EECS 545 → Machine Learning

Lecture 3 → Linear Regression (Part 2)

- $E(w) = \frac{1}{2}\sum_{n=1}^{N}\{h(x^{(n)}, w) - y^{(n)}\}^2$
- We want to find w that minimizes E(w) over the training data
- Two methods to solve this
  1. Gradient Descent → $w := w - \eta\nabla_w E(w)$
  2. Closed form solution → $E(w) = \frac{1}{2}w^T\Phi^T\Phi w - w^T\Phi^T y + \frac{1}{2}y^T y$

- Overfitting → when the Root Mean Square Error (RMSE) for the testing data is significantly higher than for the training error

$E_{RMSE} = \sqrt{2E(w^*)/N}$

w^* = optimal parameter set

- Increasing dataset size can help with the issue of overfitting
- If amount of data is small, use small order polynomial
- As data increases, complexity of polynomial can increase, but this must also align with the complexity and the requirements of the problem
- Controlling model complexity is known as regularization
- Having coefficients that are too large can also suggest that we are encountering an overfitting issue

Regularized Least Squares

- New error function → $\underbrace{E_D(w)}_{\text{data term}} + \lambda\underbrace{E_w(w)}_{\text{regularization term}}$
- λ is the regularization coefficient
- Our new objective function can be written as,

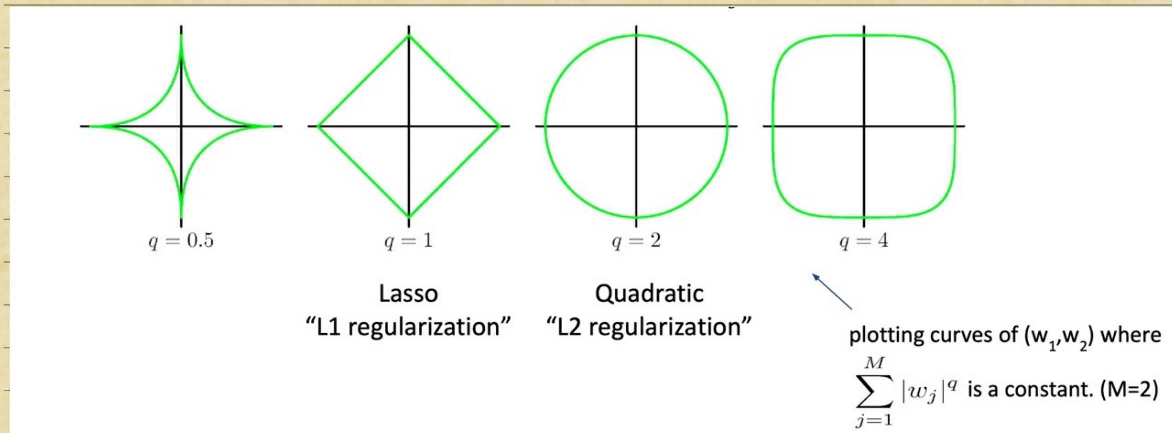
$$\tilde{E}(w) = \frac{1}{2}\sum_{n=1}^{N}(w^T\phi(x^{(n)}) - y^{(n)})^2 + \frac{\lambda}{2}\|w\|_2^2$$

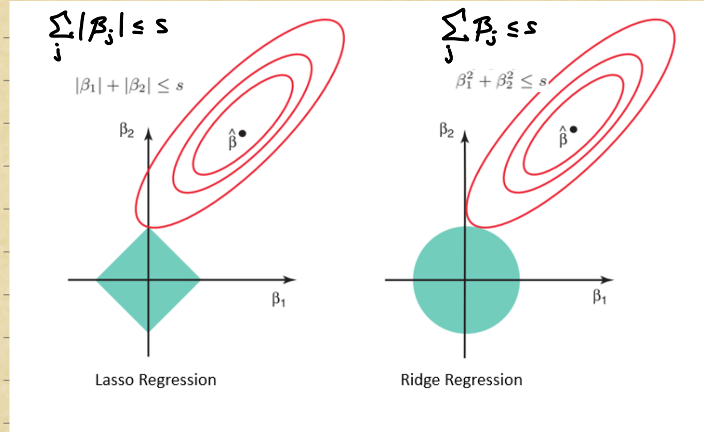
$$L2 \rightarrow \|w\|_2^2 = \sum_{j=0}^{M-1} w_j^2$$

- The goal is to minimize the objective function and so we will penalize larger coefficients
- Choosing the value of λ is important because it defines how much significance we should place on controlling the values of the coefficients
- Closed form solution → $w_{ML} = (\lambda I + \Phi^T\Phi)^{-1}\Phi^T y$

- The L2 regularization is of the form of a more general regularization formula,

$$\frac{1}{2}\sum_{n=1}^{N}(w^T\phi(x^{(n)}) - y^{(n)})^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q$$





- Lasso solutions tend to be sparser i.e. more parameters are reset to exactly zero
- Ridge solutions tend to be closer to zero

- Regularization controls tradeoff between fitting error and complexity
- Small regularization leads to complex models with the possibility of overfitting
- Large regularization leads to simple models with the possibility of underfitting

MLE for Linear Regression

- Stochastic Model → $y^{(n)} = \omega^T \phi(x^{(n)}) + \epsilon$ where $\epsilon \sim N(0, 1/\beta)$
- Likelihood function → $p(y^{(n)} | \phi(x^{(n)}), \omega, \beta) = N(y^{(n)} | \omega^T \phi(x^{(n)}), 1/\beta)$
- Data likelihood → $p(y | \Phi, \omega, \beta) = \prod_{n=1}^{N} N(y^{(n)} | \omega^T \phi(x^{(n)}), 1/\beta)$ input matrix Φ, output matrix y

$$p(y^{(n)} | \phi(x^{(n)}), \omega, \beta) = N(y^{(n)} | \omega^T \phi(x^{(n)}), \tfrac{1}{\beta})$$

$$= \sqrt{\frac{\beta}{2\pi}} \exp\left(-\frac{\beta}{2} ||y^{(n)} - \omega^T \phi(x^{(n)})||^2\right)$$

$$\Rightarrow \log \prod_{n=1}^{N} N(y^{(n)} | \omega^T \phi(x^{(n)}), \tfrac{1}{\beta})$$

$$= \sum_{n=1}^{N} \log\left(\sqrt{\frac{\beta}{2\pi}} \exp\left(\frac{-\beta}{2} ||y^{(n)} - \omega^T \phi(x^{(n)})||^2\right)\right)$$

$$= \sum_{n=1}^{N} \left(\frac{1}{2} \log \beta - \frac{1}{2} \log 2\pi - \frac{\beta}{2} ||y^{(n)} - \omega^T \phi(x^{(n)})||^2\right)$$

$$= \frac{N}{2} \log \beta - \frac{N}{2} \log 2\pi - \sum_{n=1}^{N} \frac{\beta}{2} ||y^{(n)} - \omega^T \phi(x^{(n)})||^2$$

- We maximize the log likelihood and set the gradient equal to 0

$$\nabla_\omega \log(p | \Phi, \omega, \beta) = \nabla_\omega \left(\frac{N}{2} \log \beta - \frac{N}{2} \log 2\pi - \sum_{n=1}^{N} \frac{\beta}{2} ||y^{(n)} - \omega^T \phi(x^{(n)})||^2\right)$$

$$= \beta \sum_{n=1}^{N} (y^{(n)} - \omega^T \phi(x^{(n)}) \phi(x^{(n)}))$$

$$= \beta \left(\sum_{n=1}^{N} y^{(n)} \phi(x^{(n)}) - \phi(x^{(n)}) \phi(x^{(n)})^T \omega\right) = 0$$

- In matrix form, $\beta(\Phi^T y - \Phi^T \Phi \omega) = 0$, $\omega_{ML} = (\Phi^T \Phi)^{-1} \Phi^T y$
- MLE solution is equivalent to OLS solution

Locally Weighted Linear Regression

- When predicting $f(\hat{x})$, we give high weights for neighbors of $\hat{x}$
- Points are weighted by proximity to current $\hat{x}$ in question using a kernel
  - Regression is then computed using weighted points

- Locally weighted linear regression requires two elements, a query point $\hat{x}$ and training set $\{(x^{(n)}, y^{(n)})\}_{n=1}^{N}$

① Fit ω to minimize $\sum_{n=1}^{N} r^{(n)}(\hat{x})(\omega^T \phi(x^{(n)}) - y^{(n)})^2$

② Predict $\omega^T \phi(\hat{x})$

· Standard choice → $r^{(n)}(\hat{x}) = \exp\left(-\frac{\|\phi(x^{(n)}) - \phi(\hat{x})\|^2}{2T^2}\right)$

· N.B. $r^{(n)}(\hat{x})$ depends on $\hat{x}$ and you solve linear regression problem for each query point $\hat{x}$

· Choice of T requires hyperparameter tuning