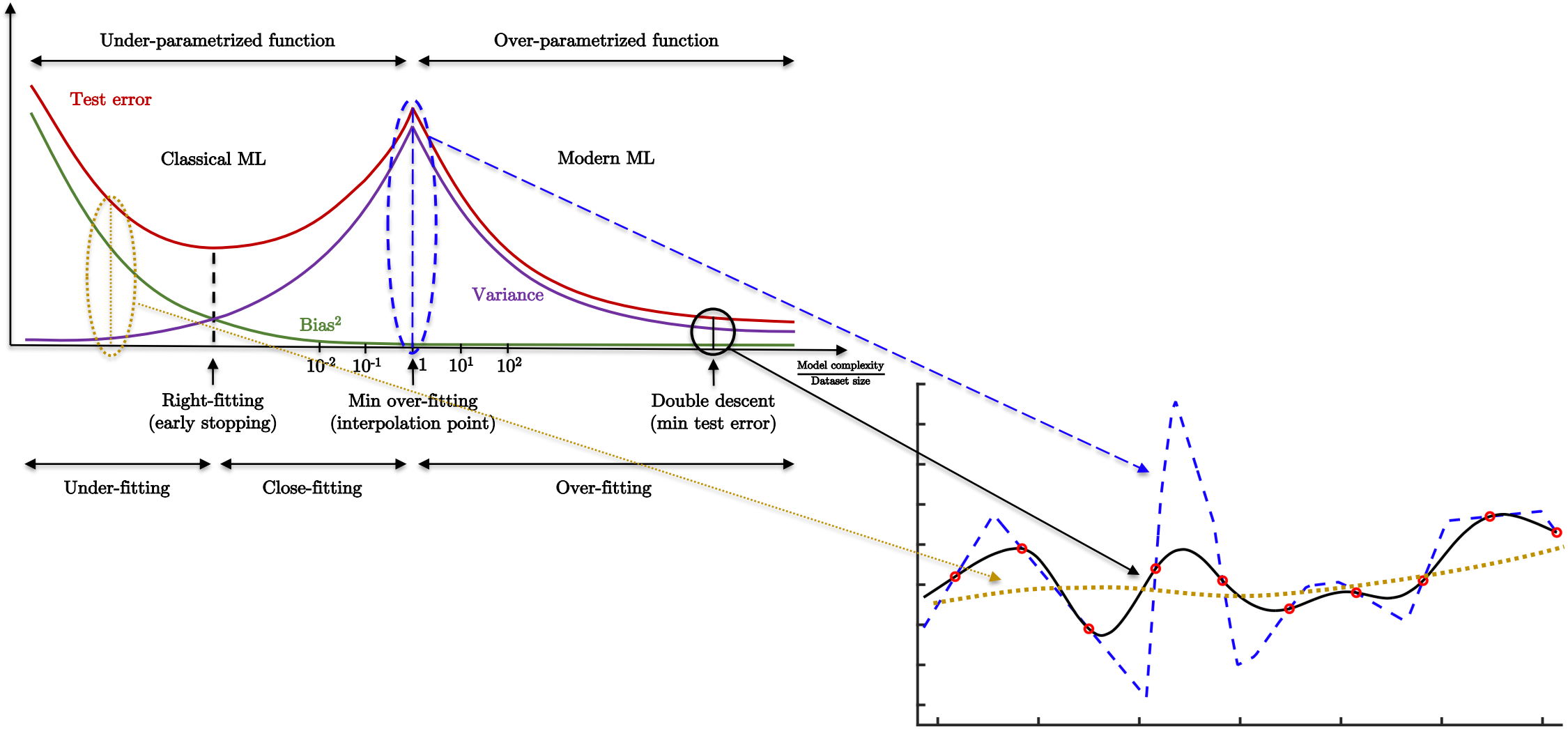


Double descent

- SGD is critical in deep learning for several reasons
  - It helps to leave saddle points in the loss landscape during optimization.
  - It finds better local (or even global) minima, allowing successful generalization.
  - It speeds up computational time (by updating the parameters more often).
  - It is necessary for the double descent phenomenon to emerge.

Double descent

- Illustration



Double descent

- Farewell to early stopping?
  - Do we "simply" need over-parametrized functions, train them, and achieve minimal error?
- Unfortunately, the double descent regularization only emerges with exceedingly large networks.
  - The critical threshold to enable double descent is $p^* = O(n.k)$, where k is the number of classes.
  - Computer Vision
    - ImageNet : $n = 10^6$ (1.3M images), $k = 10^3$ (1k classes) $\Rightarrow p^* = 10^9$
      ResNet-152 has $p = 60.2M$ (10^7) parameters $\ll 10^9$
    - ViT : $n = 10^9$ (4B images), $k = 10^4$ (30k classes) $\Rightarrow p^* = 10^{13}$
      ViT-22B has $p = 22B$ (10^{10}) parameters $\ll 10^{13}$
  - NLP : $n=10^{11}$ (300B token data), $k=10^4$ (35k unique tokens) $\Rightarrow p^* = 10^{15}$
    GPT-3 has $p = 175B$ (10^{11}) parameters $\ll 10^{15}$
- At present, practitioners use early stopping as their primary regularization technique.
  - By design, early stopping does not lead to the double descent phenomenon.

Double descent

- Some key observations
  - The double descent learning mechanism is applicable to both non-linear and linear ML models.
    - This includes techniques such as decision trees, kernel methods, and deep learning.
  - The phenomenon is independent of the nature of the datasets involved.
  - One important ML principle is that more data provides better results.
    - Both theory and empirical experiments align on this principle 🙂
    - However, this trend continually increases the critical threshold p* of required network parameters for the double descent phenomenon to manifest.

Outline

- Fitting the training set
- Reducing over-fitting
- Loss regularization
- Cross-validation
- Early stopping
- Double descent
- **Conclusion**

Conclusion

- Over-fitting and high variance are among the most common issues in machine learning.
- Regularization techniques
  - Stochastic gradient descent, the smaller the batch, the better but also slower.
  - Loss regularization and cross-validation to estimate the right amount of regularization.
  - Early stopping to terminate optimization before over-fitting.
  - Double descent with over-parametrized functions.

Questions?