In EECS 16A, you saw the OMP algorithm as a method to find a sparse set of circularly-shifted vectors whose linear combination constituted a received signal. The motivation was in the context of decoding messages sent from satellites. It turns out that that this idea of finding a sparse-linear combination can also help us in Machine Learning. In particular it helps us with the problem of outlier removal.

1 Recap of OMP

Before we see how OMP is used in Machine Learning, let's do a brief recap of the algorithm by returning to the context in which EECS 16A covered it. Suppose we have m satellites, each transmitting a unique code $\mathrm{s_i} \in \mathbb{R}^n$. Satellites encode messages by multiplying the code they transmits by some scalar $\alpha_i \in \mathbb{R}$. At any given moment, only k satellites are transmitting, so at the receiver, we get the signal

$$\boldsymbol{y} = \sum_{i \in K} \alpha_i \boldsymbol{s_i}^{(\tau_i)}$$

where τ_i is a circular shift induced by the fact that each satellite is some distance away from the receiver and K is the set of all transmitting satellites. One natural way to proceed is iteratively. We can first "predict" which satellite is transmitting by finding the code $\boldsymbol{s_i}$ and shift τ_k such that $\boldsymbol{s_i}^{(\tau_k)}$ has the largest absolute cross-correlation than any other $\boldsymbol{s_j}^{(\tau_p)}$ ($i, j \leq m$ and $k, p \leq n$). Then, we perform OLS to find the coefficient α_i of $\boldsymbol{s_i}^{(\tau_k)}$, giving us how much $\boldsymbol{s_i}^{\tau_k}$ "explains" our received signal. Finally, we compute the residual $\boldsymbol{e} = \boldsymbol{y} - \alpha_i \boldsymbol{s_i}^{(\tau_k)}$, which tells us how much of our signal we still need to explain, and repeat the process on the residual $k-1$ times (or until our residual is small). Each time we repeat, we recompute the α_i since finding more of the satellites will give us a different estimate of α_i each time. This process is summarized by algorithm 1.

Algorithm 1: OMP Algorithm from 16A

```
Input: Codes $\boldsymbol{s_i}$, Received signal $\boldsymbol{y}$, Sparsity $k$, Residual Threshold $\epsilon$
Output: The indices of the used codes F, the sent messages $\boldsymbol{x}$
$\boldsymbol{e} = \boldsymbol{y}$, j = 1, A = [], F = $\emptyset$;
while $j \leq k$ and $||\boldsymbol{e}|| \geq \epsilon$ do
    i, t = FindMaxCrossCorrelation();
    $F = F \bigcup \{i\}$;
    $A = [A|s_i^{(t)}]$;
    $\boldsymbol{x} = (A^T A)^{-1} A^T \boldsymbol{y}$;
    $\boldsymbol{e} = \boldsymbol{y} - A\boldsymbol{x}$;
    j = j + 1;
end
return $F, \boldsymbol{x}$
```

2 OMP for Outlier Removal in Linear Regression

On its face, the setup for OMP seems entirely different from our Linear Regression setup. In Linear Regression, we have a series of targets $y_i = \boldsymbol{w}^T \boldsymbol{x_i} + \epsilon_i$ which are the sum of features $\boldsymbol{x_i} \in \mathbb{R}^m$ weighted by $\boldsymbol{w}$ with some added noise ϵ_i. In matrix form, this is

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} \boldsymbol{x_1}^T \\ \boldsymbol{x_2}^T \\ \vdots \\ \boldsymbol{x_n}^T \end{bmatrix} \boldsymbol{w} + \boldsymbol{\epsilon}.$$

It might not be entirely obvious at first, but we can interpret this exactly as an OMP problem by properly deciding what to call our "message" and what to call our "codes". First, let's express the OMP problem in matrix form.

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{s_1}^{(\tau_1)} & \boldsymbol{s_2}^{(\tau_1)} & \cdots & \boldsymbol{s_m}^{(\tau_1)} \end{bmatrix} \boldsymbol{\alpha}$$

If we ignore the noise and the shifts, clearly our received signal $\boldsymbol{y}$ can be thought of as our targets, and the sparse message $\boldsymbol{\alpha}$ is the weights of our model. The codes are simply the columns of the data matrix (i.e the features of each data point). We can ignore the shifts τ_i because there is nothing in Linear Regression which suggests a "delay".

2.1 Outliers

How does this interpretation help us with outlier removal exactly? An outlier is a point which lies very far from the true linear model. Mathematically, it means for the target y_i, ϵ_i is very large, meaning it is corrupted by a lot of noise, so it can't be explained by our linear model.

Because these points can't be well explained by a linear model, if we were to run OLS on data containing outliers, the outliers would skew the model away from the actual fit because OLS will try and balance minimizing the residual to the outliers against minimizing the residual from all of the uncorrupted data.

2.2 Interpreting OMP as an Outlier Removal Problem

How does OMP help us solve the problem of outliers? Well if we knew which points were outliers, we could remove them from our training set and train with the remaining data. Recall that

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} \boldsymbol{x_1}^T \\ \boldsymbol{x_2}^T \\ \vdots \\ \boldsymbol{x_n}^T \end{bmatrix} \boldsymbol{w} + \boldsymbol{\epsilon}.$$

If we break up $\boldsymbol{\epsilon}$, we see,

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} \boldsymbol{x_1}^T \\ \boldsymbol{x_2}^T \\ \vdots \\ \boldsymbol{x_n}^T \end{bmatrix} \boldsymbol{w} + \sum \epsilon_i \boldsymbol{e_i} = \begin{bmatrix} X & I \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{\epsilon} \end{bmatrix}$$

where $\boldsymbol{e_i}$ are the standard basis vectors. What this tells us is that outliers are essentially large perturbations along the standard basis in feature space. Since we only have a few outliers, $\boldsymbol{\epsilon}$ is sparse.

Thinking about this in the context of OMP, we already know that the features of our data matrix can explain our signal. We now need to find the standard basis vectors which also explain our signal. In the context of satellites, it is as if we have m satellites transmitting their codes normally but then "adversarial" satellites which very loudly transmit one of the standard basis vectors, effectively drowning out the other satellites. If we remove the loud, "adversarial" satellites, then we can recover our true message again.

This interpretation naturally leads us to OMP initialized with $A = X$. We can then proceed as normal. Once we find the outliers, we remove them from the training set and proceed with OLS as normal. Because there are no delays, we don't need to compute the circular cross-correlation. We only need to find the maximum, absolute dot product of each standard basis vector with the residual $\boldsymbol{r}$. In other words, we need to find

$$i = \underset{i}{\operatorname{argmin}} |\langle \boldsymbol{r}, \boldsymbol{e_i} \rangle| = \underset{i}{\operatorname{argmax}} |\boldsymbol{r}[i]|$$

where r_i is the residual for the ith datapoint.

Algorithm 2: OMP to Detect Outliers

```
Input: Data matrix $X$, Predictions $\boldsymbol{y}$, Sparsity $k$, Residual Threshold $\epsilon$
Output: Set of data indices $F$ where the outliers are
$\boldsymbol{r} = \boldsymbol{y} - X(X^TX)^{-1}X^T\boldsymbol{y}$, j = 1, A = X, F = $\emptyset$;
while $j \leq k$ and $||\boldsymbol{r}|| \geq \epsilon$ do
    i = $\operatorname{argmin}_i |\boldsymbol{r}[i]|$;
    $F = F \bigcup \{i\}$;
    $A = [A|\boldsymbol{e_i}]$;
    $\boldsymbol{x} = (A^TA)^{-1}A^T\boldsymbol{y}$;
    $\boldsymbol{r} = \boldsymbol{y} - A\boldsymbol{x}$;
    j = j + 1;
end
return $F$
```

It turns out that we do not have to initialize OMP with the data matrix X. Instead, we can run it on the matrix $[X|I]$ since, assuming the features are scaled properly, the true model features should explain the outputs the most and hence be included by OMP. This, of course, breaks down if the input data is poorly normalized.