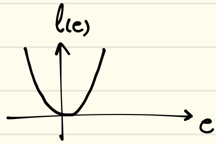
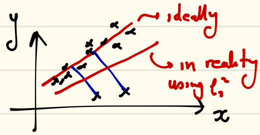
DS4400
Ehsan Elhamifar

Robust Regression



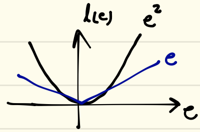
One point about using ℓ_2^2 cost function is its high intolerance for large errors $\quad (\underbrace{y-h_\theta(x)}_{e})^2 = \ell(\underbrace{y-h_\theta(x)}_{e}) = e^2$



What if the dataset contains "outliers" ?

ℓ_2^2 prefers to have "many smaller errors" than "a few large errors" !

Solution: use a more robust cost function to deal with outliers



Let $\ell(e) = |e|$ (absolute value or ℓ_1-norm)

$$\min_\theta \sum_{i=1}^{N} |y_i - h_\theta(x_i)| = \sum_{i=1}^{N} |y_i - \theta^T x_i| = \|Y - X\theta\|_1$$

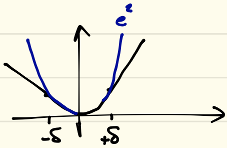
Still convex!

challenge: $|e|$ is non-diff at its minimum $\quad \frac{\partial |e|}{\partial e} = \begin{cases} +1, & e>0 \\ -1, & e<0 \\ \text{undefined}, & e=0 \end{cases}$

unlike ℓ_2^2 not easy to optimize (no closed-form, no grad desc), Yet convex!

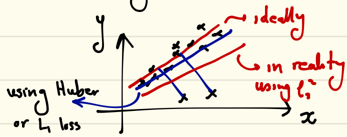
Another objective function that has the advantages of both norms robustness of $|e|$ and differentiability of e^2 is called "Huber loss"

$$\ell_\delta(e) \triangleq \begin{cases} e^2/2 & , |e| \leq \delta \\ \delta|e| - \frac{\delta^2}{2} & , |e| > \delta \end{cases}$$



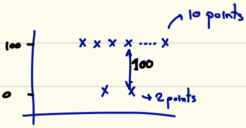
Convex, $\in C^1$ (has a continuous grad) is differentiable everywhere!

$\Longrightarrow$ Does it have a closed-form solution ?



To optimize, we can use batch / stochastic gradient descent !

Example :



DS4400
Ehsan Elhamifar

Overfitting

A major problem of learning systems is "overfitting" performing well on training samples but performing poorly on test samples!

$d=2 \Rightarrow h(x)=\theta x+b \Rightarrow \underset{2\times 2}{X^TX}\,\underset{2\times 1}{\theta} = \underset{2\times 1}{X^TY} \xrightarrow{N\geq d} \theta \checkmark$

$d=3 \Rightarrow h(x)=\theta_1 x+\theta_2 x^2+b \Rightarrow X^TX\theta = X^TY \xrightarrow{N\geq d} \theta \checkmark$

$d=5 \Rightarrow h(x)=\theta_1 x+\theta_2 x^2+\theta_3 x^3+\theta_4 x^4+b \xrightarrow{N<d} \underset{5\times 5}{X^TX}\,\underset{5\times 1}{\theta} = \underset{5\times 1}{X^TY}$, $rk(X^TX)\leq 4$

$\hookrightarrow X^TX v=0 \;\; \exists v \longrightarrow X^TX(\theta^*+\alpha v) = X^TX\theta^* = X^TY$

(Can make $\alpha\to\infty$!)

$\Downarrow$

large values θ

Treatment: Regularization

$$\min_{\theta} \sum_{i=1}^{N} \ell(y_i-\theta^T x_i) + \overset{\text{reg parameter} > 0}{\lambda}\, \underbrace{f(\theta)}_{\text{regularization term}} .$$

$f(\theta)=\|\theta\|_2^2$, $\ell(e)=e^2 \longrightarrow$ Ridge regression

$$\min_{\theta} \|Y-X\theta\|_2^2 + \overset{\text{weight decay}}{\lambda \|\theta\|_2^2} \triangleq J(\theta)$$

$J(\cdot)$ Convex & diff wrt θ

$$\frac{\partial J}{\partial \theta} = 2X^T(X\theta-Y)+2\lambda\theta$$

(1) $\nabla J(\theta)\big|_{\hat{\theta}}=0 \longrightarrow (X^TX+\lambda I)\,\theta = X^TY \longrightarrow \theta=(X^TX+\lambda I)^{-1}X^TY$

Notice that $\lambda I + X^TX$ always invertible for $\lambda>0$.

$\longrightarrow$ it is always invertible even when $N<d$!!

(2) Batch / stochastic GD

- parameter : learned during training : e.g., θ
- hyperparameter : not learned during training, set before, e.g., λ [if we want to learn λ to minimize training then we always pick $\lambda = 0$]

λ hyper parameter — not learned by the learning alg, is set in advance
(however, in a full Bayesian method, has a prob for it)

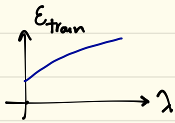
Effect of λ — $\lambda \to 0$ same as LS, overfitting if $d > N$

$\lambda \to \infty$ $\hat{\theta} \simeq 0$ $\left((X^T X + \lambda I)^{-1} X^T Y \simeq 0 \right) \to$ No overfit but high training & test error

Question How to set λ ?

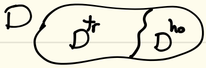
Enough to give enough expressive power to model to obtain small training and test error (even $\lambda = 10^{-4}$ is better than nothing)

+ Naive way
  - For each λ, solve $\min_{\theta} \sum_{i=1}^{N} \ell(y_i - \theta^T x_i) + \lambda f(\theta) \longrightarrow \hat{\theta}_\lambda$
  - Compute $\mathcal{E}_{train}(\lambda) = \sum_{i=1}^{N} \ell(y_i - \hat{\theta}_\lambda^T x_i)$
  - take $\lambda^* = \arg\min_{\lambda} \mathcal{E}_{train}(\lambda)$

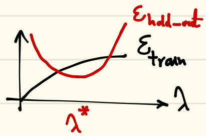


$\Rightarrow$ $\lambda^* = 0$ overfitting !!

* Right approach "Hold out" set set aside part of the training data as hold out set



+ For each λ, compute $\hat{\theta}_\lambda$ using D^{tr}
+ " " ", compute $\mathcal{E}_{holdout}(\lambda) = \sum_{(x_i, y_i) \in D^{ho}} \ell(y_i - \hat{\theta}_\lambda^T x_i)$
  - ⇒ This can be a good measure of generalization error, i.e., error on unseen data
+ take $\lambda^* = \arg\min_\lambda \mathcal{E}_{hold\text{-}out}(\lambda)$

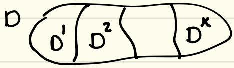


+ run regularized regression on test data using λ^*

→ This is how we can effectively deal with generalization and overfitting

Challenge if training dataset is small, hold-out set makes training set smaller → not good $\hat{\theta}_\lambda$ due to more overfitting!

Solution K-fold Cross Validation divide training set into K partitions, run training on all but one partition, evaluate on the remaining parti.



- For $\ell = 1, \ldots, K$
  - find $\hat{\theta}_\lambda$ using $\bigcup_{j \neq \ell} D^j$, for several values of λ
  - compute $\mathcal{E}^{(\ell)}_{hold\text{-}out}(\lambda)$ using $D^{(\ell)}$
- Compute the average hold-out error as

$$\mathcal{E}_{hold\text{-}out}(\lambda) = \frac{1}{K} \sum_{j=1}^{K} \mathcal{E}^{(j)}_{hold\text{-}out}(\lambda)$$

- take $\lambda^* = \arg\min_\lambda \mathcal{E}_{hold\text{-}out}(\lambda)$
- test on test samples using λ^*

* If $K = N$ (training on all but one data at a time), this is called One hold out CV! → Expensive, but effective when N small