Statistic

for machine learning

Tran Trong Khiem

AI lab tranning

2024/01/15

Definition

We define the MLE as follows:

$$\hat{\theta}_{\text{mle}} = \arg\max_{\theta} p(D|\theta)$$

Where D is dataset, θ is parameter of the model. Assume dataset are iid(independently sampled from the same distribution).We have :

$$p(D|\theta) = \prod_{n=1}^{N} p(y_n|x_n, \theta)$$

We have log likelihood as define:

$$\ell(\theta) = logp(D|\theta) = \sum_{n=1}^{N} \log p(y_n|x_n, \theta)$$

Thus the MLE is given by:

$$\hat{\theta}_{\text{mle}} = \arg\max_{\theta} \sum_{n=1}^{N} \log p(y_n|x_n, \theta)$$

Negative log likelihood (NLL)

Nagative log likelihood is defines as :

$$\text{NLL}(\theta) = -\log p(D|\theta) = -\sum_{n=1}^{N} \log p(y_n|x_n, \theta)$$

Minimizing this will give the MLE. If the model is unconditional (unsupervised), the MLE becomes:

$$\hat{\theta}_{\text{mle}} = \arg\min_{\theta} -\sum_{n=1}^{N} \log p(y_n|\theta)$$

Since we have output y_n but not have input x_n. We may want to maximize the joint likelihood of inputs and outputs. The MLE in this case becomes:

$$\hat{\theta}_{\text{mle}} = \arg\min_{\theta} -\sum_{n=1}^{N} \log p(y_n|x_n, \theta)$$

Justification for MLE(Why using MLE ?)

•1.Simple piont approximation to the Bayesian posterior $p(\theta|D)$ using a uniform prior.
we approximate the posterior by a delta function $p(\theta|D) = \delta(\theta - \hat{\theta}_{\text{map}})$.
Where $\hat{\theta}_{\text{map}}$ is the posterior mode, given by :

$$\hat{\theta}_{\text{map}} = \arg\max_{\theta} \log p(\theta|D) = \arg\max_{\theta} (\log p(D|\theta) + \log p(\theta))$$

Assume $p(\theta)$ is uniform disttribution, this equation become MLE and $\hat{\theta}_{\text{MAP}} = \hat{\theta}_{\text{MLE}}$

•2.The resulting predictive distribution $p(y|\hat{\theta}_{\text{mle}})$ is as close as possible to the empirical distribution of the data.
In the unconditional case, the empirical distribution is defined by:

$$p_D(y) = \frac{1}{N} \sum_{n} \delta(y - y_n)$$

We want to create a model whose distribution $p(y) = p(y|\theta)$ is similar to $p(y)_D$

Justification for MLE(Why using MLE ?)

•2.The resulting predictive distribution $p(y|\hat{\theta}_{\text{mle}})$ is as close as possible to the empirical distribution of the data.

A standard way to measure the (dis)similarity between probability distributions p and q is the **Kullback Leibler divergence**, or KL divergence.Given by :

$$D_{KL}(p \parallel q) = \sum_y p(y) \log \left(\frac{p(y)}{q(y)} \right)$$

Unsupervised case: Define $q(y) = p(y|\theta)$ and $p(y) = p_D(y)$.KL become :

$$\begin{aligned} D_{KL}(p \parallel q) &= \sum_y [p_D(y) \log p_D(y) - p_D(y) \log q(y)] \\ &= -H(p_D) - \frac{1}{N} \sum_{n=1}^{N} \log p(y_n|\theta) \\ &= \text{const} + \text{NLL}(\theta) \end{aligned} \tag{1}$$

Where $H(p_D)$ is entropy of p_D.

Justification for MLE(Why using MLE ?)

- **2.The resulting predictive distribution $p(y|\hat{\theta}_{\text{mle}})$ is as close as possible to the empirical distribution of the data.**

Supervised case: use the following emperical distribution :

$$p_D(x,y) = p_D(y|x)p_D(x) = \frac{1}{N}\sum_{n=1}^{N}\delta(x-x_n)\delta(y-y_n) \tag{2}$$

The expected KL then becomes:

$$\begin{aligned}\mathbb{E}_{p_D(x)}\left[D_{KL}\left(p_D(Y|x) \parallel q(Y|x)\right)\right] &= \sum_x p_D(x) \\ &= \text{const} - \sum_x\sum_y p_D(x,y)\log\frac{q(y|x)}{p_D(x,y)} \\ &= \text{const} - \frac{1}{N}\sum_{n=1}^{N}\log p(y_n|x_n,\theta) \qquad (3)\end{aligned}$$

Minimizing this is equivalent to minimizing the conditional NLL

MLE for the Bernoulli distribution

Y is a random variable representing a coin toss, where the event Y = 1 corresponds to heads, Y = 0 corresponds to tails.Let $\theta = p(Y = 1)$.The probability distribution for this rv is the Bernoulli.

$$\begin{aligned}
\text{NLL}(\theta) &= -\log \prod_{n=1}^{N} p(y_n|\theta) \\
&= -\sum_{n=1}^{N} \log p(y_n|\theta) \\
&= -\sum_{n=1}^{N} \left[\sum_{y_n=1} p(y_n = 1|\theta) \log \theta + \sum_{y_n=0} p(y_n = 0|\theta) \log(1-\theta) \right] \\
&= -[N_1 \log \theta + N_0 \log(1-\theta)] \qquad (4)
\end{aligned}$$

Where N_1 is number times of head.
N_0 is number times of tail.
We have :

$$\hat{\theta}_{\text{MLE}} = \frac{N_1}{N_0 + N_1}$$

MLE for the categorical distribution

We roll a K-sided dice N times. Let $Y_n \in \{1, \ldots, K\}$, where $Y_n \sim \text{Cat}(\theta)$.We want to estimate the probabilities θ from data set $D = \{y_n : n = 1 : N\}$.We have NLL:

$$\text{NLL}(\theta) = -\sum_k N_k \log \theta_k$$

Where N_k is number times roll k-sided.Using $\sum_k \theta_k = 1$. We have MLE is given by:

$$\hat{\theta}_{mle} = \frac{N_k}{N}$$

MLE for the univariate Gaussian

Suppose $Y \sim \mathcal{N}(\mu, \sigma^2)$ and let $D = \{y_n : n = 1, \ldots, N\}$ be an iid. sample of size N.We can estimate the parameters $\theta = (\mu, \sigma^2)$ using MLE. NLL is given by :

$$\begin{aligned} \text{NLL}(\mu, \sigma^2) &= -\sum_{n=1}^{N} \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2} \frac{(y_n - \mu)^2}{\sigma^2} \right) \right) \\ &= \frac{N}{2} \log(2\pi\sigma^2) + \frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - \mu)^2 \end{aligned} \tag{5}$$

Minimize NLL we have :

$$\hat{\mu}_{\text{MLE}} = \frac{1}{N} \sum_{n=1}^{N} y_n = \bar{y} \tag{6}$$

$$\begin{aligned} \hat{\sigma}^2_{\text{MLE}} &= \frac{1}{N} \sum_{n=1}^{N} (y_n - \hat{\mu}_{\text{MLE}})^2 \\ &= \frac{1}{N} \sum_{n=1}^{N} y_n^2 - \hat{\mu}^2_{\text{MLE}} \end{aligned}$$

MLE for the multivariate Gaussian

Write the log-likelihood, dropping irrelevant constant:

$$\log p(D|\mu, \Sigma) = \frac{N}{2} \log |\Lambda| - \frac{1}{2} \sum_{n=1}^{N} (y_n - \mu)^T \Lambda (y_n - \mu) \tag{7}$$

Where $\Lambda = \Sigma^{-1}$ is the precision matrix (inverse covariance matrix).

- **MLE for the mean**: Using the substitution $z_n = y_n - \mu$, we have :

$$\frac{\partial}{\partial \mu} \left((y_n - \mu)^T \Sigma^{-1} (y_n - \mu) \right) = \frac{\partial z_n^T \Sigma^{-1} z_n}{\partial \mu}$$

$$= -(\Sigma^{-1} + \Sigma^{-T}) z_n$$

Since $\frac{\partial z_n}{\partial \mu^T} = -I$.Hence:

$$\hat{\mu} = \frac{1}{X} \sum y_n = \bar{y}$$

So the MLE of μ is just the empirical mean.

MLE for the multivariate Gaussian

• **MLE for the covariance matrix:** use trace trick to rewrite NLL ,we have:

$$l(\hat{\mu}, \Lambda) = \frac{1}{2}\sum_n \log|\Lambda| - \frac{1}{2}\text{tr}\left[(y_n - \hat{\mu})(y_n - \hat{\mu})^T\Lambda\right]$$

$$= \frac{N}{2}\log|\Lambda| - \frac{1}{2}\text{tr}\left[S_{\bar{y}}\Lambda\right]$$

$$S_{\bar{y}} = \sum_n (y_n - \bar{y})(y_n - \bar{y})^T$$

$$= (\sum_n y_n y_n^T) - N\bar{y}\bar{y}^T$$

we can compute derivatives of the loss with respect to Λ:

$$\hat{\Sigma} = \frac{1}{N}\sum_n (y_n - \bar{y})(y_n - \bar{y})^T$$

MLE for linear regression

Linear regression as following:

$$p(y|x;\theta) = \mathcal{N}(y|w^T x, \sigma^2)$$

where $\theta = (w, \sigma^2)$.Assume that σ^2 is fixed, focus on on estimating the weights w. NLL is given by:

$$\text{NLL}(w) = \frac{1}{2\sigma^2}\sum_{n=1}^{N}(y_n - w^T x_n)^2 + \frac{N}{2}\log(2\pi\sigma^2)$$

Dropping the irrelevant additive constants, we have residual sum of squares or RSS:

$$\text{RSS}(w) = \sum_{n=1}^{N}(y_n - w^T x_n)^2 = \sum_{n=1}^{N} r_n^2$$

where r_n represents the n-th residual error. $\nabla_w \text{RSS}(w) = 0$ satisfies the following equation:

$$\hat{w}_{\text{mle}} = \arg\min \text{RSS}(w) = (X^T X)^{-1} X^T y$$

Example: minimizing the misclassification rate

If we are solving classification problem, we may want to use 0-1 loss:

$$\mathcal{L}(y_n, f(x_n;\theta)) = \begin{cases} 0 & \text{if } y_n = f(x_n;\theta) \\ 1 & \text{if } y_n \neq f(x_n;\theta) \end{cases}$$

where $f(x_n;\theta)$ is some kind of predictior. The empirical risk becomes:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{n=1}^{N} l_{01}(y_n, \theta; x_n)$$

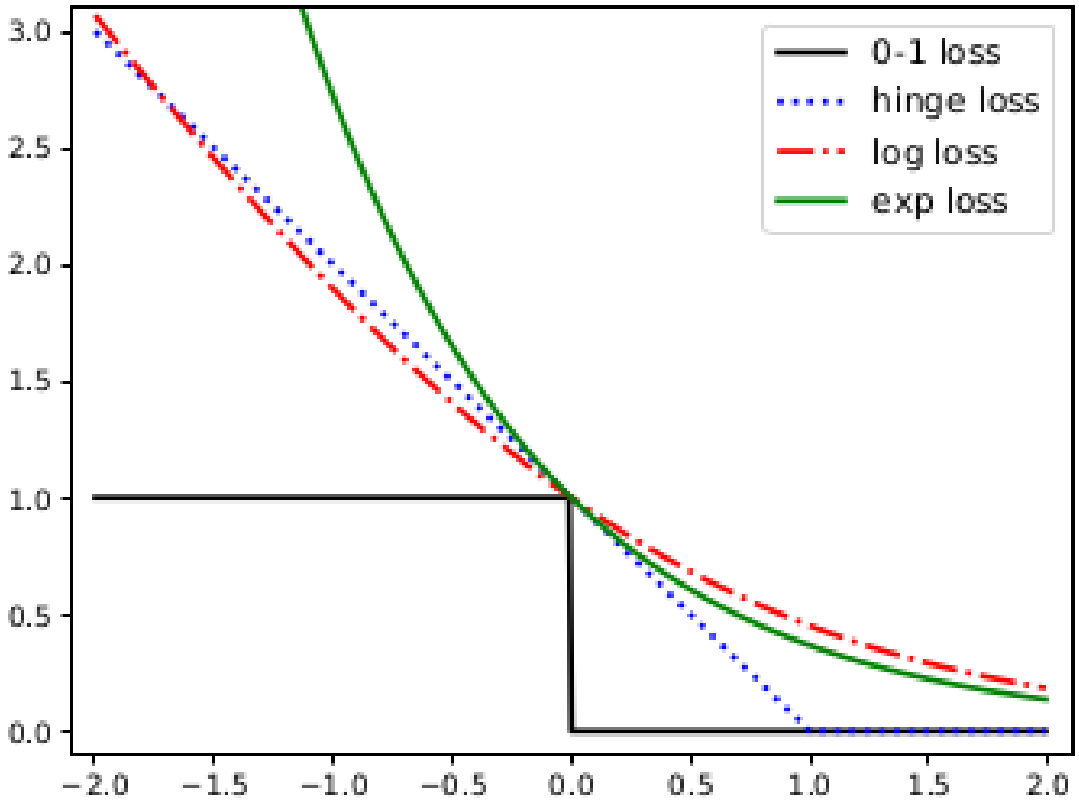
This is just the empirical misclassification rate on the training set.The corresponding empirical risk becomes:

$$\mathcal{L}(y_n, \hat{y}_n) = \frac{1}{N}\sum_{n=1}^{N} \mathrm{I}(\tilde{y}_n \hat{y}_n < 0)$$

where the dependence on x_n and θ is implicit.

Surrogate loss

The 0-1 loss used is a non-smooth step function, make it difficult to optimize. The surrogate is usually chosen to be a maximally tight convex upper bound, which is then easy to minimize.



The method of moments

Computing the MLE requires solving the equation $\nabla_\theta \text{NLL}(\theta) = 0$, but sometimes hard to solve. In such cases, we may be able to use a simpler approach known as the method of moments.In such cases, we may be able to use a simpler approach known as the method of moments(MOM).
In this approach, we equate the theoretical moments of the distribution to the empirical moments, and solve the resulting set of K simultaneous equations, where K is the number of parameters.
The theoretical moments are given by $\mu_k = \mathbb{E}[Y^k]$, for $k = 1, \ldots, K$, and the empirical moments are given by:

$$\hat{\mu}_k = \frac{1}{N} \sum_{n=1}^{N} y_n^k$$

so we just need to solve $\mu_k = \hat{\mu}_k$ for each k.

Example: MOM for the univariate Gaussian

For example, consider the case of a univariate Gaussian distribution:

$$\mu_1 = \mu = \bar{y}$$

$$\mu_2 = \sigma^2 + \mu^2 = s^2$$

Where $\bar{y}$ is the empirical mean and s^2 is the empirical average sum of squares. So $\hat{\mu} = \bar{y}$ and $\hat{\sigma}^2 = s^2 - \bar{y}^2$.In this case, the MOM estimate is the same as the MLE, but this is not always the case.

Example: MOM for the uniform distribution

Let $Y \sim \text{Unif}(\theta_1, \theta_2)$ be a uniform random variable, so :

$$p(y|\theta) = \frac{1}{\theta_2 - \theta_1} \mathbb{I}(\theta_1 \leq y \leq \theta_2)$$

The first two moments are

$$\mu_1 = \mathbb{E}[Y] = \frac{1}{2}(\theta_1 + \theta_2)$$

$$\mu_2 = \mathbb{E}[Y^2] = \frac{1}{3}(\theta_1^2 + \theta_1\theta_2 + \theta_2^2)$$

Inverting these equations gives:

$$(\theta_1, \theta_2) = \left(\mu_1 - \sqrt{3(\mu_2 - \mu_1^2)}, \mu_1 + \sqrt{3(\mu_2 - \mu_1^2)}\right)$$

Unfortunately, this estimator can sometimes give invalid results. For example, suppose $D = \{0, 0, 0, 0, 1\}$. The empirical moments are $\hat{\mu}_1 = \frac{1}{5}$ and $\hat{\mu}_2 = \frac{1}{5}$, so the estimated parameters are: $\hat{\theta}_1 = \frac{1}{5} - 2\sqrt{\frac{1}{5}(3)} = -0.493$ and $\hat{\theta}_2 = \frac{1}{5} + 2\sqrt{\frac{1}{5}(3)} = 0.893$. However, these cannot possibly be the correct parameters.

Online (recursive) estimation

If the entire dataset D is available before training starts, we say that we are doing *batch learning*. However, in some cases, the dataset arrives sequentially, so $D = \{y_1, y_2, \ldots\}$ in an unbounded stream. In this case, we want to perform *online learning*.

Example: recursive MLE for the mean of a Gaussian

We know that the batch estimate for the mean is given by:

$$\hat{\mu}_t = \frac{1}{t}\sum_{n=1}^{t} y_n$$

This is just a running sum of the data, so we can easily convert this into a recursive estimate as follows:

$$\hat{\mu}_t = \frac{1}{t}\sum_{n=1}^{t} y_n = \frac{(t-1)\hat{\mu}_{t-1} + y_t}{t}$$

$$= \hat{\mu}_{t-1} + \frac{y_t - \hat{\mu}_{t-1}}{t}$$

Exponentially-weighted moving average

We will compute the following exponentially weighted moving average or EWMA, also called an exponential moving average or EMA:

$$\hat{\mu}_t = \beta\mu_{t-1} + (1-\beta)y_t$$

Where $0 < \beta < 1$. The contribution of a data point k steps in the past is weighted by $\beta^k(1-\beta)$. Thus the contribution from old data is exponentially decreasing. In particular, we have :

$$\begin{aligned}\hat{\mu}_t &= \beta\mu_{t-1} + (1-\beta)y_t \\ &= \beta^2\mu_{t-2} + \beta(1-\beta)y_{t-1} + (1-\beta)y_t \\ &= \beta y_0 + (1-\beta)\beta y_1 + \cdots + (1-\beta)^{t-1}\beta y_{t-1} + (1-\beta)^t y_t\end{aligned}$$

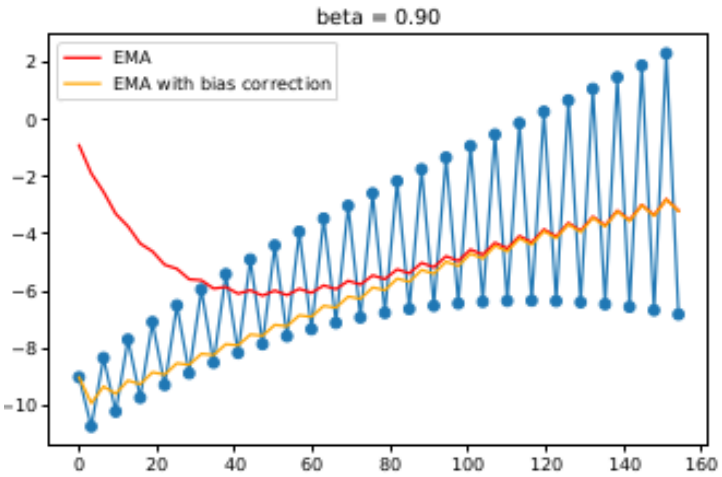
We have :

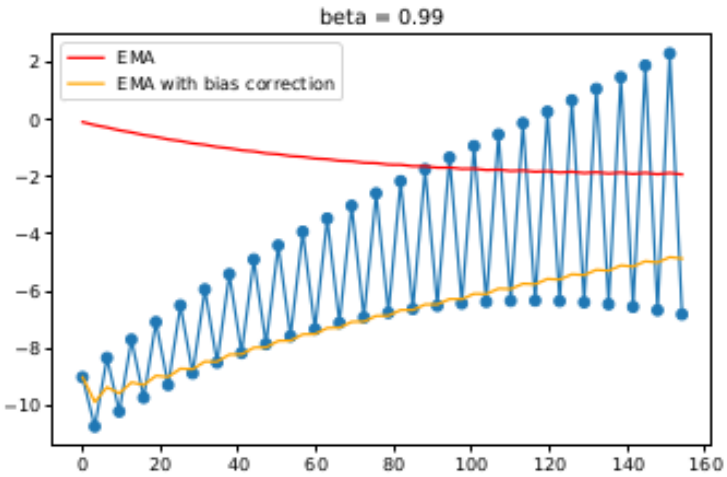
$$(1-\beta)\sum_{k=0}^{t}\beta^k = (1-\beta^t)$$

Exponentially-weighted moving average

Since $0 < \beta < 1$, we have $\beta^{t+1} \to 0$ as $t \to \infty$, so smaller β forgets the past more quickly, and adapts to the more recent data more rapidly. Since the initial estimate starts from $\hat{\mu}_0 = 0$, there is an initial bias. This can be corrected by scaling as follows:

$$\tilde{\mu}_t = \frac{\hat{\mu}_t}{1 - \beta^t}$$





Introduction

A fundamental problem with MLE, and ERM, is that it will try to pick parameters that minimize loss on the training set, but this may not result in a model that has low loss on future data. This is called overfitting.

The core of the problem is that the model has enough parameters to perfectly fit the observed training data, so it can perfectly match the empirical distribution. However, in most cases the empirical distribution is not the same as the true distribution, so putting all the probability mass on the observed set of N examples will not leave over any probability for novel data in the future. That is, the model may not generalize.

The main solution to overfitting is to use regularization, which means to add a penalty term to the NLL (or empirical risk). Thus we optimize an objective of the form :

$$L(\theta;\lambda) = \frac{1}{N}\sum_{n=1}^{N}(\mathcal{L}(y_n,\theta;x_n) + \lambda C(\theta))$$

Where $\lambda > 0$ and $C(\theta)$ is some form of complexity penalty.

Introduction

A common use of complexity penalty is $C(\theta) = -log(\theta)$, where $p(\theta)$ is the prior for θ.So, the regularized objective becomes:

$$L(\theta; \lambda) = -\frac{1}{N} \sum_{n=1}^{N} (\log p(y_n|x_n, \theta) - \lambda \log p(\theta))$$

By setting $\lambda = 1$, we can equivalently minimize the following:

$$L(\theta; \lambda) = -\frac{1}{N} \sum_{n=1}^{N} (\log p(y_n|x_n, \theta) + \log p(\theta)) = -\left[\log p(D|\theta) + \log p(\theta)\right]$$

Minimizing this is equivalent to maximizing the log posterior:

$$\hat{\theta} = \arg\max_{\theta} \log p(\theta|D) = \arg\max_{\theta} \left[\log p(D|\theta) + \log p(\theta) - \text{const}\right]$$

This is known as MAP estimation, which stands for maximum a posterior estimation.

Example: MAP estimation for the Bernoulli distribution

Consider again the coin tossing example.To avoid such overfitting, we can add a penalty to θ. We can do this by using a beta distribution as our prior, $p(\theta) = \text{Beta}(\theta|a, b)$ where $a, b > 1$ encourages values of θ near to $\frac{a}{a+b}$. The log likelihood plus log prior becomes :

$$\mathcal{L}(\theta) = \log p(D|\theta) + \log p(\theta)$$

$$= [N_1 \log \theta + N_0 \log(1-\theta)] + [(a-1)\log(\theta) + (b-1)\log(1-\theta)]$$

We find that the MAP estimate is:

$$\theta_{\text{map}} = \frac{N_1 + a - 1}{N_1 + N_0 + a + b - 2}$$

If we set $a = b = 2$ (which weakly favors a value of θ near 0.5), the estimate becomes:

$$\theta_{\text{map}} = \frac{N_1 + 1}{N_1 + N_0 + 2}$$

This is called add-one smoothing, and is a simple but widely used technique to avoid the zero count problem

Example: MAP estimation for the multivariate Gaussian *

Shrinkage estimate

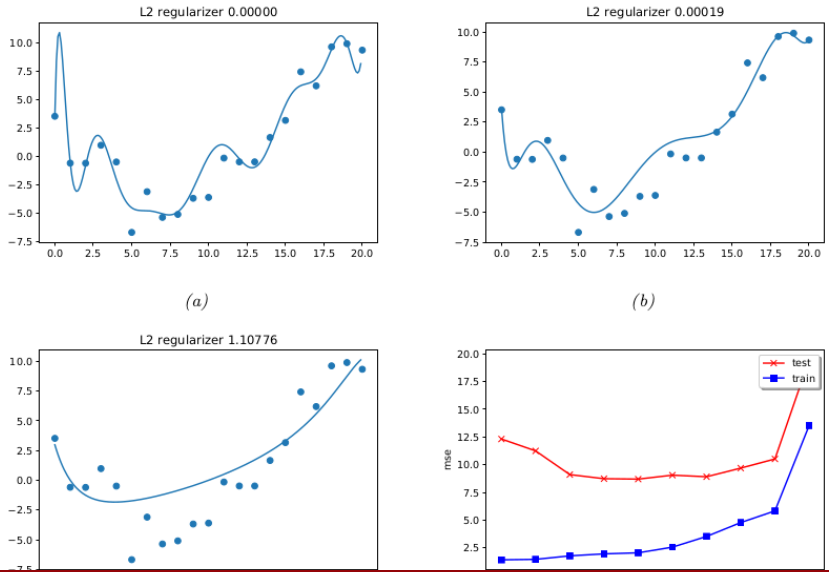
This is a distribution over positive definite matrices.The parameters are defined in terms of a prior scatter matrix, S, and a prior sample size or strength N.

Example: weight decay

We saw how using polynomial regression with too high of a degree can result in overfitting. One solution is to reduce the degree of the polynomial.We can do this by using a zero-mean Gaussian prior, $p(w)$. The resulting MAP estimate is given by:

$$\hat{w}_{\text{map}} = \arg\min \left(\text{NLL}(w) + \lambda \|w\|_2^2 \right)$$

Where $\|w\|_2^2 = \sum_{d=1}^{D} w_d^2$. This equaltion is called l_2 regularization or weight decay. In the case of linear regression, this kind of penalization scheme is called ridge regression.
The larger λ the less flexible model.



Picking the regularizer using a validation set

A key question when using regularization is how to choose the strength of the regularizer λ:
A small value means we will focus on minimizing empirical risk, which may result in overfitting.
A large value means we will focus on staying close to the prior, which may result in underfitting.
$\rightarrow$ **The basic idea is to partition the data into two disjoint sets, the training set D_{train} and a validation set D_{valid}.**
We fit the model on D_{train} (for each setting of λ) and then evaluate its performance on D_{valid}. We then pick the value of λ that results in the best validation performance.
Let us define the regularized empirical risk on a dataset as follows:

$$R_\lambda(\theta, D) = \frac{1}{|D|} \sum_{(x,y)\in D} \ell(y, f(x;\theta)) + \lambda C(\theta)$$

Picking the regularizer using a validation set

For each λ, we compute the parameter estimate :

$$\hat{\theta}_\lambda(D_{\text{train}}) = \arg\min_\theta R_\lambda(\theta, D_{\text{train}})$$

We then compute the **validation risk** :

$$R_\lambda^{\text{val}} \hat{=} \quad R_0\left(\hat{\theta}_\lambda(D_{\text{train}}), D_{\text{valid}}\right)$$

This is an estimate of the population risk, which is the expected loss under the true distribution $p^*(x,y)$. Finally, we pick :

$$\lambda^* = \arg\min_{\lambda \in S} R_\lambda^{\text{val}}$$

After picking λ^*, we can refit the model to the entire dataset, $D = D_{\text{train}} \cup D_{\text{valid}}$, to get :

$$\hat{\theta}^* = \arg\min_\theta R_{\lambda^*}(\theta, D)$$

Cross-validation

If the size of the training set is small, leaving aside 20% for a validation set can result in an unreliable estimate of the model parameters.
A simple but popular solution to this is to use cross validation (CV). The idea is as follows: we split the training data into K folds; then, for each fold $k \in \{1, \ldots, K\}$, we train on all the folds but the k-th, and test on the k-th. We have :

$$R_\lambda^{\text{cv}} = \frac{1}{K} \sum_{k=1}^{K} R_0 \left(\hat{\theta}_\lambda(D_{-k}), D_k \right)$$

Where D_k is the data in the k-th fold, and D_{-k} is all the other data.
We can use the CV estimate as an objective inside of an optimization routine to pick the optimal hyperparameter,

$$\hat{\lambda} = \arg\min_\lambda R_\lambda^{\text{cv}}.$$

Finally, we combine all the available data (training and validation), and re-estimate the model parameters using

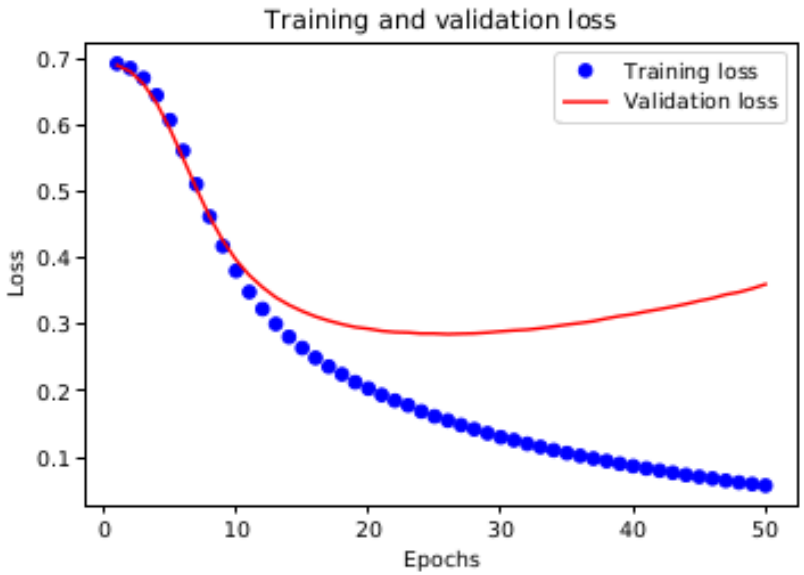
$$\hat{\theta} = \arg\min_\theta R_{\hat{\lambda}}(\theta, D).$$

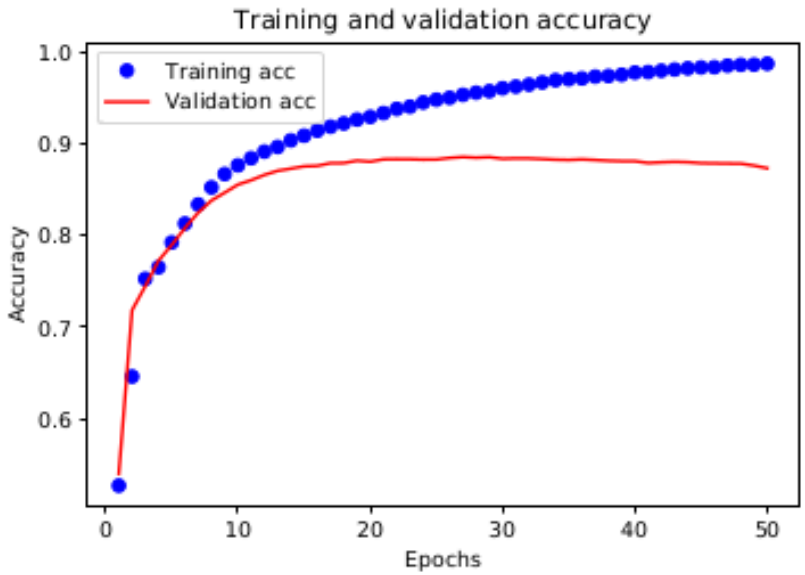
The one standard error rule

CV gives an estimate of $\hat{R}_\lambda$, but does not give any measure of uncertainty.
A standard frequentist measure of uncertainty of an estimate is the standard error of the mean, which is the mean of the sampling distribution of the estimate. We can compute this as follows:
Step1 : Let $L_n = \ell(y_n, f(x_n; \hat{\theta}_\lambda(D_{-n})))$ be the loss on the n-th example, where we use the parameters that were estimated using whichever training fold excludes n.
Step2 : Let $\hat{\mu} = \frac{1}{N}\sum_{n=1}^{N} L_n$ be the empirical mean and $\hat{\sigma}^2 = \frac{1}{N}\sum_{n=1}^{N}(L_n - \hat{\mu})^2$ be the empirical variance.
Step3 : We define our estimate to be $\hat{\mu}$, and the standard error of this estimate to be $\text{se}(\hat{\mu}) = \frac{\hat{\sigma}}{\sqrt{N}}$. Note that σ measures the intrinsic variability of L_n across samples, whereas $\text{se}(\hat{\mu})$ measures our uncertainty about the mean $\hat{\mu}$.

Early stopping

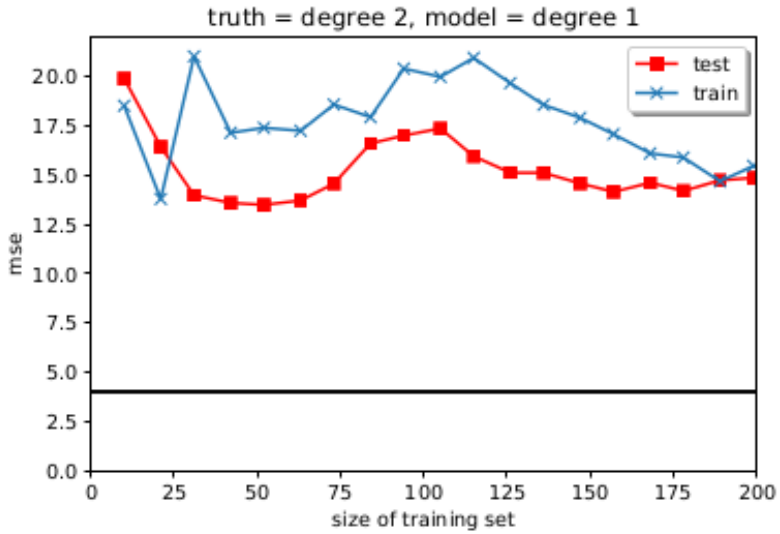
A very simple form of regularization, which is often very effective in practice (especially for complex models), is known as **early stopping** If we detect signs of overfitting (by monitoring performance on the validation set), we can stop the optimization process, to prevent the model memorizing too much information about the training set.



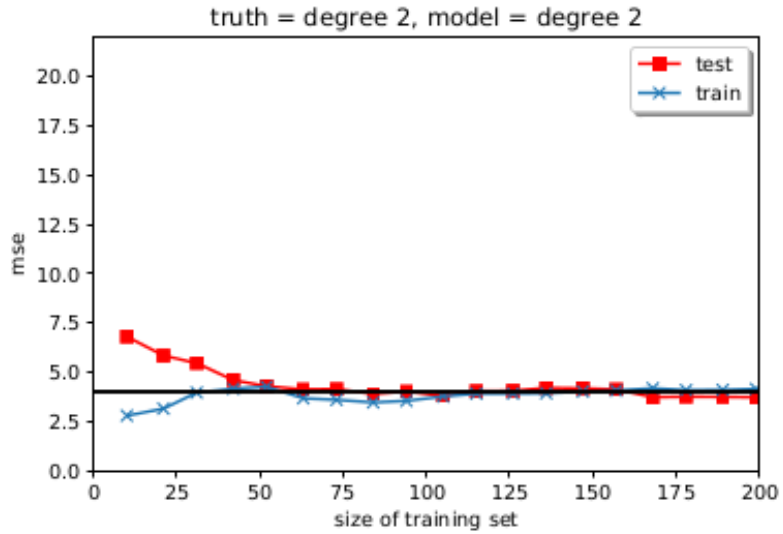


Using more data

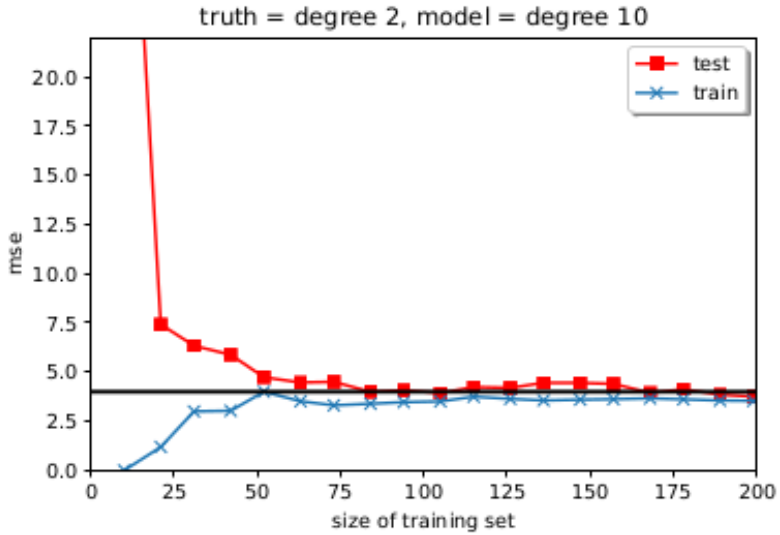
As the amount of data increases, the chance of overfitting (for a model of fixed complexity) decreases (assuming the data contains suitably informative examples, and is not too redundant).
We show the MSE on the training and test sets for four different models (polynomials of increasing degree) as a function of the training set size N . (A plot of error vs training set size is known as a learning curve.)
The horizontal black line represents the Bayes error, which is the error of the optimal predictor (the true model) due to inherent noise.
First, the test error for degree 1 remains high, even as N increases, since the model is too simple to capture the truth; this is called underfitting. The test error for the other models decreases to the optimal level (the noise floor), but it decreases more rapidly for the simpler models, since they have fewer parameters to estimate. The gap between the test error and training error is larger for more complex models, but decreases as N grows.
Second,the training error (blue line) initially increases with N , at least for the models that are sufficiently flexible


truth = degree 2, model = degree 1
test
train
mse
size of training set

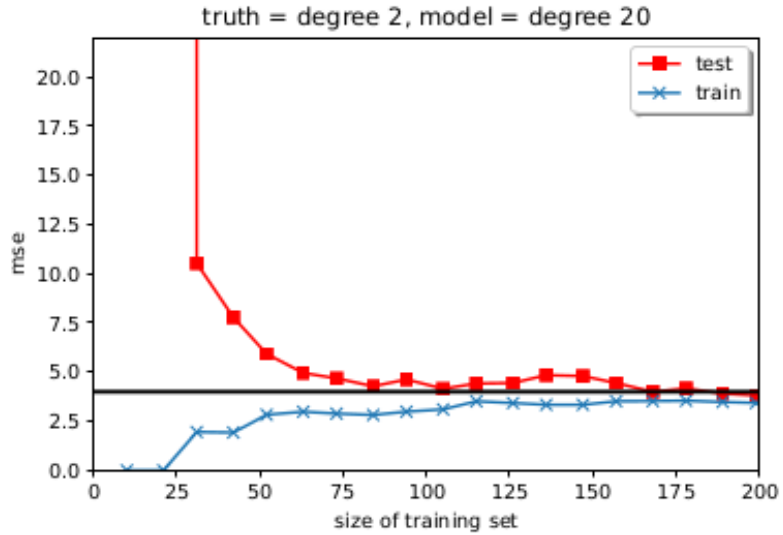
(a)


truth = degree 2, model = degree 2
test
train
mse
size of training set

(b)


truth = degree 2, model = degree 10
test
train
mse
size of training set

(c)


truth = degree 2, model = degree 20
test
train
mse
size of training set

(d)