# CSE 446 Notes

Residual Sum of Squares (RSS): $\mathrm{RSS}(\mathrm{w}) = (\mathrm{y} - \mathrm{Xw})^{\mathrm{T}}(y - Xw)$, $\widehat{w_{LS}} = (X^T X)^{-1} X^T y$ (Least squares optimal solution)
Handling intercept: (1) Show that $\frac{1}{\mathrm{n}}\sum_{\mathrm{i}} x_i = 0 (*)$ (2) "Demean" linear regression model so * holds
Interpreting the result: When other features are fixed, the estimated change in $\hat{y}$ for one-unit change of x
Reason for minimizing RSS: if $\mathrm{y_i} = \mathrm{wx_i} + \epsilon_{\mathrm{i}}$ and $\mathrm{e} \sim \mathrm{N}(0, \sigma^2)$ which means $\mathrm{y} \sim (\mathrm{x^T w}, \sigma^2)$, the MLE for w produces minimum RSS.
Feature map: in polynomial regression, transform each data point $\mathrm{x_i}$ to higher-dimensional feature vector $\mathrm{h}(\mathrm{x_i})$
$\mathrm{h_j}(x)$: j[th] feature associated with input x (j[th] basis function)
Supervised Learning: learning a function that maps inputs to outputs based on labelled examples
For linear regression, the loss function is RSS.
Handling an intercept in linear regression: model is $\mathrm{y} = \mathrm{wx} + \mathrm{b}$, $\hat{b} = \frac{1}{n}\sum_{i=1}^{n} y_i$, w still the same.
In the case of feature map, replace x with h
Training Error: $\mathrm{e_{train}} = \frac{1}{n}\sum_{\mathrm{i=1}}^{\mathrm{n}} loss(y_i, \hat{f}_w(x_i))$, dependent on how loss function is defined. Training error is overly optimistic on training data (unless training set includes everything you might ever see).
Generalization error: $\int_{\mathrm{x,y}} \big(y - f_w(x)\big)^2 P(x,y)dxdy$
As model complexity goes up, training error goes down; generalization error goes down, then goes up.
Testing Error: same as training error, but on a labelled testing data set.

Three sources of error:
Data inherited noise: $\mathrm{y} = \mathrm{wx} + \epsilon$, irreducible.
Bias: Let $\mathrm{f}_{\bar{w}}(x) = E_{train}[f_{w_{train}}(x)]$, then $\mathrm{Bias_x} = f_{w_{true}}(x_i) - f_{\bar{w}}(x_i)$ -> is our model flexible enough to catch true w?
Variance: How much do specific fit varies from expected fit? $\mathrm{Var_x} = E_{train}\left[\left(f_{\widehat{w_{train}}}(x_i) - f_{\bar{w}}(x_i)\right)^2\right]$
Low complexity: high bias, low variance High complexity: high variance, low bias
If $\mathrm{y} = \mathrm{wX} + \epsilon$ and $\epsilon \sim \mathrm{N}(0, \sigma)$, average $\mathrm{prediction\ error_i} = \sigma^2 + Bias_i^2 + \mathrm{Variance_i}$
$\mathrm{Mean\ Squared\ Error\ (MSE)} = E_{train}\left[\left(f_{w_{true}}(x_t) - f_{\hat{w}_{train}}(x_t)\right)^2\right] = Bias^2 + Var$

Regularization: imposes "simpler" solutions by a complexity penalty
Rigid Regression: $\mathrm{L}_2$ penalized least-squares regression. Trade of model complexity with training error
$\widehat{w}_{ridge} = argmin_w \left(\sum_{i=1}^{n}\left(y_i - x_i^T w\right)^2\right) + \lambda\left|\left|w\right|\right|_2^2$, solves to $\widehat{w_{ridge}} = (X^T X + \lambda I)^{-1} X^T y$
Shrinkage Property: assume $\mathrm{X^T}X = I$(a special case), $\widehat{w_{ridge}} = \frac{n}{n+\lambda}\,\widehat{w_{LS}}$
Assume $\mathrm{X^T}X = nI$ and $\mathrm{y} = \mathrm{Xw} + \epsilon$, $\epsilon \sim \mathrm{N}(0, \sigma^2)$, we have $\widehat{w}_{\mathrm{rigid}} = \frac{n}{n+\lambda} w + \frac{1}{n+\lambda} X^T \epsilon$

$$\mathrm{predicted\ everage\ error} = \sigma^2 + \frac{\lambda^2}{(n+\lambda)^2}(w^T x)^2 + \frac{d\sigma^2 n}{(n+\lambda)^2}\left|\left|x\right|\right|_2^2$$

$\lim_{\lambda \to 0} \widehat{w}_{ridge} = \widehat{w}_{LS}$, $\lim_{\lambda \to \infty} \widehat{w}_{ridge} = 0$.

Leave-One-Out (LOO) cross validation: D\j is the set of training data with data point j removed

$$\mathrm{error_{LOO}} = \frac{1}{\mathrm{n}}\sum_{j=1}^{n}\left(y_i - f_{D/j}(x_i)\right)^2$$

$\mathrm{error_{LOO}}$ is usually pessimistic because learning with less data points gives worse answer.
LOO is almost unbiased, so it can be used for model selection (i.e. picking the right $\lambda$ in regularization).
k-fold cross validation: Divide data to k equal parts $\mathrm{D}_1, D_2, D_3, \ldots, D_k$
Estimation error on set $\mathrm{D_i}$: $\mathrm{error_{D_i}} = \frac{1}{|D_i|}\sum_{x_i, y_i \in D_i}\left(y_i - f_{D/D_i}(x_i)\right)^2$
k-fold error: average over all sets $\mathrm{error_{k-fold}} = \frac{1}{k}\sum_{i=1}^{k} error_{D_i}$
Faster to compute and much more pessimistic than LOO (because model is trained with even less data points)
Usually choose $\mathrm{k} = 10$

Given a set of input data: (1) split to train and test set (2) use k-fold cross validation on training set to train predictor and choose magic parameters such as $\lambda$ (3) Assessment: use test set to check the accuracy of the model

Greedy Model/Feature Selection: (1) Start from an empty set of features $\mathrm{F}_0 = \{\}$ (2) Run learning algorithm on current feature set to get weight for features in $\mathrm{F}_0$ (3) Choose the best next feature $\mathrm{h_i}$ which minimizes the training error when using $\{\mathrm{F}_0, \mathrm{h_i}\}$ (4) Recuse from step (1)
Theorem for Penalized Least Squares: For any $\lambda \geq 0$ for which $\widehat{w}_\mathrm{r}$ achieves the minimum, there exists a $\mathrm{v} \geq 0$ such that $\widehat{w}_\mathrm{r} = argmin_w \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2$ subject to $\mathrm{r(w)} \leq \mathrm{v}$
LASSO Regression: $\widehat{w}_{ridge} = argmin_w \left(\sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2\right) + \lambda \left|\left|w\right|\right|_1$, keep $\sum_{\mathrm{i}=1}^{\mathrm{n}} y_i = 0$ and $\sum_{\mathrm{i}=1}^{\mathrm{n}} x_i = 0$ for not dealing with offset terms.
Optimizing LASSO objective one coordinate at a time: For $\mathrm{j} = 1, 2, \ldots, \mathrm{n}$, $\mathrm{r_i^j} = \sum_{k \neq j} x_{ij} w_k$ and $\widehat{w}_\mathrm{j} = argmin_{w_j} \sum_{i=1}^{n} \left(\mathrm{r_i^j} - x_{i,j} w_j\right)^2 + \lambda |w_j|$
Subgradient: $\mathrm{f(y)} \geq \mathrm{f(x)} + \mathrm{g^T}(y - x)$, w is a minimum if 0 is a subgradient at w
Coordinate Descent for LASSO (Shooting Algorithm): Pick a random coordinate j, compute $\mathrm{a_j} = \sum_{i=1}^{n} x_{ij}^2$, $\mathrm{c_j} = 2 \sum_{i=1}^{n} \left(y_i - \sum_{k \neq j} x_{i,k} w_k\right) x_{i,j}$
$\mathrm{L_1}$ Regularization is one way to do variable selection (finding a sparse solution). LASSO is non-differentiable, but convex

Binary Classification: $\mathrm{loss(f(x), y)} = 1\{\mathrm{f(x)} \neq \mathrm{y}\}$
Sigmond for Binary Class: $\mathrm{P(Y = 0|w, X)} = \frac{1}{1 + \exp(w_0 + \sum_k w\_k x_k)}$
Linear Decision Rule: $\log\left(\frac{\mathrm{P(Y = 1|w, X)}}{\mathrm{P(Y = 0|w, X)}}\right) = \mathrm{w}_0 + \sum_k w\_k X_k$
Conditional Likelihood: if $\mathrm{y_i} \in \{-1, 1\}$, $\mathrm{P(Y = y|x, w)} = \frac{1}{1 + \exp(-y w^T x)}$, $\widehat{w}_{MLE} = argmin_w \sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i x_i^T w\right)\right) = J(\mathrm{w})$, $\mathrm{J(w)}$ is a convex function, no closed-form solution
To prevent overfitting: add a regularization term (i.e. $\mathrm{L_2}$)

Gradient Descent: $\mathrm{w_{t+1}} = w_t - \alpha \nabla w_t$
With Taylor Series Approximation: $\mathrm{w_{t+1}} = w_t - \alpha \nabla f(w_t)$
For $\mathrm{f(w)} = \frac{1}{2} \left|\left|X_w - y\right|\right|_2^2$, $\nabla \mathrm{f(w_t)} = (X^T X)^{-1} X^T y$, $(\mathrm{w_{t+1}} - \mathrm{w}_*) = \left(\mathrm{I} - \alpha \mathrm{X^T X}\right)(\mathrm{w^t} - \mathrm{w}_*) = \left(\mathrm{I} - \alpha \mathrm{X^T X}\right)^{\mathrm{t+1}} (w_0 - w_*)$
Total Derivative: $\sum_{\{\mathrm{i}=1\}}^{\mathrm{n}} \frac{\delta l_i(w)}{\delta w_j} + 2\lambda w_j$

Stochastic Gradient Descent: Use one data point to estimate the actual averaged gradient over all coordinates
Each data point contributes to $\frac{1}{\mathrm{N}}$ of regularization.
Estimated Derivative: $\frac{\delta l_i(w)}{\delta w_j} + \frac{2\lambda w_j}{N}$ for some random data point i
SGD has noisy convergence.