ECEN 740:
Machine Learning Engineering

Lecture 4: Learning via Empirical Risk Minimization

Tie Liu

Texas A&M University

A direct approach to learn-to-predict

- In our previous lecture, we studied the *generative* approach to machine learning, which aims to first learn the joint distribution between the observation and the target and then bases its prediction on the learned distribution
- In this lecture we shall explore a more *direct* approach which aims to directly construct a predictor $h : \mathcal{X} \to \mathcal{Y}$ from the training data
- Our approach mainly utilizes a *loss function* $\ell : \mathcal{H} \times (\mathcal{X}, \mathcal{Y}) \to \mathbb{R}_+$ such that $\ell(h, (x, y))$ is the loss of the predictor h evaluated at the data example (x, y)
- For example, the canonical 0-1 loss for classification problems is given by

$$\ell_{0-1}(h, (x, y)) = 1_{\{h(x) \neq y\}}$$

and the canonical squared-error loss for regression problems is given by

$$\ell_{se}(h, (x, y)) = (h(x) - y)^2$$

Learning via empirical risk minimization (ERM)

- Our *overarching* goal is to find a predictor h that minimizes the Bayesian risk

$$L(h) = \mathbb{E}[\ell(h, (\mathsf{x}, \mathsf{y}))]$$

  where the expectation is over the joint distribution of (x, y). As mentioned previously, the optimal predictor is known as the *Bayes* predictor

- The Bayesian risk $L(h)$, however, needs to be calculated from the joint distribution of (x, y), which is *unknown* to the learner (except for some limited prior knowledge)

- A simple idea is to use the training data examples $((x_i, y_i) : i \in [m])$ to construct an *estimate* of the Bayesian risk $L(h)$ and then find the predictor that minimizes the *estimated* risk $\hat{L}(h)$. Under this context, the Bayesian risk is also known as the *true risk*

- A highly intuitive and easy-to-implement estimate of the true risk is the *empirical risk*

$$\hat{L}(h) = \frac{1}{m} \sum_{i=1}^{m} \ell(h, (x_i, y_i))$$

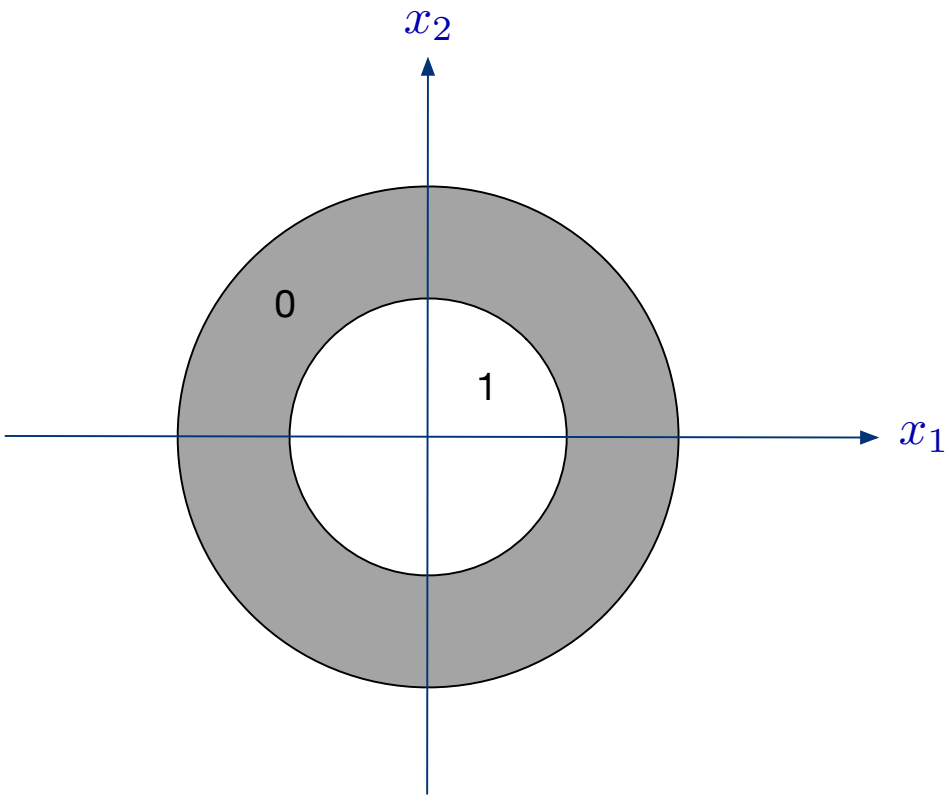
- One may then attempt to find a good predictor by minimizing the empirical risk; this is known as *empirical risk minimization (ERM)* and any predictor that minimizes the empirical risk is known as an *ERM predictor*:

$$h_{ERM} = \arg\min_{h} \left[\frac{1}{m} \sum_{i=1}^{m} \ell(h, (x_i, y_i)) \right]$$

- Note that while an ERM predictor renders the smallest *empirical* risk among all possible predictors, its performance in the real word should be measured by how it performs when tested by a *freshly* drawn data example, i.e., the *true* risk. Does an ERM predictor always render a small true risk as well?

- Somewhat surprisingly, the answer is *no* and next we shall present a rather "devastating" counter example

A counter example



- The observations are *uniformly* drawn from the big circle
- The shaded area and the small circle have the *same* area so the probabilities that an observation falls in either region is $1/2$
- The target $y = 1$ if the observation x falls in the small circle; otherwise, the target $y = 0$

- Note that for this problem, with probability 1, the observations for the training data examples will be *distinct*
- A very lazy predictor is thus given by

$$h(x) = \begin{cases} y_i, & \text{if } x = x_i \text{ for some } i \in [m] \\ 0, & \text{otherwise} \end{cases}$$

- The above predictor is, in fact, an *ERM* predictor under the canonical 0-1 loss because its *empirical* risk is always equal to 0
- What about the *true* risk?
- Note that with probability 1, a *freshly* drawn observation is *unseen*, so the aforementioned ERM predictor will always predict the target as 0
- On the other hand, the *true* target depends on the location of the observation and can take values of either 0 or 1 with equal probabilities. Therefore, the *true* risk of the aforementioned ERM predictor is $1/2$

- We have thus found an ERM predictor whose performance is always the *best* possible for the training data examples but always the *worst* possible for a freshly drawn data example
- Furthermore, this performance gap *cannot* be improved by increasing the size of the training data examples (as long as it is finite)
- What caused this huge gap between the empirical risk and the true risk?
- Instead of trying to learn the dependency of the target on the observation, the aforementioned ERM predictor simply tries to *remember* what happened with the training example while making *no* attempts to make good predictions on previously unseen observations
- We can all speak out of our own experience that such a *memorization* strategy usually does not work well in the real world

Over-fitting

- We say that *over-fitting* occurs whenever a learning algorithm outputs a *learner* whose performance on the training examples is excellent, but its performance on a freshly drawn data example is very poor
- Note that learning is based on training data examples that are *randomly* drawn from the data-generating distribution. As long as the number of training data examples is *finite*, they can only provide a *partial, random* view to the data-generating distribution
- Over-fitting occurs whenever the construction of a learner cages *too much* to the pattern of the training data examples and the prediction rule does *not* reflect on the true data-generating distribution
- The aforementioned "memorization" strategy is an extreme example of "caging to the training examples". While caging can lead to excellent performance on the training data, such a performance may not translate to freshly drawn data examples

Predictor modeling

- A simple approach for avoiding over-fitting is to *pre-select* a class of predictors in our search of the ERM solution:

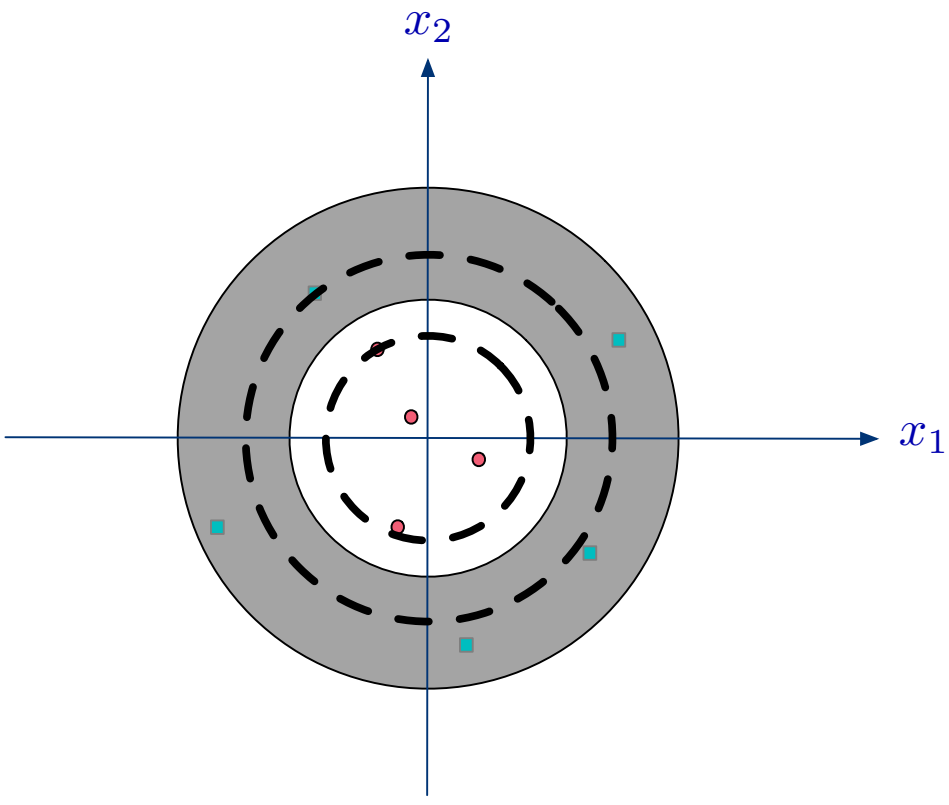
$$h_{ERM} = \arg\min_{h \in \mathcal{H}} \left[\frac{1}{m} \sum_{i=1}^{m} \ell(h, (x_i, y_i)) \right]$$

  where $\mathcal{H}$ is a pre-selected *hypothesis class*

- The use of a properly selected hypothesis class can *prevent* an ERM solution from caging too much to the training data examples and ensure that the performance an ERM predictor on the training data examples can *translate* to freshly drawn, unseen examples

- Next, let us revisit the previous counter example with a carefully selected hypothesis class

A counter example (Cont'd)



- Consider the hypothesis class $\mathcal{H} = \{h_a : a \geq 0\}$ where

$$h_a(x) = 1_{\{\|x\| \leq a\}}$$

- For a given set of training examples, *any* predictor h_a whose decision boundary is in between the two dashed circles is an ERM solution with *zero* empirical risk

- The true risk, however, is nonzero as long as the decision boundary deviates from the small circle

- Note that as the number of training examples increases, the decision boundary of the ERM solution will "converge" to the small circle. As as result, the true risk of an ERM solution will approach zero

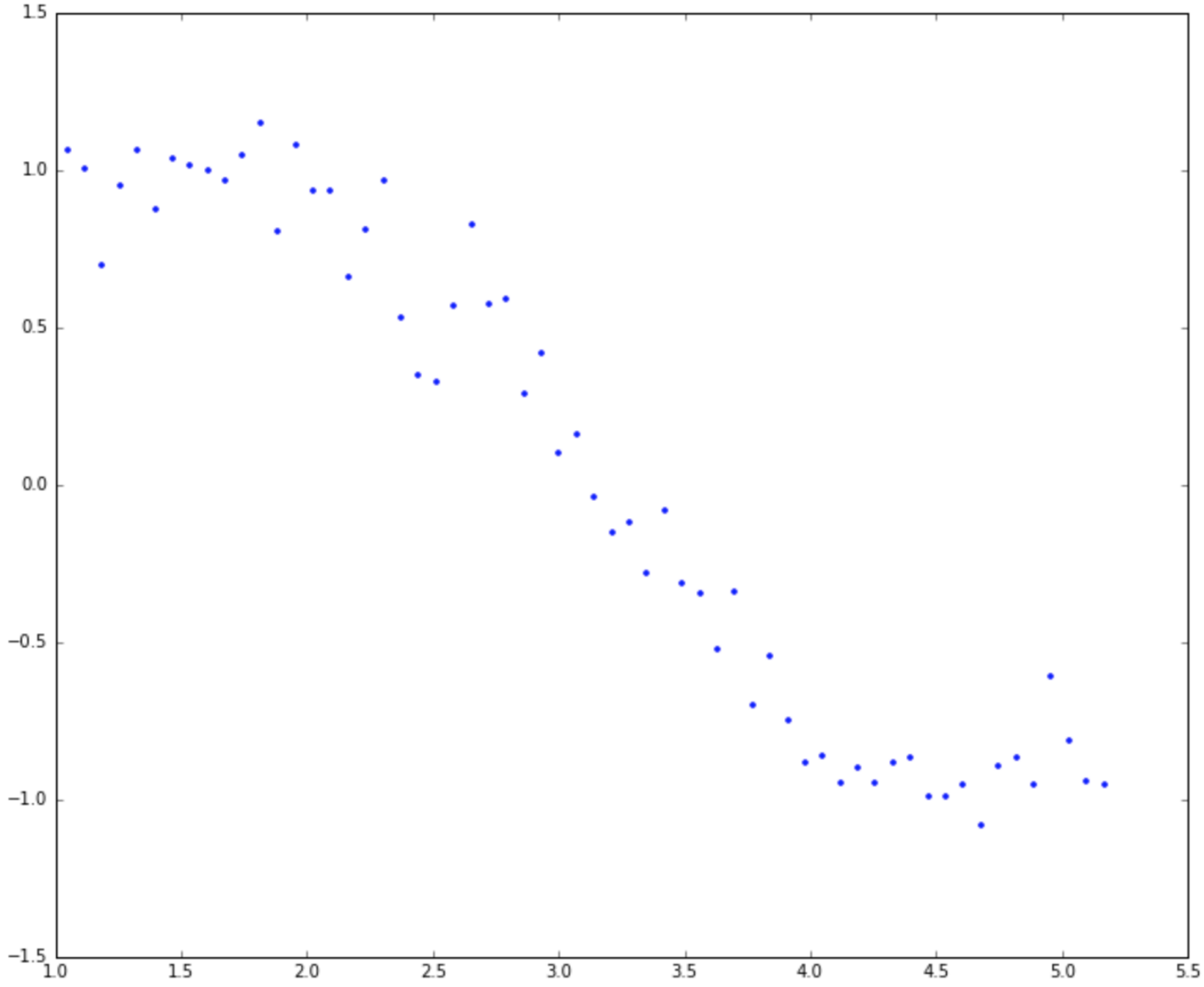
- This is in sharp contrast to the the case where there is no restriction on the search of an ERM solution

From over-fitting to generalization error

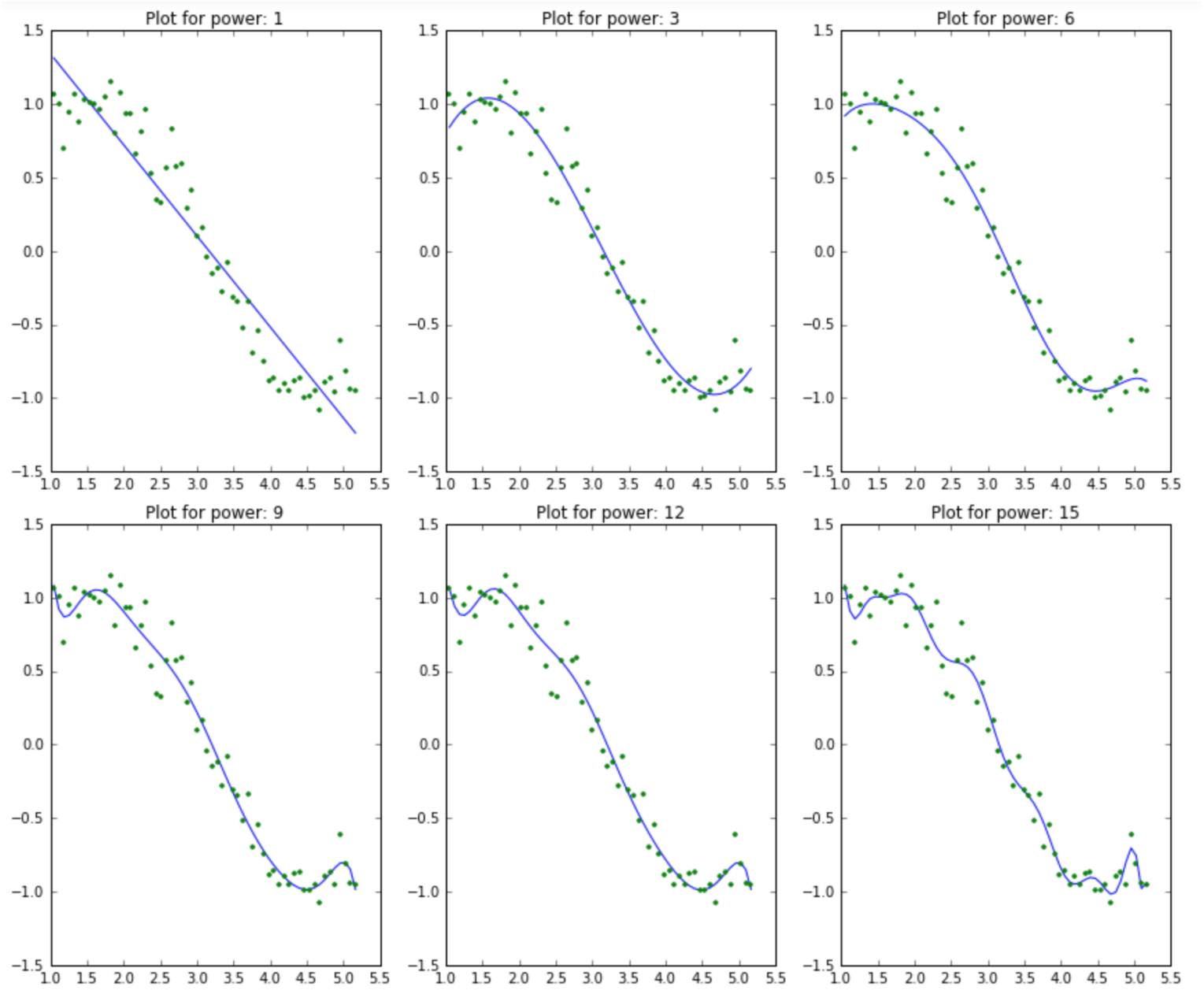
- For a given learner, we define the *excess* of the true risk over the empirical risk as *generalization error*
- The concept of generalization error thus leads to the following *decomposition* of the true risk:

$$\text{True risk} = \text{Empirical risk} + \text{Generalization error}$$

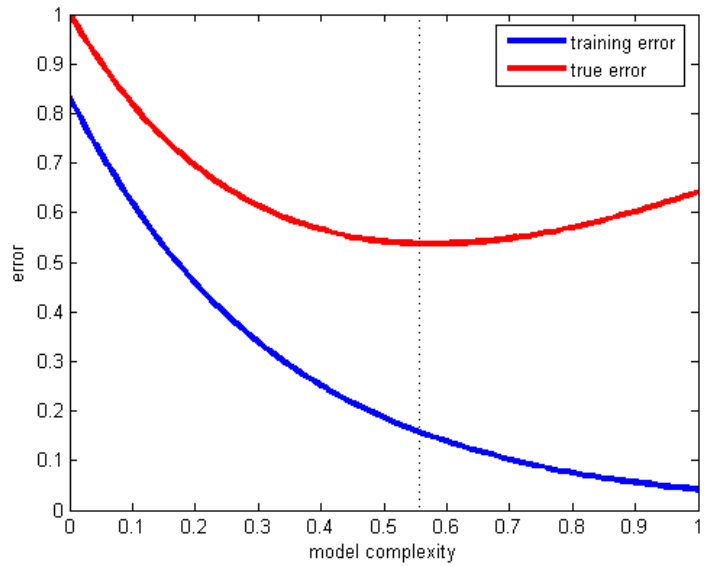
- The choice of the hypothesis class $\mathcal{H}$ can have a *significant* impact on both empirical risk and generalization error
- Next let us take a look at a regression problem where the data examples are generated from a sinusoidal curve (between 60 and 300 degrees) plus some random noise


1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5

Polynomial regression under squared-error loss



- The figures from the previous slide show the ERM solutions from the hypothesis classes $\mathcal{H} = \mathcal{P}_k$ for $k = 1, 3, 6, 9, 12, 15$, where $\mathcal{P}_k$ is the collection of all polynomials whose degrees are less than or equal to k
- Note that on one hand, the empirical risk decreases as the hypothesis class becomes larger (k increases). As a result, for small values of k, the true risk also decreases as k increases
- On the other hand, as k continues to increase, the ERM solution starts to cave to the random noise, and as a result, the true risk starts to increase. This is the classical situation of *over-fitting*



Hyper-parameter tuning and cross validation

- In practice, instead of zooming in on one particular hypothesis class, we usually train *multiple* learners over a family of hypothesis classes indexed by a *hyper-parameter*
- To choose the best learner, we can split the training data into two parts: one for *training* and the other for *hyper-parameter tuning*. In literature, this is known as *cross validation*
- While cross validation is very helpful in terms of "validating" the real-world performance of the learner, it also takes *away* valuable data examples that can be potentially used for training
- Estimating the true risk of a learner *without* splitting the training data is a much more challenging (and fun) task

Regularized loss minimization (RLM)

- In addition to using a pre-selected hypothesis class to restrict our search of an ERM solution, another approach for avoiding over-fitting is to introduce an extra *regularization* term in the search of an ERM solution:

$$h_{RLM} = \arg\min_{h \in \mathcal{H}} \left[\frac{1}{m} \sum_{i=1}^{m} \ell(h, (x_i, y_i)) + \lambda \cdot R(h) \right]$$

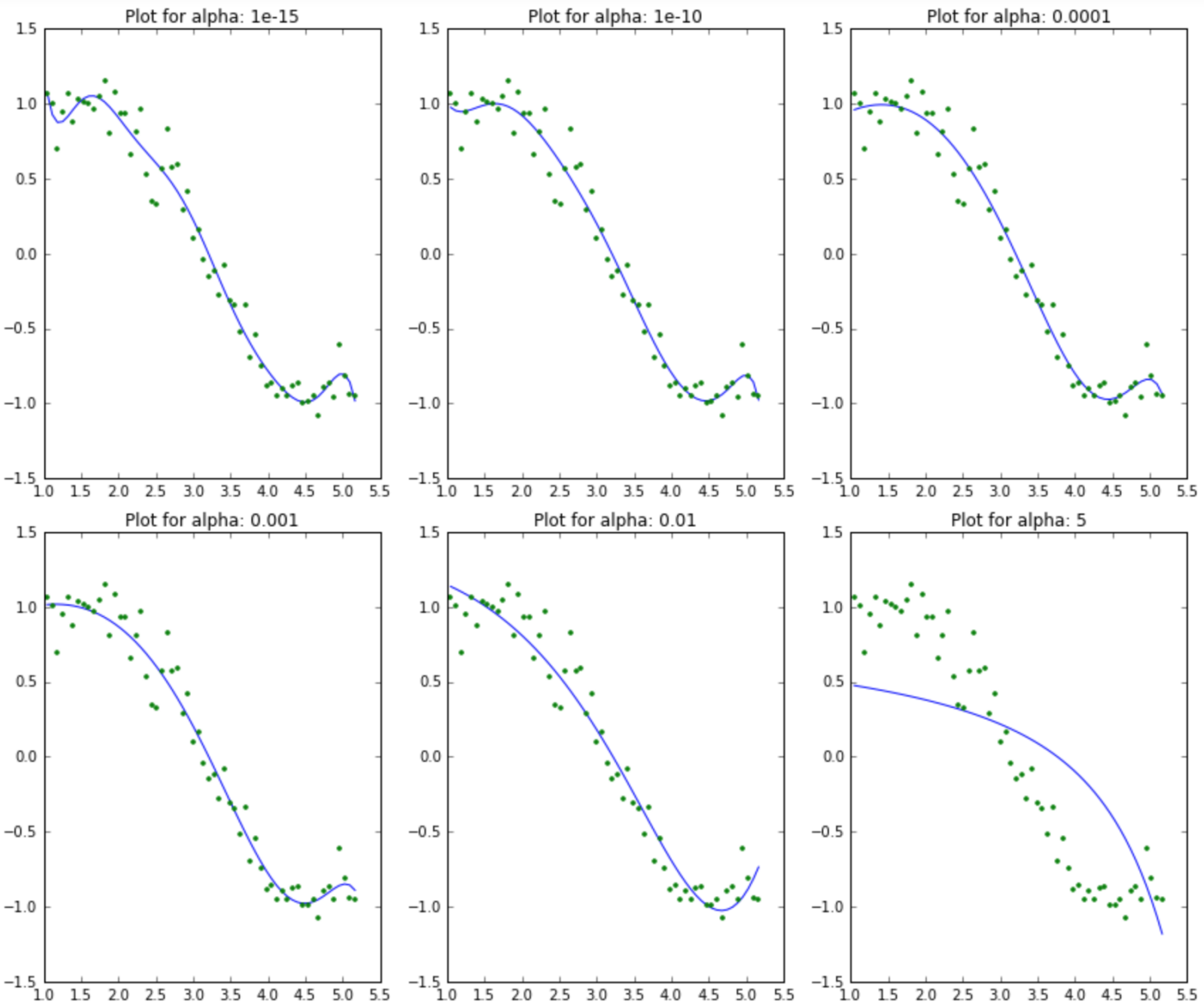
  where $R : \mathcal{H} \to \mathbb{R}_+$ is a *regularizer* that models our *predisposition* on the hypotheses, and λ is a non-negative *hyper-parameter* that controls the balance between empirical risk and our belief. In literature, this is known as *regularized loss minimization (RLM)*

- Similar to the hypothesis class $\mathcal{H}$, the choice of the regularizer is based on our prior knowledge on the data-generating distribution
- The hyper-parameter λ can be tuned via *cross-validation*
- But how to choose an appropriate regularizer?

Polynomial regression (Cont'd)

	rss	intercept	coef_x_1	coef_x_2	coef_x_3	coef_x_4	coef_x_5	coef_x_6	coef_x_7	coef_x_8	coef_x_9
model_pow_1	3.3	2	-0.62	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
model_pow_2	3.3	1.9	-0.58	-0.006	NaN	NaN	NaN	NaN	NaN	NaN	NaN
model_pow_3	1.1	-1.1	3	-1.3	0.14	NaN	NaN	NaN	NaN	NaN	NaN
model_pow_4	1.1	-0.27	1.7	-0.53	-0.036	0.014	NaN	NaN	NaN	NaN	NaN
model_pow_5	1	3	-5.1	4.7	-1.9	0.33	-0.021	NaN	NaN	NaN	NaN
model_pow_6	0.99	-2.8	9.5	-9.7	5.2	-1.6	0.23	-0.014	NaN	NaN	NaN
model_pow_7	0.93	19	-56	69	-45	17	-3.5	0.4	-0.019	NaN	NaN
model_pow_8	0.92	43	-1.4e+02	1.8e+02	-1.3e+02	58	-15	2.4	-0.21	0.0077	NaN
model_pow_9	0.87	1.7e+02	-6.1e+02	9.6e+02	-8.5e+02	4.6e+02	-1.6e+02	37	-5.2	0.42	-0.015
model_pow_10	0.87	1.4e+02	-4.9e+02	7.3e+02	-6e+02	2.9e+02	-87	15	-0.81	-0.14	0.026
model_pow_11	0.87	-75	5.1e+02	-1.3e+03	1.9e+03	-1.6e+03	9.1e+02	-3.5e+02	91	-16	1.8
model_pow_12	0.87	-3.4e+02	1.9e+03	-4.4e+03	6e+03	-5.2e+03	3.1e+03	-1.3e+03	3.8e+02	-80	12
model_pow_13	0.86	3.2e+03	-1.8e+04	4.5e+04	-6.7e+04	6.6e+04	-4.6e+04	2.3e+04	-8.5e+03	2.3e+03	-4.5e+02
model_pow_14	0.79	2.4e+04	-1.4e+05	3.8e+05	-6.1e+05	6.6e+05	-5e+05	2.8e+05	-1.2e+05	3.7e+04	-8.5e+03
model_pow_15	0.7	-3.6e+04	2.4e+05	-7.5e+05	1.4e+06	-1.7e+06	1.5e+06	-1e+06	5e+05	-1.9e+05	5.4e+04

Regularized polynomial regression ($k = 15$)



	rss	intercept	coef_x_1	coef_x_2	coef_x_3	coef_x_4	coef_x_5	coef_x_6	coef_x_7	coef_x_8	coef_x_9	coef_x_10	coef_x_11
alpha_1e-15	0.87	95	-3e+02	3.8e+02	-2.4e+02	66	0.96	-4.8	0.64	0.15	-0.026	-0.0054	0.00086
alpha_1e-10	0.92	11	-29	31	-15	2.9	0.17	-0.091	-0.011	0.002	0.00064	2.4e-05	-2e-05
alpha_1e-08	0.95	1.3	-1.5	1.7	-0.68	0.039	0.016	0.00016	-0.00036	-5.4e-05	-2.9e-07	1.1e-06	1.9e-07
alpha_0.0001	0.96	0.56	0.55	-0.13	-0.026	-0.0028	-0.00011	4.1e-05	1.5e-05	3.7e-06	7.4e-07	1.3e-07	1.9e-08
alpha_0.001	1	0.82	0.31	-0.087	-0.02	-0.0028	-0.00022	1.8e-05	1.2e-05	3.4e-06	7.3e-07	1.3e-07	1.9e-08
alpha_0.01	1.4	1.3	-0.088	-0.052	-0.01	-0.0014	-0.00013	7.2e-07	4.1e-06	1.3e-06	3e-07	5.6e-08	9e-09
alpha_1	5.6	0.97	-0.14	-0.019	-0.003	-0.00047	-7e-05	-9.9e-06	-1.3e-06	-1.4e-07	-9.3e-09	1.3e-09	7.8e-10
alpha_5	14	0.55	-0.059	-0.0085	-0.0014	-0.00024	-4.1e-05	-6.9e-06	-1.1e-06	-1.9e-07	-3.1e-08	-5.1e-09	-8.2e-10
alpha_10	18	0.4	-0.037	-0.0055	-0.00095	-0.00017	-3e-05	-5.2e-06	-9.2e-07	-1.6e-07	-2.9e-08	-5.1e-09	-9.1e-10
alpha_20	23	0.28	-0.022	-0.0034	-0.0006	-0.00011	-2e-05	-3.6e-06	-6.6e-07	-1.2e-07	-2.2e-08	-4e-09	-7.5e-10

Norm regularization

- In literature, penalizing the sum of the *square* of the coefficients is known as *ridge regularization* or *Tikhonov regularization*
- Ridge regression is closely related to two important ideas in machine learning: *support vector machine (SVM)* and the *kernel method*, which we will study next in our lecture
- Alternatively, one may choose to penalize the sum of the *absolute value* of the coefficients. This is known as *Lasso regularization*
- The key difference between these techniques is that Lasso shrinks the less important feature's coefficient to *zero*, thus removing some features altogether. This works well for *feature selection* in case we have a huge number of features

Summary

- Learning via empirical risk minimization (ERM) is widely considered as the *foundational* direct discriminative learning approach
- Learning via ERM is subject to the potential risk of *over-fitting*. To avoid over-fitting, it is necessary to use a pre-selected hypothesis class to restrict our search of an ERM solution
- The choice of the hypothesis class needs to balance between *fitting* (measured by empirical risk) and *generalization* (measured by generalization error)
- The *generalization* of an ERM solution can be further improved by introducing a *regularization* term to the objective function (at the expense of slightly increasing the empirical risk)
- Understanding generalization is at the heart of learning theory. For machine learning practitioners, it is important to always keep the notion of generalization on mind for algorithm design and analysis