Linear Models for Regression

Key concepts:

- Sum of squared error for regression loss
- Gradient descent for loss minimization
- Closed form solution for minimizing SSE
- Maximum likelihood estimation and SSE’s probabilistic interpretation
- Overfitting and regularization

CS534

Example Regression Problems

- Predict housing price based on
  - House size, lot size, Location, # of rooms …
- Predict stock price based on
  - Price history of the past month …
- Predict the abundance of a species based on
  - Environmental conditions, shrubs? trees? …

A Basic Set Up

Given: a training set, containing a total of N training examples $(\mathbf{x}_i, y_i)$, for $i = 1, \dots, N$

Goal: learn a function $\hat{y}$ to minimize some loss function on the training data

To begin, we consider a linear hypothesis space (thus the name linear regression):

Let $\mathbf{x} = [1, x_1, x_2, \dots, x_d]^T$

$$\hat{y}(\mathbf{x}) = w_0 + w_1 x_1 + \cdots + w_d x_d = \mathbf{w}^T \mathbf{x}$$

where $\mathbf{w} = [w_0, w_1, \dots, w_d]^T$

A Popular Loss Function: Sum-Squared Error

- Given a set of training examples

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \ldots, (\mathbf{x}_N, y_N)$$

- Hypothesis space (representation):

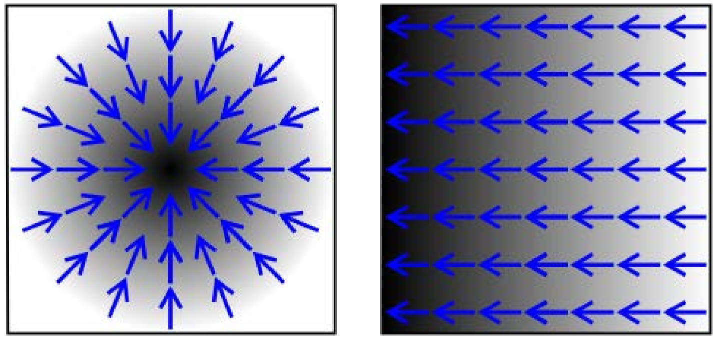
$$\hat{y}(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$$

- Loss function (objective):

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{N} (\mathbf{w}^T \mathbf{x}_i - y_i)^2$$

Optimization of the loss function

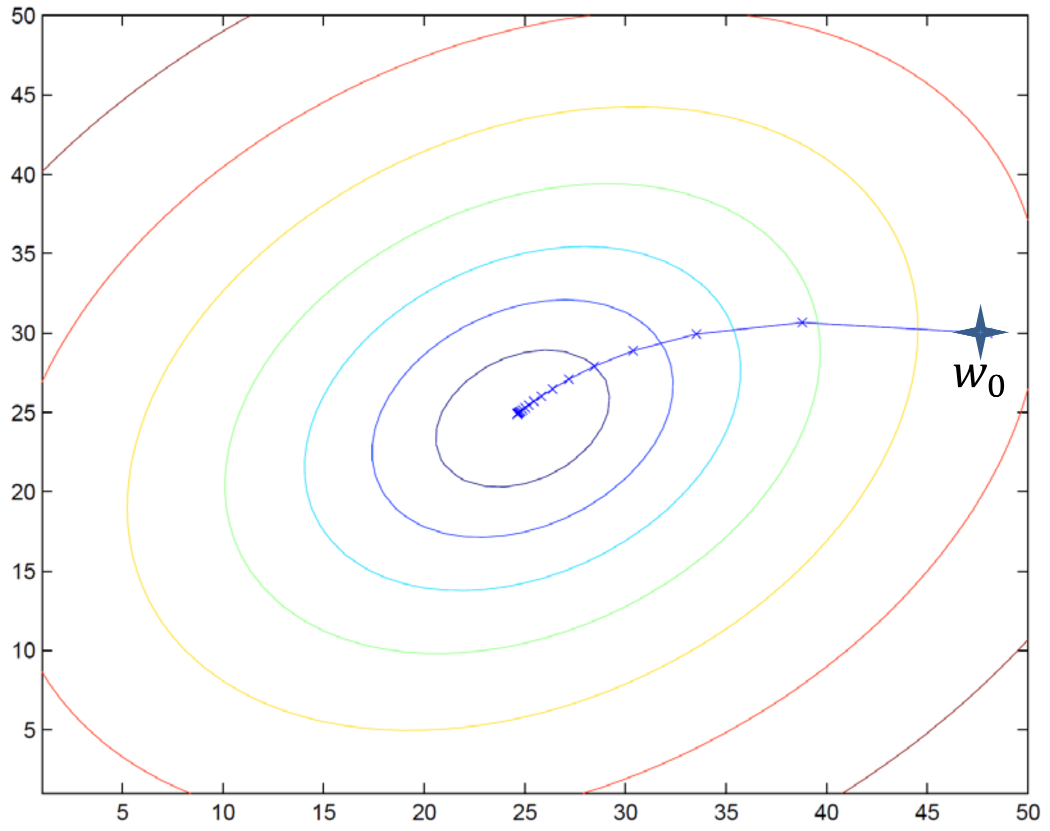
- Many ways to optimize this objective function
- A simple approach is **gradient descent**
  - Start with some random guess of the parameter
  - Iteratively improve the parameter by following the **steepest descent direction**
- **Gradient**: multivariate generalization of derivative, points in the direction of greatest rate of increase of the function



From Wikipedia

Values of the function in black and white, black representing higher values
Gradient represented by blue arrows.

Gradient descent to minimize $f(\mathbf{w})$



1. Start from some initial guess $\mathbf{w}^0$

2. Find the direction of steepest descent – opposite of the gradient direction $\nabla f(\mathbf{w})$

3. Take a step toward that direction (step size controlled by λ)

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \lambda \nabla f(\mathbf{w}^t)$$

4. Repeat until no local improvement is possible
   $(\,|\nabla f(\mathbf{w}^t)| \le \epsilon)$

Figure shows the contours of a quadratic function. Gradient descent starts from w_0, and gradually moves toward the optimal solution.

From derivative to gradient

Derivative with examples:

$$f(x) = x^k; \quad \frac{\partial f}{\partial x} = kx^{k-1}$$

$$f(x) = e^x + 2x^2; \quad \frac{\partial f}{\partial x} = e^x + 4x$$

Generalizing to multivariate function:

For $f(\mathbf{x})$ where $\mathbf{x} = [x_1, \dots, x_d]^T$

Gradient: $\nabla f(\mathbf{x}) = \left[\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_d}\right]^T$

Gradient Descent for SSE

$$\boxed{E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(\mathbf{w}^T\mathbf{x}_i - y_i)^2}$$

We need to compute the gradient of $E(\mathbf{w})$:

$$\frac{\partial E}{\partial w_k} = \sum_{i=1}^{N}(\mathbf{w}^T\mathbf{x}_i - y_i)x_{ik}$$

$$\nabla E(\mathbf{w}) = \sum_{i=1}^{N}(\mathbf{w}^T\mathbf{x}_i - y_i)\mathbf{x}_\mathrm{i} = \sum_{i=1}^{N}(\widehat{y}_i - y_i)\mathbf{x}_\mathrm{i}$$

Batch Gradient Descent for SSE

```
$\mathbf{w} = \mathbf{w}^0$
Repeat{
        $\nabla E(\mathbf{w}) = \sum_{i=1}^{N} (\mathbf{w}^T \mathbf{x}_i - y_i)\mathbf{x}_\mathrm{i}$
        $\mathbf{w} \leftarrow \mathbf{w} - \lambda \nabla E(\mathbf{w})$
} until $|\nabla E(\mathbf{w})| \leq \epsilon$
```

- The update rule goes through all training examples before taking each update step – thus called batch gradient descent
  - Examples with larger error $(\mathbf{w}^T \mathbf{x}_i - y_i)$ contribute more to the update
- λ: learning rate or step size
  - Large λ makes faster progress, but could lead to oscillation
  - Small λ avoids overstepping, but converges slower
- ϵ: convergence criterion
- The SSE Objective is convex – converge to global optimal

Stochastic gradient descent

```
$\mathbf{w} = \mathbf{w}^0$
Repeat until convergence {
    for $i = 1\ to\ N$    $\mathbf{w} \leftarrow \mathbf{w} - \lambda(\mathbf{w}^T\mathbf{x}_i - y_i)\mathbf{x}_\mathrm{i}$
}
```

- It repeatedly runs through the training examples
- It updates $\mathbf{w}$ immediately when seeing an example, based on the gradient w.r.t. that training instance
- Comparing batch vs. stochastic gradient descent:
  - Batch gradient descent needs to go through all training examples for each update step – not appropriate for large datasets
  - Each step of Stochastic gradient descent can see objective oscillating
  - For large datasets, Stochastic gradient descent approaches converge much faster than batch

Alternative way to optimizing $E(\mathbf{w})$

- By setting the gradient to zero, we can directly find the extreme point of $E(\mathbf{w})$
- Define:

$$\boldsymbol{X} = \begin{bmatrix} x_{10} & x_{11} & \cdots & x_{1d} \\ x_{20} & x_{21} & \cdots & x_{2d} \\ \cdots & \cdots & \cdots & \cdots \\ x_{N0} & x_{N1} & \cdots & x_{Nd} \end{bmatrix}, \qquad Y = \begin{bmatrix} y_1 \\ \cdots \\ y_N \end{bmatrix}$$

We can show that

$$\nabla E(\mathbf{w}) = \boldsymbol{X}^{\mathrm{T}}(\boldsymbol{X}\mathbf{w} - Y)$$

Setting it to zero:

$$\Rightarrow \quad \boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\mathbf{w} - \boldsymbol{X}^{\mathrm{T}}Y = 0$$

$$\Rightarrow \quad \mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{w} = \mathbf{X}^{\mathrm{T}}\boldsymbol{Y} \quad \Rightarrow \quad \mathbf{w} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}Y$$

Normal equation

Probabilistic Interpretation of SSE

- One might ask, why is SSE a good objective?
- There is a natural probabilistic explanation for SSE with some simple probabilistic assumptions
- Before that, we will provide a brief of maximum likelihood estimation

Background: Maximum Likelihood Estimation

Parameter Estimation

- Given observation of some random variable(s), assuming the variable(s) follow some fixed distribution, how to estimate the parameters?

- Example:
  - coin tosses (Bernoulli distribution, parameter: *p – probability of head*)
  - Dice or grades (Discrete distribution, parameters: p_i for $i = 1, \ldots, k-1$ with k categories)
  - Height and weight of a person (continuous variable, parameters depends on the distribution, for example for Gaussian Distribution, the parametes are μ, the mean and Σ, the covariance

Maximum Likelihood Principle

- We will use a general term M to denote the model (parameters) we try to estimate
- We will use a general term D to denote the data that we observe
  - D is a set of examples, let D_m denote the m-th example in D
- Assuming a fixed model M, what is the probability of observing D?
  - This can be written as $P(D; M)$
- The maximum likelihood principle seeks to find M that maximizes this probability
- This is called the likelihood function
  - $L(M) = P(D; M) = \prod_{m=1}^{n} P(D_m; M)$ --- here we are making the IID assumption, that is examples in D are independently and identically distributed, thus we can use the product
- It is often more convenient to work with the log of $L(M)$
  - $l(M) = \log L(M) = \sum_{m=1}^{n} \log P(D_m; M)$

Example: Coin Toss

- You are given a coin, and want to decide p- the probability of head of this coin
- You toss it for 500 times, and observe heads 242 times
- What is your estimate of p?

$$p = \frac{242}{500} = 0.484$$

- But why? – this is doing maximum likelihood estimation
- In other words, $p = 0.484$ leads to the highest probability of observing 242 heads out of 500 tosses among all possible values of p
- Now let's go over the math to convince ourselves

Example Cont.

- We have $D = \{x_1, \dots, x_n\}$, where x_m is the outcome of the m-th coin toss,
  - 1 means head, and 0 means tail
- Lets write down the likelihood function: $L(p) = P(D; p) = \prod_{m=1}^{n} P(x_m; p)$
- For binary variable, we have a nice compact form to write down its probability mass function $P(x_m; p) = p^{x_m}(1-p)^{1-x_m}$
- So we have

$$L(p) = \prod_{m=1}^{n} p^{x_m}(1-p)^{1-x_m}$$

- Take the log:

$$l(p) = \log L(p) = \sum_{m=1}^{n} \log p^{x_m}(1-p)^{1-x_m}$$

$$= \sum_{m=1}^{n} x_m \log p + \sum_{m=1}^{n} (1-x_m) \log(1-p) = n_1 \log p + n_0 \log(1-p)$$

$\boxed{n_1 = \sum_{m=1}^{n} x_m \text{ and } n_0 = n - n_1}$

Maximizing the likelihood

- We take derivative of $l(p)$:

$$\frac{dl(p)}{dp} = \frac{n_1}{p} - \frac{n_0}{1-p}$$

- Setting it to zero:

$$\frac{n_1}{p} = \frac{n_0}{1-p}$$

- Solving this leads to:

$$p = \frac{n_1}{n}$$

End of background on MLE

Back to Linear Regression

- **Assumption**: target variable y and the input $\mathbf{x}$ are related as follows:

$$y = \mathbf{w}^T\mathbf{x} + \epsilon$$

where $\epsilon \sim N(0, \sigma^2)$, i.e., it is Gaussian noise

- Equivalently we have: $y|\mathbf{x} \sim N(\mathbf{w}^T\mathbf{x}, \sigma^2)$
- Given a set of observed $\mathbf{x}$ and y pairs: $(\mathbf{x}_1, y_1)$, $(\mathbf{x}_2, y_2)$ …, $(\mathbf{x}_N, y_N)$, **independently and identically distributed (IID),** use maximum likelihood estimation to estimate $\mathbf{w}$
- **Likelihood function:** given the inputs $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$, what is the probability of observing target outputs $y_1, y_2, \dots, y_N$?

$$L(\mathbf{w}) = p(y_1, \dots, y_N | \mathbf{x}_1, \dots, \mathbf{x}_N; \mathbf{w}) = \prod_{i=1}^{N} p(y_i | \mathbf{x}_i; \mathbf{w})$$

Maximum likelihood estimation of $\mathbf{w}$

We will now work with log of the likelihood:

$$l(w) = \log L(\mathbf{w}) = \log \prod_{i=1}^{N} p(y_i | \mathbf{x}_i; \mathbf{w})$$

$$= \sum_{i=1}^{N} \log \frac{1}{\sqrt{2\pi}\sigma} \exp \frac{-(y_i - \mathbf{w}^T \mathbf{x}_i)^2}{2\sigma^2}$$

$$= -N \log \sqrt{2\pi}\sigma + \sum_{i=1}^{N} \frac{-(y_i - \mathbf{w}^T \mathbf{x}_i)^2}{2\sigma^2}$$

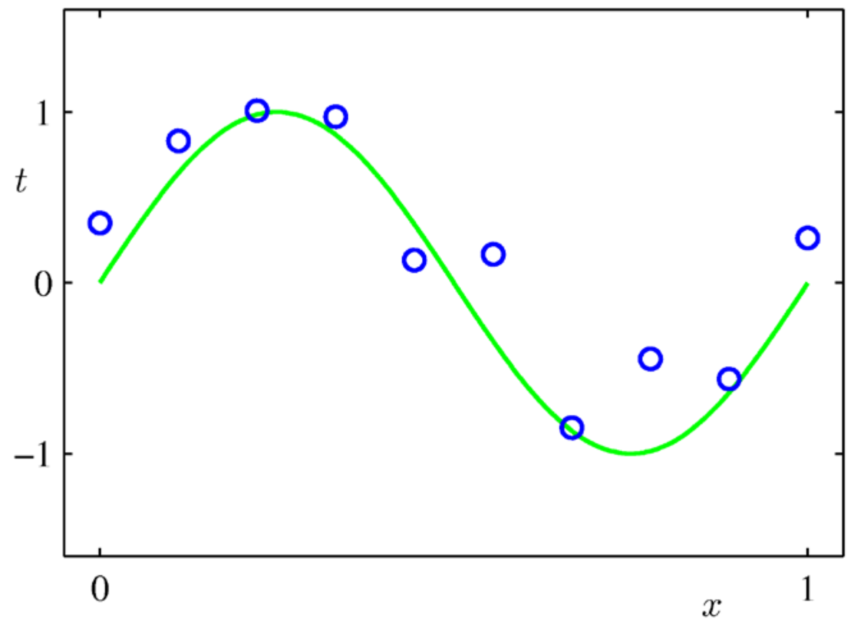
$$\arg\max_{\mathbf{w}} l(\mathbf{w}) = \boxed{\arg\min_{\mathbf{w}} \frac{1}{2} \sum_{i=1}^{N} (y_i - \mathbf{w}^T \mathbf{x}_i)^2}$$

Maximum Likelihood Estimation under <u>Gaussian noise</u> leads to Sum of Squared Error objective

What we have seen so far

- We introduced linear regression
  - Goal: learn a linear function $\hat{y}(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$
  - Approach:
    - Choose a loss function: SSE
    - Optimize the loss function (Gradient descent, or analytically solving the optimization problem)
- What if the relationship between $\mathbf{x}$ and y is not linear? For example, quadratic or cubic?
  - Linear regression can still be used
  - All we need is to introduce nonlinear features

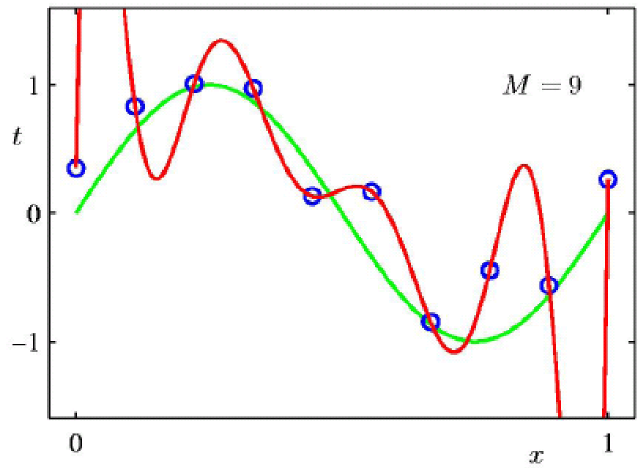
- Revisit example: Polynomial Curve Fitting

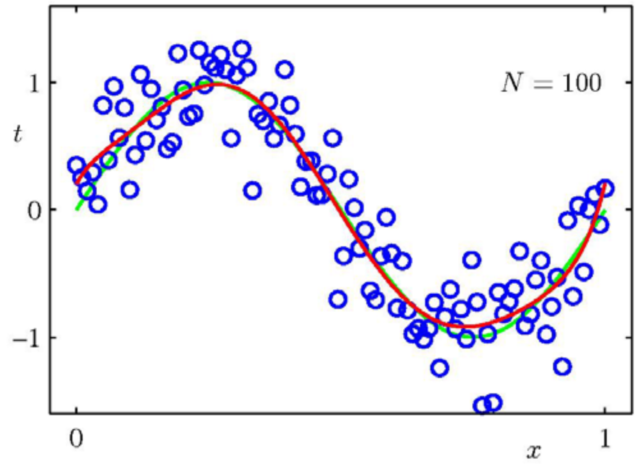


$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

- In this example, there is only one feature x. We learn a function of M-order polynomial
- Alternatively, we could also view this as learning a linear model considering $(1, x, x^2, \ldots, x^M)$ as the features.
- Note that this new feature space is derived from the original input x
- We refer to such derived features as the basis functions

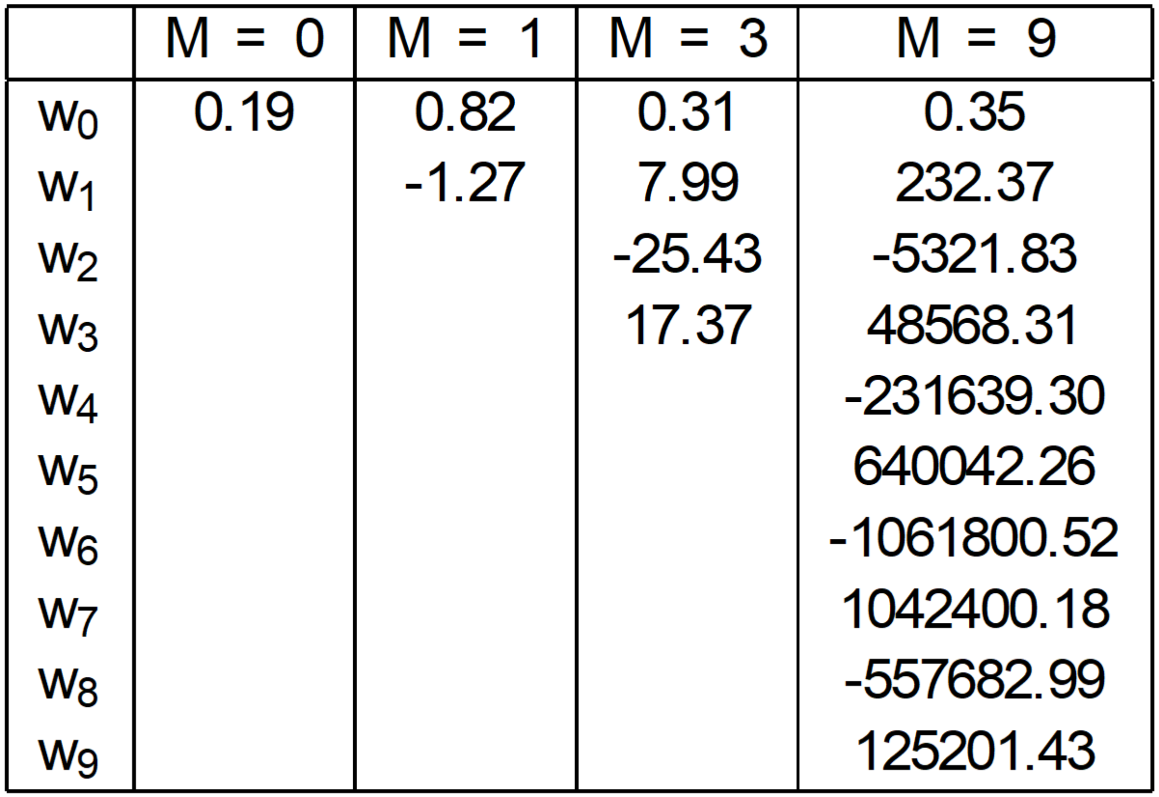
Over-fitting issue





- What can we do to curb over-fitting
  - Use less complex model
  - Use more training examples
  - **Regularization**

In linear regression, overfitting can often be characterized by large weights



	$M = 0$	$M = 1$	$M = 3$	$M = 9$
w_0	0.19	0.82	0.31	0.35
w_1		-1.27	7.99	232.37
w_2			-25.43	-5321.83
w_3			17.37	48568.31
w_4				-231639.30
w_5				640042.26
w_6				-1061800.52
w_7				1042400.18
w_8				-557682.99
w_9				125201.43

Regularized Linear Regression

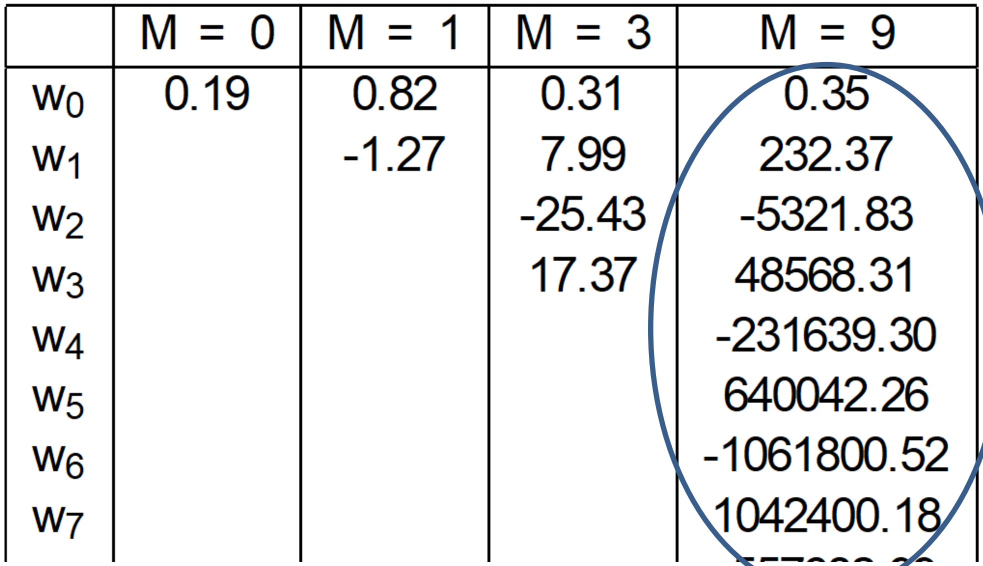
- Consider the following loss function:

$$E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

Data term + Regularization term (penalize complex models)

$$\sum_{i=1}^{N}(y_i - \mathbf{w}^T\mathbf{x}_i)^2 + \lambda \sum_{j=0}^{M}|w_j|^q$$

Encourage small weight values



	M = 0	M = 1	M = 3	M = 9
w_0	0.19	0.82	0.31	0.35
w_1		-1.27	7.99	232.37
w_2			-25.43	-5321.83
w_3			17.37	48568.31
w_4				-231639.30
w_5				640042.26
w_6				-1061800.52
w_7				1042400.18

L2 Regularized Linear Regression

- With the SSE loss and a **quadratic regularizer**, we get

$$\frac{1}{2}\sum_{i=1}^{N}(y_i - \mathbf{w}^T\mathbf{x}_i)^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

- which is minimized by

$$\mathbf{w} = (\lambda \boldsymbol{I} + \mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T Y$$

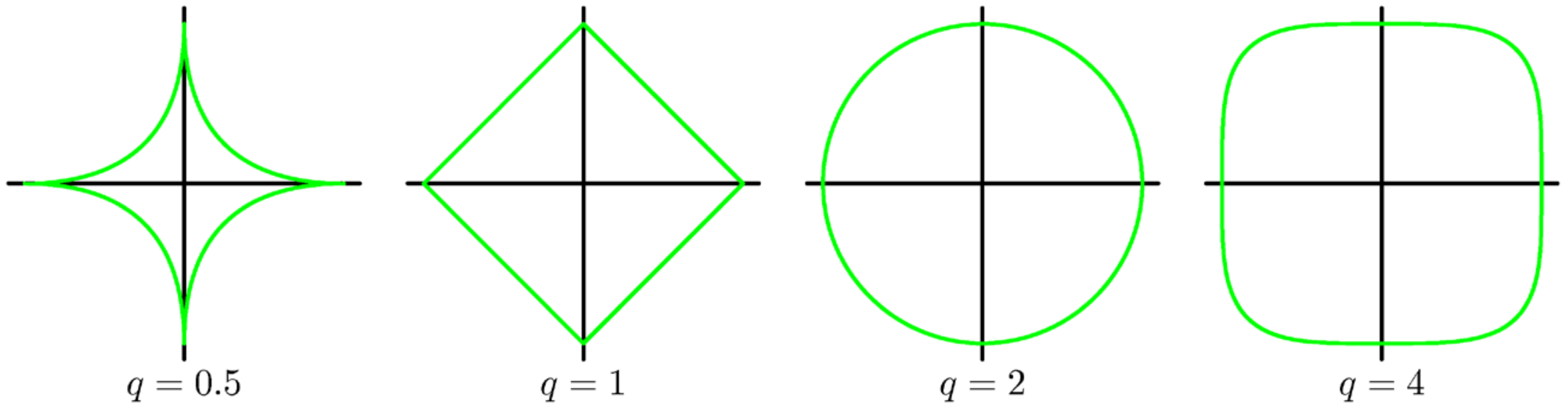
- λ: regularization coefficient, which controls the trade-off between model complexity and the fit to the data
  - Larger λ encourages simple model (driving more elements of $\mathbf{w}$ to 0)
  - Small λ encourages better fit of the data (driving SSE to zero)

More Regularizations

$$\sum_{i=1}^{N}(y_i - \mathbf{w}^T\mathbf{x}_i)^2 + \lambda\sum_{j=0}^{M}|w_j|^q$$

Equivalent to minimizing SSE subject to $\sum_{i=0}^{M}|w_i|^q \leq \epsilon$

A good explanation of this equivalence is provided here:
http://math.stackexchange.com/questions/335306/why-are-additional-constraint-and-penalty-term-equivalent-in-ridge-regression

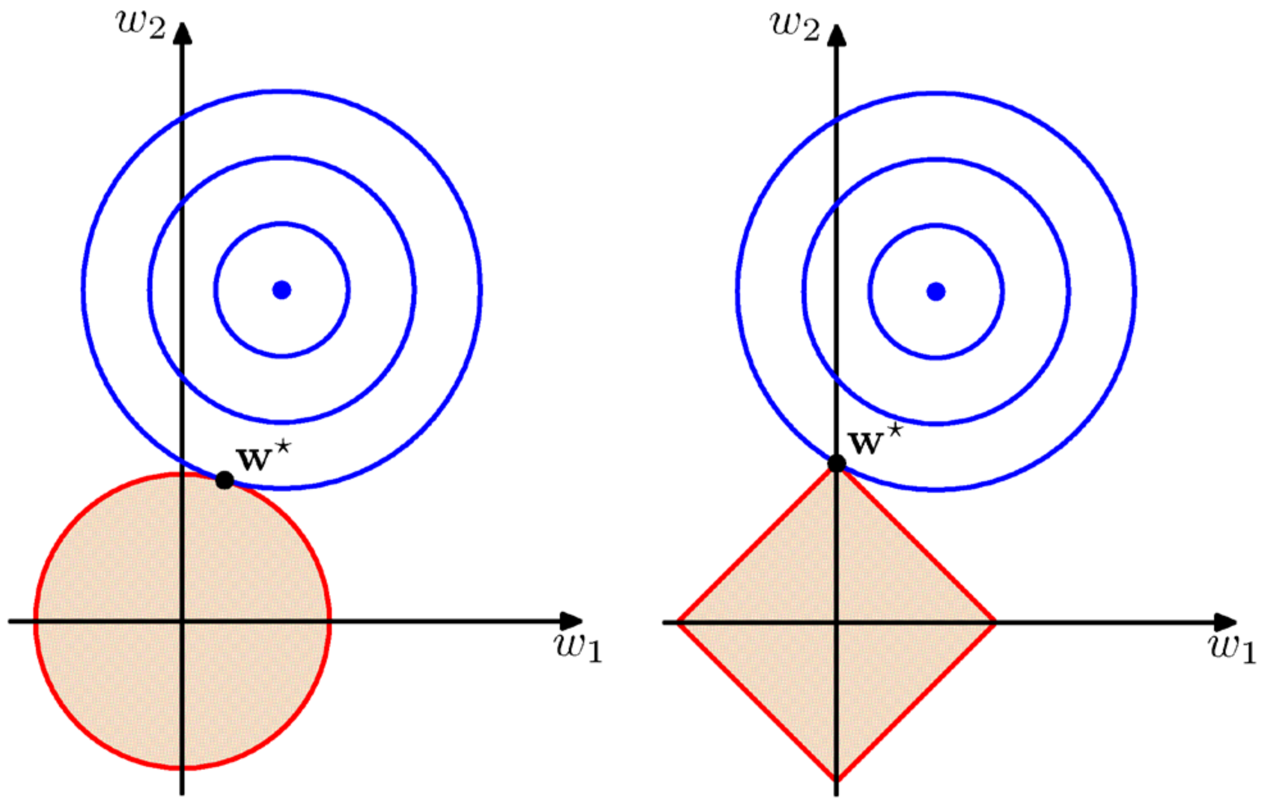


Lasso, L-1 L-2

Shape is determined by q, size determined by λ

Regularized Linear Regression

- Lasso ($q = 1$) tends to generate sparser solutions (majority of the weights shrink to zero) than a quadratic regularizer ($q = 2$, often called ridge regression).



Commonly used regularizors

- L-2 regularization $$\sum_{i=1}^{N}(y_i - \mathbf{w}^T\mathbf{x}_i)^2 + \lambda\sum_{j=0}^{M} {w_j}^2$$

  Poly-time close-form solution
  Curbs overfitting but does not produce sparse solution

- L-1 regularization $$\sum_{i=1}^{N}(y_i - \mathbf{w}^T\mathbf{x}_i)^2 + \sum_{j=0}^{M}|w_j|$$

  Poly-time approximation algorithm
  Sparse solution – potentially many zeros in $\mathbf{w}$

- L-0 regularization $$\sum_{i=1}^{N}(y_i - \mathbf{w}^T\mathbf{x}_i)^2 + \sum_{j=0}^{M} I(w_j \neq 0)$$

  Seek to identify optimal feature subset
  NP-complete problem!

Summary

- Linear regression with SSE objective
  - Gradient descent algorithm for optimization
  - Closed-form solution by solving the normal equation
  - Can be viewed as maximum likelihood estimation assuming IID Gaussian noise with zero mean

- Using basis functions allows us to learn nonlinear functions

- Controlling over-fitting by Regularization
  - Quadratic regularization is equivalent to adding a diagonal matrix to $\mathrm{X}^T\mathrm{X} \rightarrow (\lambda I + \mathrm{X}^T\mathrm{X})$
  - LASSO regularization encourages sparse solution