$\LaTeX$ command declarations here.

EECS 545: Machine Learning

Lecture 05: Linear Regression II

- Instructor: **Jacob Abernethy**
- Date: January 25, 2015

Lecture Exposition Credit: Benjamin Bray and Zhe Du

Outline for this Lecture

- Overfitting
- Regularized Least Squares
- Locally-Weighted Linear Regression
- Maximum Likelihood Interpretation of Linear Regression

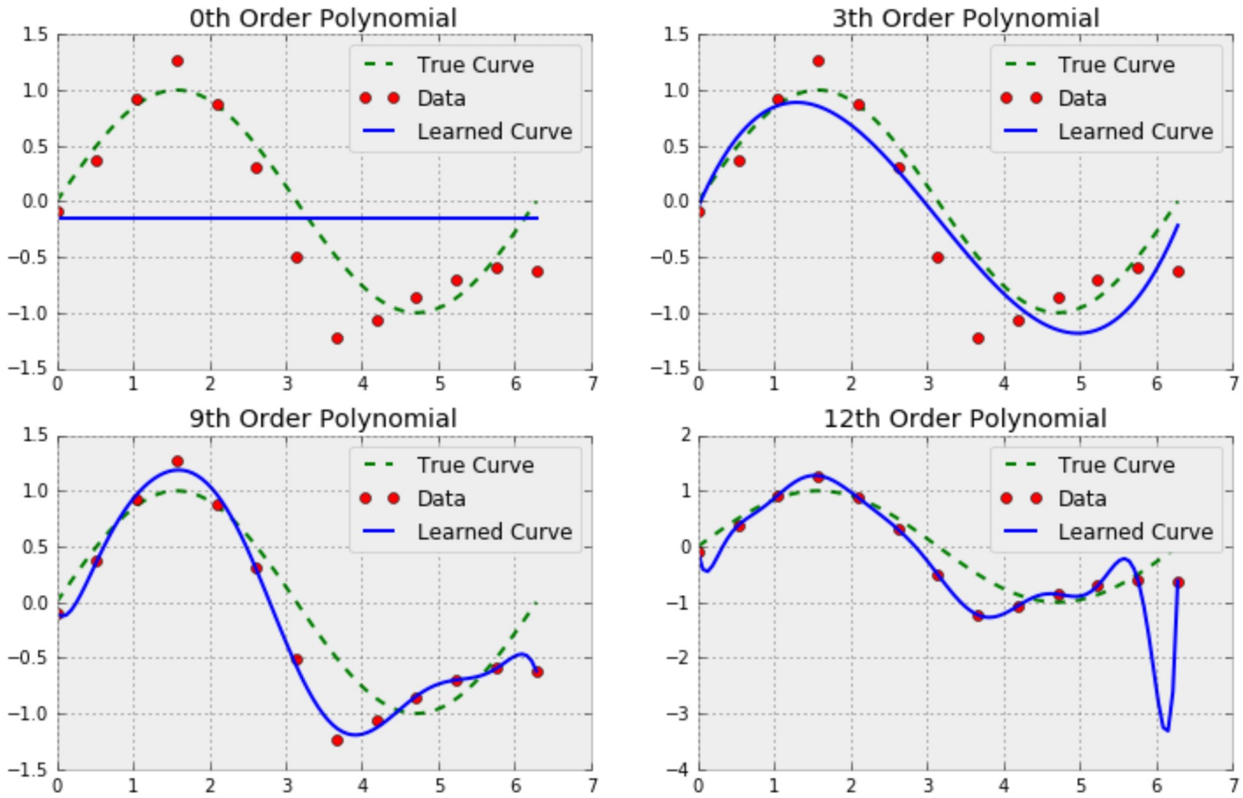
Reading List

- Required:
  - **[PRML]**, §3.2: The Bias-Variance Decomposition
  - **[PRML]**, §3.3: Bayesian Linear Regression
- Optional:
  - **[MLAPP]**, Chapter 7: Linear Regression

> In this lecture, we will first look at how degree of linear regression and sample dataset size will cause *overfitting* in linear regression. To deal with overfitting, *regularized least squares* will be introduced. When predicting the label of a new observation in linear regression, if we want to rely more on nearby training data than distant training data, we will resort to *locally-weighted linear regression*. Finally, we will show regular linear regression and regularized linear regression can be interpreted from probabilistic perspective each using *maximum likelihood estimation* and *maximum a posteriori estimation*.

Overfitting

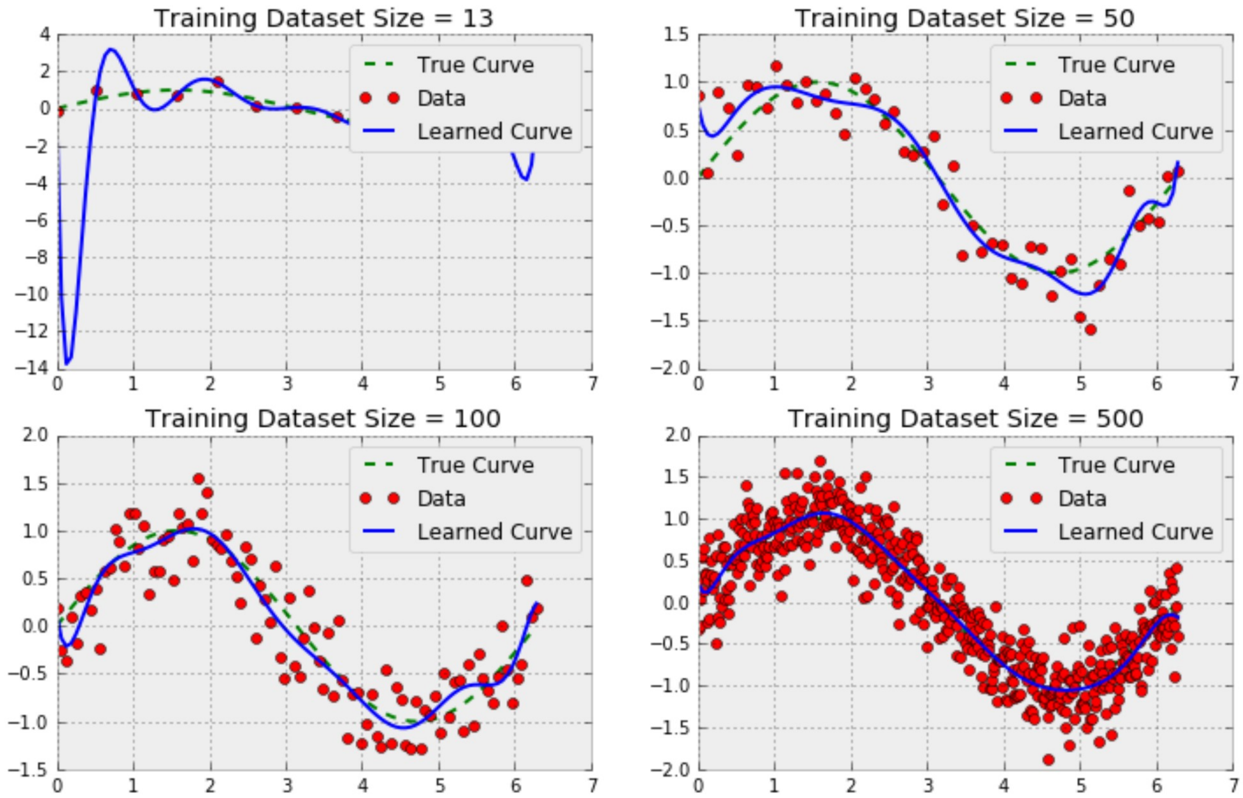
Overfitting: Degree of Linear Regression



Remark

- In the above plots, we try to predict the true sinusoidal curve hidden in the data with polynomial degrees 0, 3, 9 and 12.
- In the first plot (degree=0), we only get a horizontal line. The learned curve can neither fit the training data nor match true curve. This is called **Underfitting**.
- In the second plot (degree=3), the predicted plot fits both data and true curve perfectly. This is a good degree for our setting.
- In the third (degree=9) and fourth (degree=12) plots, as the degree increases, the traning data are fitted better but the learned curve deviates from the true curve further. This is called **Overfitting**.
- Explanations of why degree could impact the predicted curve will come in the **Remark** later.

Overfitting: Dataset Size

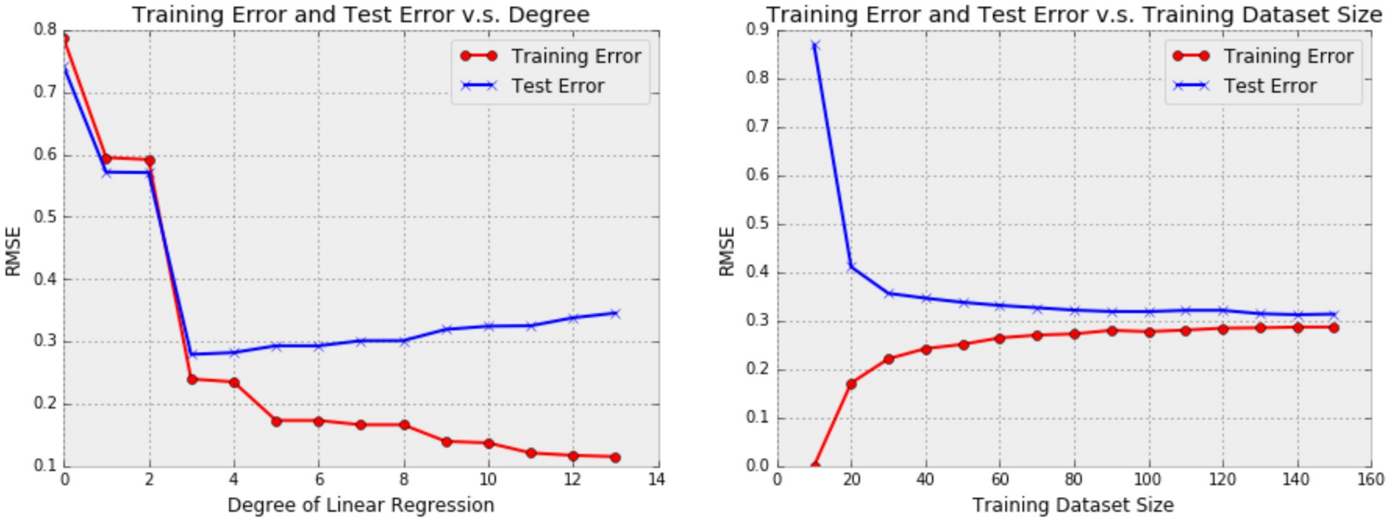


Remark

- In the above plots, we try to predict the true sinusoidal curve hidden in the data with polynomial degree 12 and training dataset size 13, 50, 100 and 500.
- In the first plot (size=13), **overfitting** occurs as we have seen just now.
- Comparing the four plots altogether, we could see as the size increases, overfitting diminishes. And although more and more data points cannot be fitted by the learned curve (training error is becoming higher), but it matches the true curve better (test error will be smaller).
- Explanations of why training data size impact the predicted curve will come in the **Remark** later.

Overfitting: Overall Performance

- On the left plot, we fix the dataset size and vary the polynomial degree
- On the right plot, we fix the polynomial degree and vary the dataset size



Remark

- The plot below is the **root mean squared error (RMSE)** of training dataset and test dataset with respect to different degree and dataset size.
- NOTE: For simplicity of presentation, we divided the dataset into training set and test set. However, it's not legitimate to find the optimal hyperparameter based on the test set. We will talk about legitimate ways of doing this when we cover model selection and cross-validation.
- Combining the last 10 plots, we could have:

- **Degree**
  - When degree is really small, the regressor is not powerful enough (i.e. degree is small) to learn the underlying true model. At this time, underfitting occurs. Both training error and test error are high.
  - As degree increases, regressor become more powerful enough to roughly learn the true model but not that powerful to also take the noise into considerations. At this time, underfitting reduces. Both training error and test error will be smaller.
  - As degree further increases, regressor become so powerful that it could fit most of the data in training dataset perfectly. And since the data are noisy, the learned model actually deviates from the true model. At this time, overfitting occurs. Training error could become very small (even 0 sometimes), while test error increases.

- **Dataset Size**
  - When training dataset size is small, a powerful regressor (i.e. degree is high) can fit every single sample in training dataset. Therefore, noise is also considered. This is equivalent to the ending zone of the plot (left) with respect degree.
  - As training dataset size increases, since the power of regressor is finite (degree is fixed), regressor starts to fail to fit every single sample. Instead it seeks to learn a curve such that samples will fall on both sides equally. Since the noise are assumed to 0 mean Gaussian noise which is symmetric with respect to 0, this is actually close to the underlying true curve.
  - The training error and test error will converge to 0.3 which is the variance of noise (you could check this by examining the Python code). This is not a coincidence and can be derived. We will cover this later when we study **bias-variance tradeoff**
- $$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(y_n - t_n)^2}$$

Rule of Thumb to Choose the Degree

- For a small number of datapoints, use a low degree
  - Otherwise, the model will overfit!
- As you obtain more data, you can gradually increase the degree
  - Add more features to represent more data
  - **Warning:** Your model is still limited by the finite amount of data available. The optimal model for finite data cannot be an infinite-dimensional polynomial!)
- Use **regularization** to control model complexity.

Regularized Linear Regression

Coefficients of Overfitting

- Before we move to regularized linear regression, let's first look at what happened to the coefficients $\mathbf{w}$ when there is overfitting.

	M=0 (Underfitting)	M=3 (Good)	M=9 (Overfitting)	M=12 (Overfitting)
w_0	10.1602	8.02909	0.122852	0.016863
w_1		-74.3778	-3.48275	-0.688068
w_2		147.913	40.871	12.458941
w_3		25.6371	-256.45	-131.571402
w_4			925.83	895.620409
w_5			-1917.45	-4100.361242
w_6			2120.59	12789.763400
w_7			-1063.37	-26957.669491
w_8			208.248	37298.742788
w_9			41.1305	-31986.424244
w_10				15153.404691
w_11				-2906.327049
w_12				41.318900

Remark

- The table above corresponds to the coefficients (multiplied by 100 for better visualization) for different degrees.
- We could see that when overfitting occurs, we get some really crazy and large numbers!
- So one intuition to handle overfitting is to *penalize* large coefficients.

Regularized Least Squares: Objective Function

- Recall the objective function we minimizes in last lecture is

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(\mathbf{w}^T\phi(\mathbf{x}_n) - t_n\right)^2$$

- To penalize the large coefficients, we will add one penalization/regularization term to it and minimize them altogether.

$$E(\mathbf{w}) = \underbrace{\frac{1}{2}\sum_{n=1}^{N}\left(\mathbf{w}^T\phi(\mathbf{x}_n) - t_n\right)^2}_{E_D(\mathbf{w})} + \underbrace{\boxed{\frac{\lambda}{2}\|\mathbf{w}\|^2}}_{E_W(\mathbf{w})}$$

  of which $E_D(\mathbf{w})$ represents the term of sum of squared errors and $E_W(\mathbf{w})$ is the regularization term.
- λ is the regularization coefficient.
- If λ is large, $E_{\mathbf{W}}(\mathbf{w})$ will dominate the objective function. As a result we will focus more on minimizing $E_W(\mathbf{w})$ and the resulting solution $\mathbf{w}$ tends to have smaller norm and the $E_D(\mathbf{w})$ term will be larger.

Regularized Least Squares: Derivation

- Based on what we have derived in last lecture, we could write the objective function as

$$\begin{aligned} E(\mathbf{w}) &= \frac{1}{2}\sum_{n=1}^{N}\left(\mathbf{w}^T\phi(\mathbf{x}_n) - t_n\right)^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2 \\ &= \frac{1}{2}\mathbf{w}^T\Phi^T\Phi\mathbf{w} - \mathbf{t}^T\Phi\mathbf{w} + \frac{1}{2}\mathbf{t}^T\mathbf{t} + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} \end{aligned}$$

- The gradient is

$$\begin{aligned} \nabla_{\mathbf{w}}E(\mathbf{w}) &= \Phi^T\Phi\mathbf{w} - \Phi^T\mathbf{t} + \lambda\mathbf{w} \\ &= (\Phi^T\Phi + \lambda I)\mathbf{w} - \Phi^T\mathbf{t} \end{aligned}$$

- Setting the gradient to 0, we will get the solution

$$\boxed{\hat{\mathbf{w}} = (\Phi^T\Phi + \lambda I)^{-1}\Phi^T\mathbf{t}}$$

- In the solution to ordinary least squares which is $\hat{\mathbf{w}} = (\Phi^T\Phi)^{-1}\Phi^T\mathbf{t}$, we cannot guarantee $\Phi^T\Phi$ is invertible. But in regularized least squares, if $\lambda > 0$, $\Phi^T\Phi + \lambda I$ is always invertible.

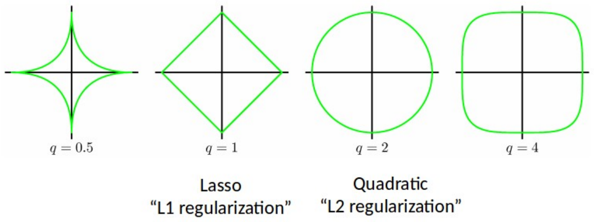
Regularized Least Squares: Different Norms

- The ℓ^p norm of a vector $\mathbf{x}$ is defined as

$$\|\mathbf{x}\|_p = (\sum_{j=1}^{M} |x_j|^p)^{\frac{1}{p}}$$

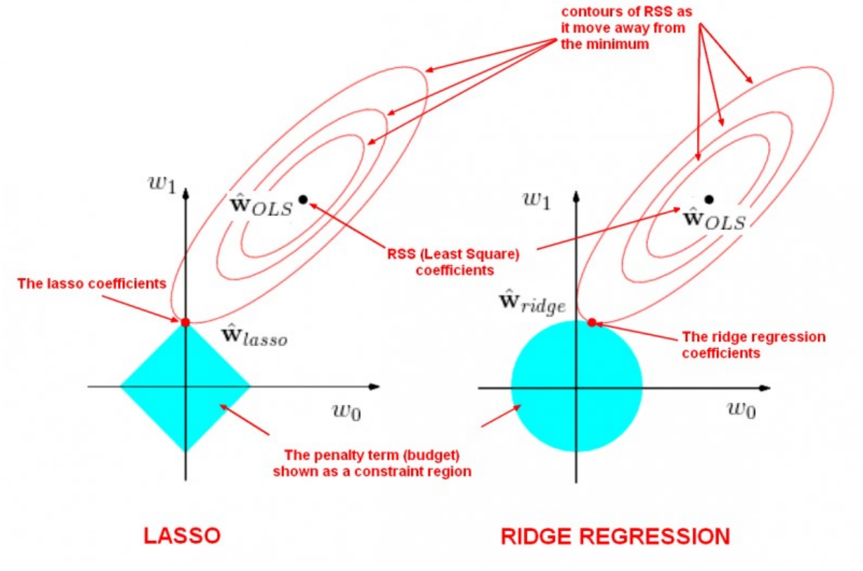
- For the regularized least squares above, we used ℓ^2 norm. We could also use other ℓ^p norms for different regularizers and the objective function becomes

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(\mathbf{w}^T\phi(\mathbf{x}_n) - t_n\right)^2 + \frac{\lambda}{2}\|\mathbf{w}\|_p^p$$



(Contour of Different p-norms)

- **Lasso** regularization (ℓ^1 regularization) tends to generate sparser solutions than **ridge** regression (ℓ^2 regularization)


contours of RSS as it move away from the minimum
w_1
$\hat{\mathbf{w}}_{OLS}$
RSS (Least Square) coefficients
The lasso coefficients
$\hat{\mathbf{w}}_{lasso}$
w_0
The penalty term (budget) shown as a constraint region
LASSO
w_1
$\hat{\mathbf{w}}_{OLS}$
$\hat{\mathbf{w}}_{ridge}$
The ridge regression coefficients
w_0
RIDGE REGRESSION

(Image taken from [here](http://gerardnico.com/wiki/data_mining/lasso))

Remark

- RSS is residual of sum of squares, which is the sum of squared errors $E_D(\mathbf{w})$ we use.
- This plot is to illustrate intuitively why lasso has sparser solution than ridge regression.
- Our objective function is

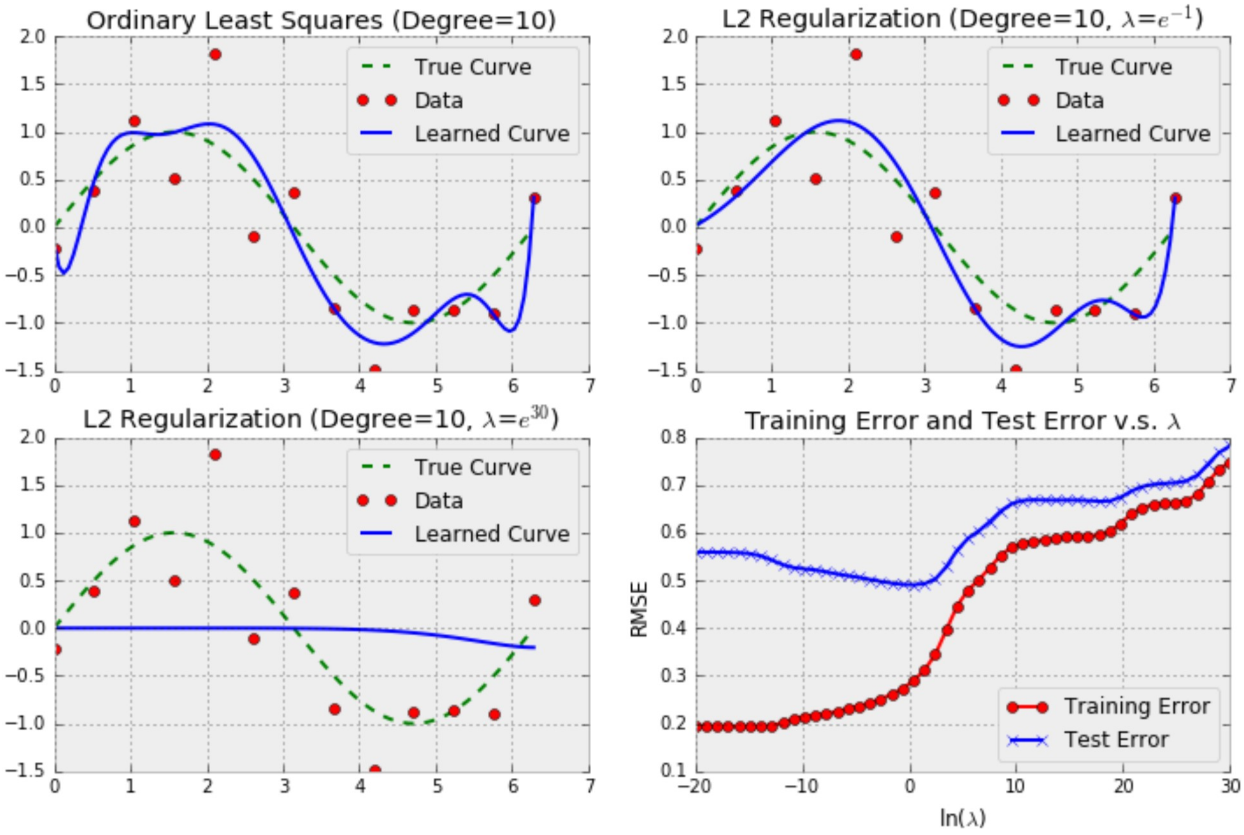
$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(\mathbf{w}^T\phi(\mathbf{x}_n) - t_n\right)^2 + \frac{\lambda}{2}\|\mathbf{w}\|_p^p$$

To illustrate, lets look at an *equivalent* constrained problem. (Not exactly equivalent, just for illustration purpose)

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\sum_{n=1}^{N}\left(\mathbf{w}^T\phi(\mathbf{x}_n) - t_n\right)^2 \\ \text{subject to} \quad & \frac{\lambda}{2}\|\mathbf{w}\|_p^p \le C \end{aligned}$$

- The objective function $\frac{1}{2}\sum_{n=1}^{N}\left(\mathbf{w}^T\phi(\mathbf{x}_n)-t_n\right)^2$ is the red contours in the plots. The optimal solution without the constraint is $\hat{\mathbf{w}}_{OLS}$, which is just the solution to ordinary least squares. The farther we are away from $\hat{\mathbf{w}}_{OLS}$ the larger objective function will be.
- The feasible area that satisfying the constraint is the cyan area in the plots. The solution must fall in these areas. For lasso, it's a diamond; for ridge regression, it's a circle.
- As we increase the value of objective function, the contour will expand. The first time the contour *touch* the feasible cyan area, the touching point would be the optimal solution. This is because our solution should both be as close to $\hat{\mathbf{w}}_{OLS}$ as possible and fall in the feasible area.
- Since lasso has diamond area with four corners, the contours tend to touch the corner first. And since the corner is on the axis where coordinate has at least one zero component, this guarantees the sparsity of solution. On the contrary, the circle area in ridge regression cannot give us this property.

Regularized Least Squares: Example



Remark

- In the second plot, we can see that after the regularization term is added, the learned curve looks much more like the true curve than the learned curve without regularization in the first plot.
- However, in the third plot, when the coefficient λ for regularization is too large, the minimization will mostly focus on minimizing $\|\mathbf{w}\|$, thus deviating from the true curve in a great deal. (We will see the coefficients $\mathbf{w}$) are really small for this case in next slide)
- In the fourth plot, as we increase λ, the training error is monotonically increasing because we are far away from the task of minimizing sum of squared error $E_D(\mathbf{w})$. As for the the test error, it has minimum value near $\lambda=1$, which is the balancing point in the tradeoff between minimizing $E_D(\mathbf{w})$ and minimizing regularization term $E_W(\mathbf{w})$.

Regularized Least Squares: Coefficients

- Let's look at how the coefficients change after we add regularization

	lambda=0	lambda=exp^1	lambda=exp^10
w_0	-37.944695	3.526268	0.013455
w_1	-9695.495916	21.044695	0.024688
w_2	41314.376371	16.316530	0.041741
w_3	-68438.476447	10.130538	0.071136
w_4	60334.703714	2.893024	0.112146
w_5	-31806.204765	-2.888794	0.144259
w_6	10530.716110	-3.110100	0.110873
w_7	-2210.274340	2.108990	-0.022436
w_8	285.581838	-0.479255	-0.031165
w_9	-20.723502	0.048360	0.009544
w_10	0.646246	-0.001850	-0.000727

Remark

- From the table above, we can see the regularization term has effectively constrained those huge coefficients.
- However, when λ is large ($\lambda = e^{10}$), the coefficients are "over-regularized".

Regularized Least Squares: Summary

- Simple modification of linear regression
- ℓ^2 Regularization controls the tradeoff between *fitting error* and *complexity*.
  - Small ℓ^2 regularization results in complex models, but with risk of overfitting
  - Large ℓ^2 regularization results in simple models, but with risk of underfitting
- It is important to find an optimal regularization that *balances* between the two

Locally-Weighted Linear Regression

Locally-Weighted Linear Regression

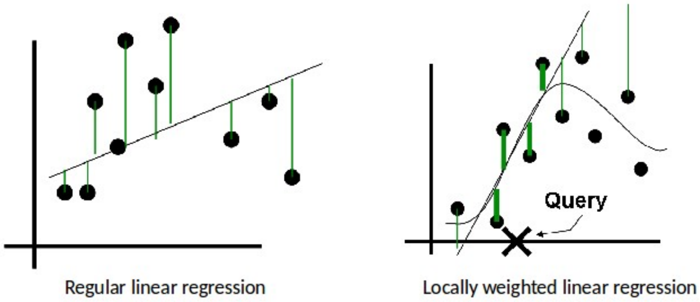
- **Main Idea:** Given a new observation $\mathbf{x}$, we generate the coefficients $\mathbf{w}$ and prediction $y(\mathbf{x}, \mathbf{w})$ by giving high weights for *neighbors* of $\mathbf{x}$.
- **Regular vs. Locally-Weighted Linear Regression**
- **Linear Regression**

1. Fit $\mathbf{w}$ to minimize $\sum_n (\mathbf{w}^T\phi(\mathbf{x}_n) - t_n)^2$ of which $\{(\mathbf{x}_n, t_n)\}_{n=1}^N$ is the training dataset.
2. For every new observation $\mathbf{x}$ to be predicted, output $\mathbf{w}^T\phi(x)$
Note: **One** $\mathbf{w}$ for **all** observations to be predicted.

- **Locally-weighted Linear Regression**

1. For **every** new observation $\mathbf{x}$ to be predicted, generate the weights r_n for every training sample $(\mathbf{x}_n, t_n)$. (The closer $\mathbf{x}_n$ is to $\mathbf{x}$, the larger r_n will be)
2. Fit $\mathbf{w}$ to minimize $\sum_n r_n(\mathbf{w}^T\phi(\mathbf{x}_n) - t_n)^2$ of which $\{(\mathbf{x}_n, t_n)\}_{n=1}^N$ is the training dataset.
3. Output $\mathbf{w}^T\phi(x)$
Note: **One** $\mathbf{w}$ for **only one** observations to be predicted.

Regular vs. Locally-Weighted Linear Regression



</center>

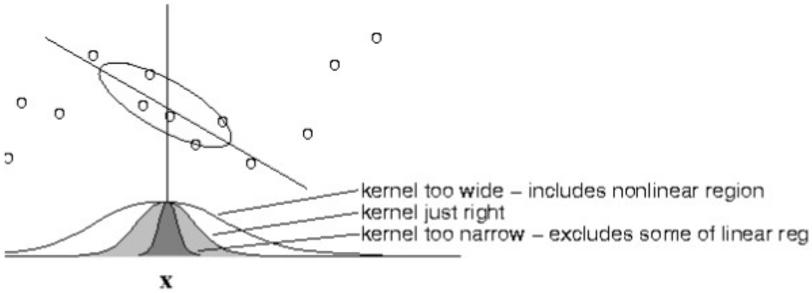
- For linear regression with **degree=1**, the learned curve for regular linear regression is a **straight line** while the learned curve for locally-weighted linear regression is a **curve**. For queries/observations with different features (x-axis value), the weights $\mathbf{r}$ are different thus the solution $\hat{\mathbf{w}}$ are also different. For 1-degree regression, $\hat{\mathbf{w}}$ correspond to the interception and slope of learned curve, so the slope for locally-weighted linear regression is always changing and what we get cannot be a straight line.

Locally-Weighted Linear Regression: Weights

- The standard choice for weights $\mathbf{r}$ uses the **Gaussian Kernel**, with **kernel width** τ

$$r_n = \exp\left(-\frac{||\mathbf{x}_n - \mathbf{x}||^2}{2\tau^2}\right)$$

- Choice of kernel width matters.



- The bell shape is the weight curve, which has maximum at the query point $\mathbf{x}$ and decreases as we move farther.
- The best kernel includes as many training points as can be accommodated by the model. Too large a kernel includes points that degrade the fit; too small a kernel neglects points that increase confidence in the fit.

Remark

- Note weight $\mathbf{r}$ and resulting $\hat{\mathbf{w}}$ depend on $\mathbf{x}$ (query point); we must solve linear regression for each query point $\mathbf{x}$.
- Can be reformulated as a modified version of least squares problem.
- The best kernel width requires hyperparameter tuning

Locally-Weighted Linear Regression: Derivation

- Recall that in regular linear regression, we have

$$E(\mathbf{w}) = \sum_{n=1}^{N} \left(\mathbf{w}^T \phi(\mathbf{x}_n) - t_n\right)^2 = \|\Phi \mathbf{w} - \mathbf{t}\|^2$$

- In locally-weighted linear regression, we are to minimize

$$E(\mathbf{w}) = \sum_{n=1}^{N} r_n (\mathbf{w}^T \phi(\mathbf{x}_n) - t_n)^2 = \sum_{n=1}^{N} (\sqrt{r_n}\,\mathbf{w}^T \phi(\mathbf{x}_n) - \sqrt{r_n}\, t_n)^2 = \|\sqrt{R}\Phi \mathbf{w} - \sqrt{R}\mathbf{t}\|^2$$

  of which

$$R = \begin{bmatrix} r_1 & & & \\ & r_2 & & \\ & & \ddots & \\ & & & r_N \end{bmatrix}$$

- Recall the solution to $\arg\min \|\Phi \mathbf{w} - \mathbf{t}\|^2$ is $\Phi^\dagger \mathbf{t}$. Similarly, the solution to $\arg\min \left\|\sqrt{R}\Phi \mathbf{w} - \sqrt{R}\mathbf{t}\right\|^2$ is

$$\boxed{\hat{\mathbf{w}} = (\sqrt{R}\Phi)^\dagger \sqrt{R}\mathbf{t}}$$

> **Remark**
>
> - When $\sqrt(R)\Phi$ has linearly independent columns, the solution can be written as
>
> $$\boxed{\hat{\mathbf{w}} = (\Phi^T R \Phi)^{-1} \Phi R \mathbf{t}}$$
>
> *Recall when A has linearly independent columns, $A^\dagger = (A^T A)^{-1} A^T$*
>
> - We omitted $\frac{1}{2}$ in the above $E(\mathbf{w})$ because it will not affect the solution $\mathbf{w}$. We add $\frac{1}{2}$ in last lecture because it can cancel out the coefficient 2 generated by differentiating the quadratic expression.

Probabilistic Interpretation of Least Squares Regression

- We have showed derived the solution to least squares regression by minimizing objective function. Now we will provide a probabilistic perspective. Specifically, we will show the solution to **regular least squares** is just the **maximum likelihood** estimate of $\mathbf{w}$ and the solution to **regularized least squares** is the **Maximum a Posteriori** estimate.

Some Background

- Gaussian Distribution

$$\mathcal{N}(x, \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[\frac{(x-\mu)^2}{2\sigma^2}\right]$$

- **Maximum Likelihood Estimation** and **Maximum a Posteriori Estimation (MAP)**
  - For distribution $t \sim p(t|\theta)$. θ is some unknown parameter (like mean or variance) to be estimated.
  - Given observation $\mathbf{t} = (t_1, t_2, \ldots, t_N)$,
    - The Maximum Likelihood Estimator is

$$\theta_{ML} = \arg\max \prod_{n=1}^{N} p(t_n|\theta)$$

    - If we have some prior knowledge about θ, the MAP estimator is

$$\theta_{MAP} = \arg\max \prod_{n=1}^{N} p(\theta|t_n) \quad (\text{Posteriori Probability of } \theta)$$

Maximum Likelihood Estimator $\mathbf{w}_{ML}$

- We assume the **signal+noise** model of single data $(\mathbf{x}, t)$ is
$$t = \mathbf{w}^T\phi(\mathbf{x}) + \epsilon$$
$$\epsilon \sim \mathcal{N}(0, \beta^{-1})$$
of which $\mathbf{w}^T\phi(\mathbf{x})$ is the true model, ϵ is the perturbation/randomness.
- Since $\mathbf{w}^T\phi(\mathbf{x})$ is deterministic/non-random, we have
$$t \sim \mathcal{N}(\mathbf{w}^T\phi(\mathbf{x}), \beta^{-1})$$
- The **likelihood function** of t is just **probability density function (PDF)** of t
$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|\mathbf{w}^T\phi(\mathbf{x}), \beta^{-1})$$
- For inputs $\mathcal{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and target values $\mathbf{t} = (t_1, \ldots, t_n)$, the data likelihood is
$$p(\mathbf{t}|\mathcal{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} p(t_n|\mathbf{x}_n, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^T\phi(\mathbf{x}_n), \beta^{-1})$$
- **Notation Clarification**
  - $p(t|x, w, \beta)$ it the PDF of t whose distribution is parameterized by $x, \mathbf{w}, \beta$.
  - $\mathcal{N}(\mathbf{w}^T\phi(\mathbf{x}), \beta^{-1})$ is Gaussian distribution with **mean** $\mathbf{w}^T\phi(\mathbf{x})$ and **variance** β^{-1}.
  - $\mathcal{N}(t|\mathbf{w}^T\phi(\mathbf{x}), \beta^{-1})$ is the PDF of $\mathbf{t}$ which has Gaussian distribution $\mathcal{N}(\mathbf{w}^T\phi(\mathbf{x}), \beta^{-1})$

Maximum Likelihood Estimator $\mathbf{w}_{ML}$

- **Main Idea of Maximum Likelihood Estimate**
  - Given $\{\mathbf{x}_n, t_n\}_{n=1}^N$, we want to find $\mathbf{w}_{ML}$ that maximizes data likelihood function
$$\mathbf{w}_{ML} = \arg\max p(\mathbf{t}|\mathcal{X}, \mathbf{w}, \beta) = \arg\max \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^T\phi(\mathbf{x}_n), \beta^{-1})$$
and by derivation we will show $\mathbf{w}_{ML}$ is equivalent to the least squares solution $\hat{\mathbf{w}} = \Phi^\dagger \mathbf{t}$.
- **Intuition about Maximum Likelihood Estimation**
  - Finding maximum likelihood estimate $\mathbf{w}_{ML} = \arg\max p(\mathbf{t}|\mathcal{X}, \mathbf{w}, \beta)$ is just finding the parameter $\mathbf{w}$ under which for data $\mathcal{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, observed $\mathbf{t} = (t_1, \ldots, t_n)$ is the most likely result to be generated among all possible $\mathbf{t}$.

Maximum Likelihood Estimator $\mathbf{w}_{ML}$: Derivation

- Single data likelihood is
$$p(t_n|\mathbf{x}_n, \mathbf{w}, \beta) = \mathcal{N}(t_n|\mathbf{w}^T\phi(\mathbf{x}_n), \beta^{-1}) = \frac{1}{\sqrt{2\pi\beta^{-1}}}\exp\left\{-\frac{1}{2\beta^{-1}}(t_n - \mathbf{w}^T\phi(x_n))^2\right\}$$
- Single data log-likelihood is
$$\ln p(t_n|\mathbf{x}_n, \mathbf{w}, \beta) = -\frac{1}{2}\ln 2\pi\beta^{-1} - \frac{\beta}{2}(\mathbf{w}^T\phi(x_n) - t_n)^2$$
We use logarithm because maximizer of $f(x)$ is the same as maximizer of $\log f(x)$. Logarithm can convert product to summation which makes life easier.
- Complete data log-likelihood is
$$\begin{aligned}\ln p(\mathbf{t}|\mathcal{X}, \mathbf{w}, \beta) &= \ln\left[\prod_{n=1}^{N} p(t_n|\mathbf{x}_n, \mathbf{w}, \beta)\right] = \sum_{n=1}^{N} \ln p(t_n|\mathbf{x}_n, \mathbf{w}, \beta) \\ &= \sum_{n=1}^{N}\left[-\frac{1}{2}\ln 2\pi\beta^{-1} - \frac{\beta}{2}(\mathbf{w}^T\phi(x_n) - t_n)^2\right]\end{aligned}$$

- Maximum likelihood estimate $\mathbf{w}_{ML}$ is

$$\begin{aligned}\mathbf{w}_{ML} &= \arg\max_{\mathbf{w}} \ln p(\mathbf{t}|\mathcal{X},\mathbf{w},\beta)\\ &= \arg\max_{\mathbf{w}} \sum_{n=1}^{N}\left[-\frac{1}{2}\ln 2\pi\beta^{-1} - \frac{\beta}{2}(\mathbf{w}^T\phi(x_n)-t_n)^2\right]\\ &= \arg\max_{\mathbf{w}} \sum_{n=1}^{N}\left[-\frac{\beta}{2}(\mathbf{w}^T\phi(x_n)-t_n)^2\right]\\ &= \arg\min_{\mathbf{w}} \sum_{n=1}^{N}\left[(\mathbf{w}^T\phi(x_n)-t_n)^2\right]\end{aligned}$$

- Familiar? Recall the objective function we minimized in least squares is $E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(\mathbf{w}^T\phi(\mathbf{x}_n)-t_n\right)^2$, so we could conclude that

$$\boxed{\mathbf{w}_{ML} = \hat{\mathbf{w}}_{LS} = \Phi^{\dagger}\mathbf{t}}$$

MAP Estimator $\mathbf{w}_{MAP}$

- The **MAP estimator** is obtained by

$$\begin{aligned}\mathbf{w}_{MAP} &= \arg\max p(\mathbf{w}|\mathbf{t},\mathcal{X},\beta) && \text{(Posteriori Probability)}\\ &= \arg\max \frac{p(\mathbf{w},\mathbf{t},\mathcal{X},\beta)}{p(\mathcal{X},t,\beta)}\\ &= \arg\max \frac{p(\mathbf{t}|\mathbf{w},\mathcal{X},\beta)p(\mathbf{w},\mathcal{X},\beta)}{p(\mathcal{X},t,\beta)}\\ &= \arg\max p(\mathbf{t}|\mathbf{w},\mathcal{X},\beta)p(\mathbf{w},\mathcal{X},\beta) && (p(X,t,\beta)\text{ is irrelevant to }\mathbf{w})\\ &= \arg\max p(\mathbf{t}|\mathbf{w},\mathcal{X},\beta)p(\mathbf{w})p(\mathcal{X})p(\beta) && \text{(Independence)}\\ &= \arg\max p(\mathbf{t}|\mathbf{w},\mathcal{X},\beta)p(\mathbf{w}) && \text{(Likelihood}\times\text{Prior)}\end{aligned}$$

  We are just using **Bayes Theorem** for the above steps.
- The only difference from ML estimator is we have an extra term of PDF of $\mathbf{w}$. This is the **prior belief** of $\mathbf{w}$. Here, we assume,

$$\mathbf{w}\sim\mathcal{N}(\mathbf{0},\alpha^{-1}I)$$

- ML vs. MAP
  - Maximum Likelihood: We know **nothing** about $\mathbf{w}$ initially and every $\mathbf{w}$ are equally likelihood
  - Maximum a Posteriori: We know **something** about about $\mathbf{w}$ initially and certain $\mathbf{w}$ are more likely (depending on **prior** $p(\mathbf{w})$). In another way, $\mathbf{w}$ are weighted.
- Assumption $\mathbf{w}\sim\mathcal{N}(\mathbf{0},\alpha^{-1}I)$ makes sense because
  - In **regularized least squares**
    - We already know large coefficient $\mathbf{w}$ that may lead to overfitting should be avoided.
    - When we increase the regularization coefficient λ, the smaller $\|\mathbf{w}\|$ will be.
  - When use $\mathbf{w}\sim\mathcal{N}(\mathbf{0},\alpha^{-1}I)$
    - $\mathbf{w}\sim\mathcal{N}(\mathbf{0},\alpha^{-1}I)$ encodes the assumption that $\mathbf{w}$ with a smaller norm $\|\mathbf{w}\|$ is more "likely" than a $\mathbf{w}$ with a bigger norm.
    - When we increase α, variance is smaller, small $\|\mathbf{w}\|$ will be much more likely

MAP Estimator $\mathbf{w}_{MAP}$: Derivation

- $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \alpha^{-1}I)$ is **multivariate Gaussian** which has PDF

$$p(\mathbf{w}) = \frac{1}{\left(\sqrt{2\pi\alpha^{-1}}\right)^N} \exp\left\{-\frac{1}{2\alpha^{-1}} \sum_{n=1}^{N} w_n^2\right\}$$

- So the MAP estimator is

$$\begin{aligned}\mathbf{w}_{MAP} &= \arg\max_{\mathbf{w}} p(\mathbf{t}|\mathbf{w}, \mathcal{X}, \beta)p(\mathbf{w}) = \arg\max_{\mathbf{w}} \left[\ln p(\mathbf{t}|\mathbf{w}, \mathcal{X}, \beta) + \ln p(\mathbf{w})\right] \\ &= \arg\min_{\mathbf{w}} \left[\sum_{n=1}^{N} \frac{\beta}{2}(\mathbf{w}^T\phi(x_n) - t_n)^2 + \frac{\alpha}{2}\sum_{n=1}^{N} w_n^2\right] \\ &= \arg\min_{\mathbf{w}} \left[\sum_{n=1}^{N} \frac{1}{2}(\mathbf{w}^T\phi(x_n) - t_n)^2 + \frac{1}{2}\frac{\alpha}{\beta}\|\mathbf{w}\|^2\right]\end{aligned}$$

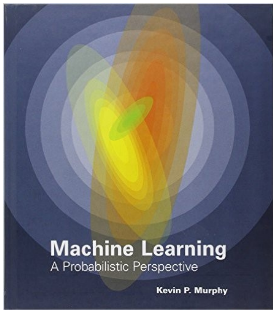
- Exactly the objective in regularized least squares! So

$$\boxed{\mathbf{w}_{MAP} = \hat{\mathbf{w}} = \left(\Phi^T\Phi + \frac{\alpha}{\beta}I\right)^{-1}\Phi^T\mathbf{t}}$$

Remark

- Of the above expression, $\frac{\alpha}{\beta}$ corresponds to the regularization coefficient λ we used in previous regularized least squares.
- **Priors**: Represent prior beliefs about acceptable values for model parameters.
- Example: In linear regression, ℓ^2 regularization can be interpreted as placing a Gaussian Prior on the regression coefficients.
- All statistical models and machine learning algorithms make assumptions.
  - All reasoning is based on implicit assumptions.
  - A Bayesian will tell you that his prior is a way of explicitly stating those assumptions.
- This can all get very philosophical, but...
  - Bayesian reasoning is best seen as a useful tool.
  - Many concepts in machine learning have Bayesian interpretations.
    - Choice of loss / error function, regularization, etc.

- For a fully Bayesian take on machine learning, check out the **Murphy** textbook:



- We will cover more about Bayesian reasoning when we move to **Bayesian Linear Regression**, a linear regression that is used for streaming data.