ML & Optim Lect 7

- Practical issues:
  - train/test/validation splits
  - cross-validation when we don't have enough data for fixed splits
- Regularization

  (ex: ridge regression, LASSO, regularized logistic regression)
- Risk decomposition: how well do we need to optimize?
- Convex sets & convex optimization

Practical considerations

train vs test vs validation splits

↓

use to solve our optim prob to get $\hat{f}$ that approximately solves

$$\underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n_{train}} \sum_{i=1}^{n_{train}} \ell(f(x_i), y_i)$$

evaluate the fitted model

$$R(f) \approx \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} \ell(f(x_i), y_i)$$

hyperparameter selection (e.g. how many iterations, which optimization algorithm, which model class)

evaluate the models w/ different hyperparameters

$$R(f) \approx \frac{1}{n_{val}} \sum_{i=1}^{n_{val}} \ell(f(x_i), y_i)$$

and choose model w/ lowest empirical risk

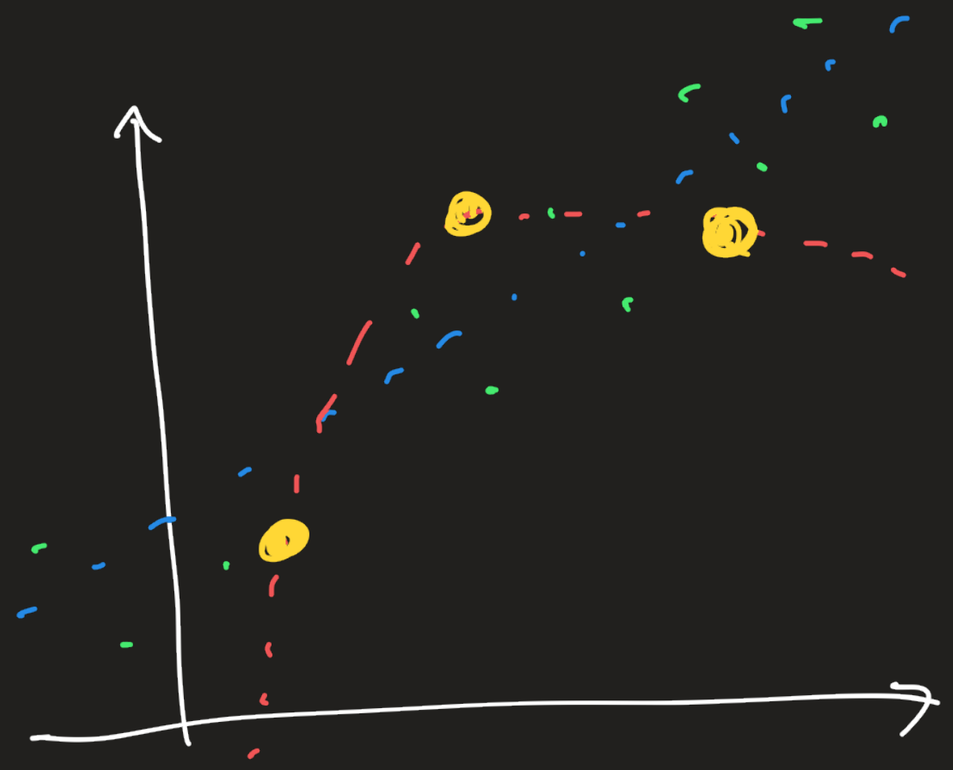
General flow for ML (supervised, ERM)

- Fix our problem domain, risk (loss) measure, collect data $\{(x_i, y_i)\}_{i=1}^{n}$
- Split our data into train/validation/test (60/20/20)
- Determine (ahead of time) a set of models and hyperparameters to try
- Fit each of these using the same training data
- Measure their performance using the validation data, and select f_{opt}
- Report an estimate of the population risk of f_{opt} using the test data

Regularization

In ML we often don't have a strong idea of what is the ideal model for our data, so we try a bunch of different models and choose the one that gives best performance. If we pick <u>too</u> powerful/complicated (high capacity) of a class of models, we can <u>overfit</u>:

$$R_n(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n} \ell(\hat{f}(x_i), y_i) \ll R(f^*)$$



population distribution

training data

OLS fit

quadratic fit

For this population dist, we don't have enough training data to fit a model more complicated than linear.

Overfitting occurs when we don't have enough training data from our population distribution to accurately predict the behavior of our model class.

To avoid overfitting:

1) structural risk minimization: Occam's razor, use very simple models when they suffice

2) use <u>regularization</u> penalize the complexity of your model in the fitting objective

Regularized ERM is obtained by adding a regularization term to your objective to encourage models to be simple

Ex: encourage the model to have small coefficients by adding an l_2 regularization term (aka weight decay)

$$w_* = \operatorname*{argmin}_w \frac{1}{n}\|y - Xw\|_2^2 + \lambda \|w\|_2^2$$

This is called ridge regression (OLS + l_2-regularization)

$$w^* = \text{argmin} \ \frac{1}{n}\|y - Xw\|_2^2 + \lambda \|w\|_2^2$$

where $\lambda > 0$ is called the regularization parameter and its value determines the relative importance we assign to fitting our data vs having small weights.

Recall OLS:

$$w_{OLS} = \text{argmin} \ \frac{1}{n}\|y - Xw\|_2^2$$
$$= X^+ y = (X^T X)^{-1} X^T y$$

For ridge regression

$$w_{RR} = (X^T X + \lambda n I)^{-1} X^T y$$

↑ regularization parameter times training set size

← interpolates b/w w_{OLS} when $\lambda = 0$ and $w_{RR} \to 0$ when $\lambda \to \infty$

Aside:

$$\omega_{OLS} = \underset{\omega}{\operatorname{argmin}} \; \frac{1}{n} \|y - X\omega\|_2^2$$

$$= \underset{\omega}{\operatorname{argmin}} \; (y - X\omega)^T (y - X\omega)$$

$$= \underset{\omega}{\operatorname{argmin}} \; y^T y - \omega^T X^T y - y^T X\omega + \omega^T X^T X\omega$$

$$= \underset{\omega}{\operatorname{argmin}} \; \omega^T X^T X\omega - 2\omega^T X^T y$$

this function is minimized when its gradient w.r.t. $\omega = 0$

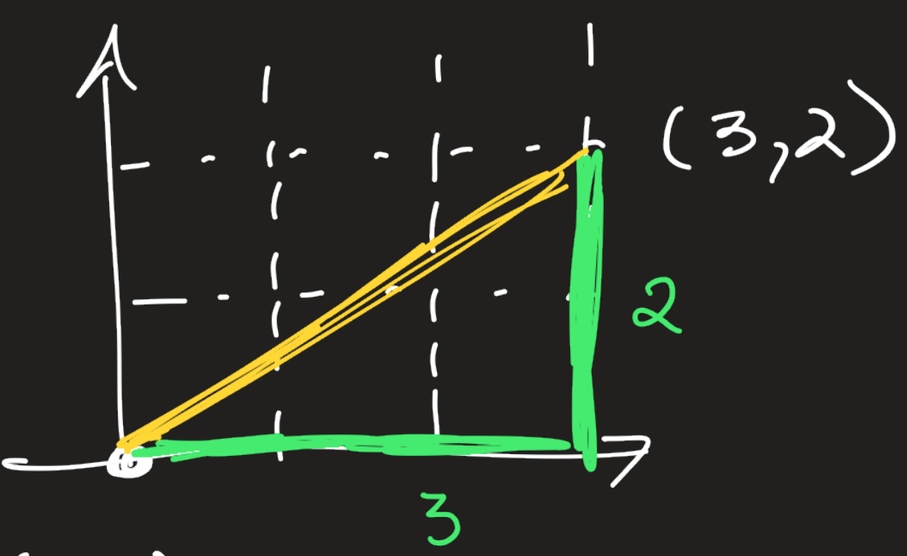
i.e. $2X^T X\omega^* - 2X^T y = 0$

$$\Rightarrow \boxed{\omega_{OLS} = (X^T X)^{-1} X^T y}$$

l_1 - regularization (aka LASSO regularization)

↳ Least Absolute Shrinkage and Selection Operator)

$$\|w\|_1 = \sum_{i=1}^{d} |w_i|$$



$$\left\|\begin{pmatrix}3\\2\end{pmatrix}\right\|_1 = 5$$

$$\left\|\begin{pmatrix}3\\2\end{pmatrix}\right\|_2^2 = 3^2 + 2^2 = 13$$

LASSO (l_2 loss + l_1 regularization)

$$w^* = \underset{w}{\operatorname{argmin}} \frac{1}{n}\|Xw - y\|_2^2 + \lambda\|w\|_1$$

This does not have a closed-form solution.

l_1 regularization widely used to fit sparse models

Regularization is a powerful technique that is ubiquitous in ML. In general we use the formulation

$$\hat{f} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \; R_n(f) + \lambda J(f)$$

where $J : \mathcal{F} \to \mathbb{R}$ is the regularizer and λ is the regularization parameter. This is RERM.

Risk decomposition for ML

all realizable

f^* — Bayes optimal estimator; minimizes the population risk

$$f^* = \operatorname*{argmin}_f R(f)$$

$f^*_{\mathcal{F}}$ — restrict our candidates to a model class for a more tractable optim. prob

$$f^*_{\mathcal{F}} = \operatorname*{argmin}_{f \in \mathcal{F}} R(f)$$

$\hat{f}_{\mathcal{F}}$ — use the empirical risk as a realizable estimate of pop. risk

$$\hat{f}_{\mathcal{F}} = \operatorname*{argmin}_{f \in \mathcal{F}} R_n(f) \qquad (*)$$

$\hat{f}_T$ — numerically solve the ERM prob using T iterations of some alg.

The goal in numerically optimizing (*) is to ensure that $R(\hat{f}_T)$ is as close to $R(f^*)$ as possible:

$$R(\hat{f}_T) = \underbrace{R(\hat{f}_T) - R(\hat{f}_{\mathcal{F}})}_{\text{optim error}} + \underbrace{R(\hat{f}_{\mathcal{F}}) - R(f^*_{\mathcal{F}})}_{\text{generalization error}}$$

$$+ \underbrace{R(f^*_{\mathcal{F}}) - R(f^*)}_{\text{approximation error}} + R(f^*)$$

Take-away: if $\max(\text{generalization error}, \text{approx error}) = \mathcal{O}(\varepsilon)$

then we only need optim error $= \mathcal{O}(\varepsilon)$