(FINISH) REGRESSION

(1) Bayesian Linear Regression
(aka Ridge Regression)

Likelihood: $Y | \mathbf{x} \sim \mathcal{N}(\mathbf{w}^T \mathbf{x}, \sigma^2)$ — same as last line

Prior: $\tilde{\mathbf{w}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \Rightarrow w_i \sim \mathcal{N}(0, \tau^2)$ IID components of $\tilde{\mathbf{w}}$

$\Rightarrow P(\tilde{\mathbf{w}}) = \prod_{i=1}^{d} \frac{1}{\sqrt{2\pi \tau^2}} e^{-\frac{(w_i - 0)^2}{2\tau^2}}$

Now $\hat{\mathbf{w}}_{MAP} = \arg\max_{\mathbf{w}} \log P(\tilde{\mathbf{w}} | \mathbf{D})$

$= \arg\max_{\mathbf{w}} \log \left[ \frac{P(\mathbf{D}|\tilde{\mathbf{w}}) P(\tilde{\mathbf{w}})}{P(\mathbf{D})} \right]$

$= \arg\max_{\mathbf{w}} \left[ \log P(\mathbf{D}|\tilde{\mathbf{w}}) + \log P(\tilde{\mathbf{w}}) \right]$ (trace)

$= \arg\max_{\mathbf{w}} \left[ \frac{1}{2} \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \frac{\delta}{2n} \right]$

$= \arg\min_{\mathbf{w}} \left[ \frac{1}{n} \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \frac{\delta}{n} \right]$

Avg. Loss trade-off: Regularization!
Bayesian Loss = Avg-Fitting-Error + Norm-Square of \( \beta \)

\[\lambda \to 0 \quad \text{relatively uninformative prior} \]
\[\Rightarrow \hat{\beta}_{\text{MAP}} = \hat{\beta}_{\text{MLE}} \]

\[\lambda \to \infty \]
\[\hat{\beta}_{\text{MAP}} \to \overline{0} \]
\[\Rightarrow \hat{\beta}_j = \overline{W}^TX_i \to \overline{0} \quad \forall i \]

Hmm, that's a bit odd. Why always zero?

replace \[\sum_{j=0}^{\lambda} \frac{d}{d\lambda} w_j^2 \]
with \[\sum_{j=1}^{\lambda} \frac{d}{d\lambda} w_j^2 \]
don't penalize norm of \( \overline{W} \)

Now as \( \lambda \to \infty \)
\[w_j \to 0 \quad \forall j \neq 0 \]
\[\overline{W} \to \frac{1}{n} \sum_{i=1}^{n} y_i \quad \text{[Avg of dataset. Why?] \{HW 2\}} \]

So \( \hat{\beta}_j \to \overline{y}_{\text{mean}} \quad \text{(Makes sense)} \)

Okay so \( \hat{\beta}_{\text{MAP}} \) minimizes \[\frac{1}{n} \| Y - X\overline{W} \|_2^2 + \lambda \| \overline{W} \|_2^2 \]

So \[\frac{\partial \mathcal{L}(\overline{W})}{\partial \overline{W}} = \frac{1}{n} \left[ -2Y^TX + 2\overline{W}^TX^TY \right] + \frac{2}{n} \overline{W} = 0 \]
\[
\frac{1}{n} X^T Y = \frac{1}{n} X^T X \bar{\omega} + \frac{1}{n} \bar{\omega}^2
\]

\[
\Rightarrow (X^T X + \lambda I) \bar{\omega} = X^T Y
\]

\[
\Rightarrow \hspace{1cm} \hat{\omega}_{\text{MAP}} = (X^T X + \lambda I)^{-1} X^T Y
\]

Compare with \( \hat{\omega}_{\text{MLE}} = X^T Y = (X^T X)^{-1} X^T Y \)

\[\frac{1}{n} X^T X \rightarrow (X^T X + \lambda I)\]

might not be invertible

shifts eigenvalue of \( X^T X \) up by \( \lambda \)

\[
\left\{
\begin{array}{l}
\hspace{1cm} \quad (A + \lambda I) \bar{\omega} = A \bar{\omega} + \lambda \bar{\omega}
\end{array}
\right.
\]

This is a Bayesian Interpretation of a Linear Algebra Procedure!

3. Polynomial Regression

Consider 1D data \( x \in \mathbb{R}^n \) for now (for simplicity)

\[\begin{array}{c}
\bar{X} \\
\bar{x}
\end{array}
\]

Doesn't look like a linear relationship. Can we do better?
Linear Regression: \( \hat{y} = w_0 + w_1 x \)

Polynomial Regression: \( \hat{y} = w_0 + w_1 x + w_2 x^2 + \ldots + w_d x^d \)
(of degree \( d \))

But we can easily rewrite this as:

\[
\begin{bmatrix}
  1 \\
  x \\
  x^2 \\
  \vdots \\
  x^d
\end{bmatrix}
\begin{bmatrix}
  w_0 \\
  w_1 \\
  \vdots \\
  w_d
\end{bmatrix}
\]

\[
= \mathbb{R}^d \phi(x)
\]

higher-dim feature space
"embedding"

\( \phi : \mathbb{R} \rightarrow \mathbb{R}^{d+1} \)

But \( \hat{y} = \mathbb{R}^T \phi(x) \) is still linear in \( \mathbb{R} \)!

That's all that matters!

Linearly in params \( \mathbb{R}^d \) not input \( \mathbb{R} \)

so same steps apply!

\[ \text{In general } x \in \mathbb{R}^d \rightarrow \phi(x) \in \mathbb{R}^D \quad D \neq d \]

say \( x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \)

\( \phi(x) = \left[ \begin{array}{c}
  x_1 \\
  x_2 \\
  \vdots \\
  x_D \\
  x_1^2 \\
  x_2^2 \\
  \vdots \\
  x_D^2
\end{array} \right] \)

Quadratic Feature Augmentation

Quadratic "Kernel Map"
\[ \text{we will see kernels later} \]

\( \hat{y} = \mathbb{R}^T \phi(x) \) still linear in \( \mathbb{R} \)

\( = \mathbb{R}^T A x + \mathbb{R}^T x \) Quadratic in \( x \)
NEW TOPIC: MODEL SELECTION

How do we pick \( d \) in 1-dim polynomial regression? \((x \in \mathbb{R})\)

Alternatively,

Model 0: \( y^{(0)} = w_0 \)
Model 1: \( y^{(1)} = w_0 + w_1 x \)

... 

Model D: \( y^{(D)} = w_0 + w_1 x + w_2 x^2 + \ldots + w_D x^D \)

Model Selection: Which model does is better?

Notice that 

\[
\hat{d}_{\text{bisection}} = \arg\min_{D \in \mathbb{N}} \frac{1}{n_{\text{training}}} \sum_{i=1}^{n_{\text{training}}} (y_i - \hat{w}_D x_i)^2
\]

So how about?

\[
d = \arg\min_{D} \frac{1}{n_{\text{training}}} \sum_{i=1}^{n_{\text{training}}} (y_i - \hat{w}_D x_i)^2
\]

Won't work. Why?

Problem: Model classes are nested. \( d = 100 \) will always give lowest training error (or E(train))

\( d \)th order polynomial \( \rightarrow \ldots \rightarrow \) all 10th order polynomials

\( 9 \)th order polynomials \( \rightarrow \ldots \rightarrow \) all 9th order polynomials
Large model classes (\(\leq\) strong ML models) will always do better in terms of error.

But why is low train error not a good thing? Because that's not what we care about.

4. What do we care about? Expected Error/Loss

\[ X, Y \sim P(x, y) \quad [\text{Unknown}] \]

What we want = \( \min_{\theta} E_{P(x, y)}[L(y, \hat{y}(\theta; x))] \)

\[ = \min_{\theta} \int \int L(y, \hat{y}(\theta; x)) P(x, y) dx dy \]

Space of \(x\) \quad Space of \(y\)

\[ \text{Say } \mathbb{R}^d \quad \text{Say } \mathbb{R}^1 \]

Problems
- \(P(x, y)\) unknown
- Integral hard to solve
- Min of integral

"Empirical Risk Minimization (ERM)"

\(=\) Late approximate integral with samples \((x_i, y_i) \sim P(x, y)\)

\[ E_{\text{approx}} = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{y}(x_i; \theta)) \]
## Model Selection in Practice

<table>
<thead>
<tr>
<th>ALL DATA</th>
<th>~60% (say)</th>
<th>~20%</th>
<th>~20%</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRAIN</td>
<td></td>
<td>VAL</td>
<td>TEST</td>
</tr>
</tbody>
</table>

\[
\hat{w} = \arg\min_\hat{w} E_{\text{train}}(\hat{w}, d)
\]

\[
d = \arg\min_\hat{d} \text{Eval}(\hat{w}, \hat{d})
\]

- Might try multiple models
- Used to estimate model parameters
- Used to choose model class
- Used to estimate expected error/loss
- No testing/learning on test set
- Otherwise a biased estimator