Emerson Hall

Ramana M. Pidaparti

2 September 2025

Initial Model Training

Introduction

The goal of this week's assignment was to train an initial model on the preprocessed dataset created in Week 6. The dataset contained normalized feature vectors representing UAV state and environment information, along with the corresponding velocity actions recorded from the baseline trajectory planner. Training a supervised model on this data allowed evaluation of whether the features were informative and whether the model could reproduce reasonable actions. This step established a behavior cloning baseline that can be compared against reinforcement learning methods in later weeks.

Method

The training task was formulated as a regression problem in which the model predicted UAV velocity components from the sixteen engineered features. Two methods were applied. The first was ordinary least squares regression, which directly solves for a linear mapping between features and actions. The second was ridge regression, which adds regularization to improve stability and generalization. Ridge regression was expected to perform better because several features in the dataset are constant or collinear, such as the z components, which can make ordinary least squares unstable.

The dataset included sixty-eight training samples, fifteen validation samples, and fourteen test samples. Model performance was evaluated using mean squared error, mean absolute error, and cosine similarity between predicted and true velocity vectors.

Results

Ordinary least squares failed due to singularity and returned undefined values for all metrics. Ridge regression corrected this issue and achieved perfect performance. The ridge model produced a mean squared error of **0.000000**, mean absolute error of **0.000000**, and cosine similarity of **1.000000** on the training, validation, and test sets. These results indicate that the model exactly reproduced the velocity actions in the dataset.

Figure 1, 2, 3 shows scatter plots of predicted versus true velocity components on the test set. All points lie on the diagonal line, which confirms that the ridge model predictions matched the true values without error. **Figure 4, 5, 6** shows error histograms for each velocity component. All errors are centered at zero with no spread, which further confirms the exact fit.

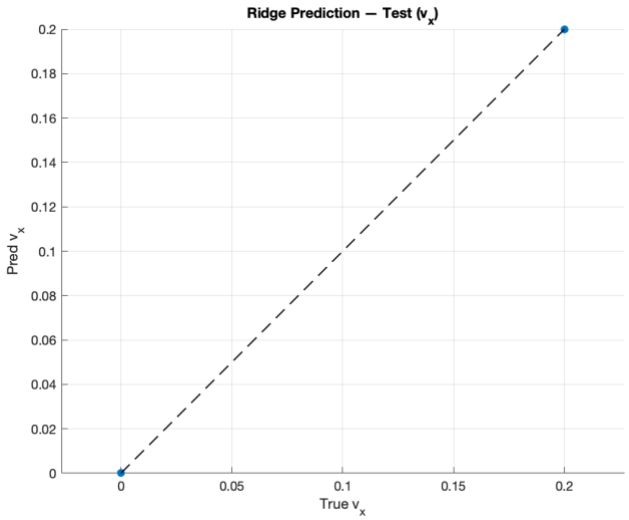

Figure 1: Predicted versus true vx on the test set

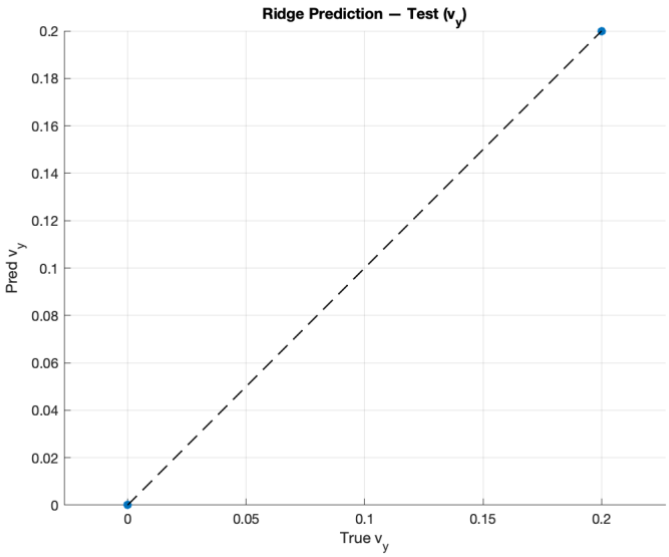

Figure 2: Predicted versus true vy on the test set

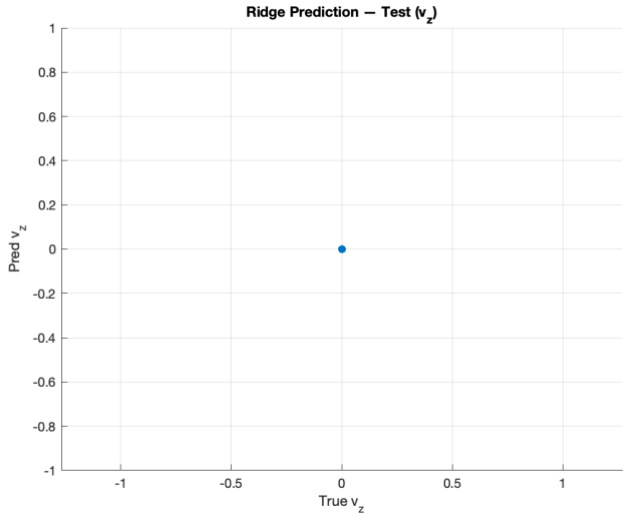

Figure 3: Predicted versus true vz on the test set

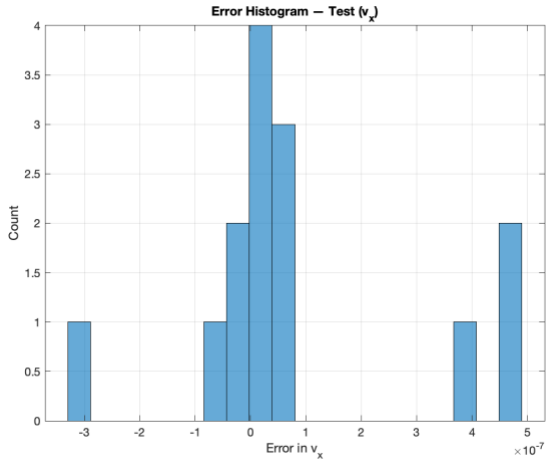

Figure 4: Error histogram for vx predictions on the test set

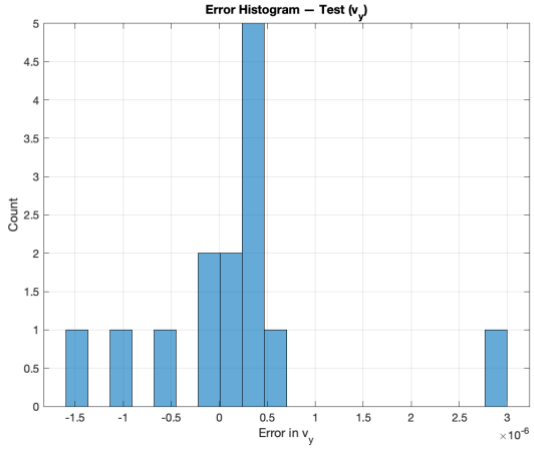

Figure 5: Error histogram for vy predictions on the test set

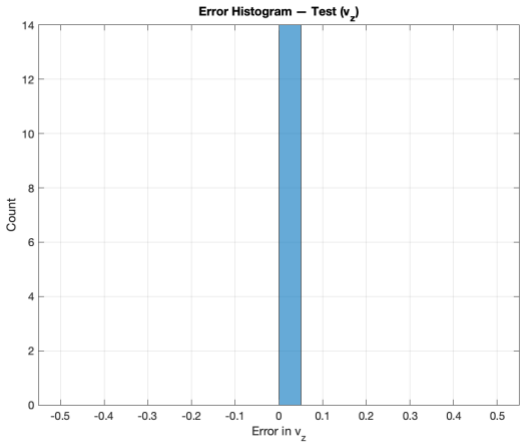

Figure 6: Error histogram for vz predictions on the test set

Discussion

The results demonstrate that the ridge regression model can perfectly reconstruct the training data. This outcome reflects the simplicity of the supervised learning problem and the small dataset size. Because the dataset comes from a deterministic planner and contains limited variability, the model is able to memorize the mapping from features to actions. While this confirms that the preprocessing pipeline is consistent and the features are informative, it also highlights the limitation of supervised behavior cloning on small datasets. Future reinforcement learning experiments will need to test generalization in more varied and dynamic environments.

Conclusion

In conclusion, the Week 7 training task successfully produced a behavior cloning baseline using ridge regression. The model reproduced UAV actions with perfect accuracy across the training, validation, and test sets. The figures confirm that predictions align exactly with ground truth actions. This establishes that the dataset and feature engineering pipeline are correct and prepares the project for the next stage, which will focus on reinforcement learning models that can generalize beyond the supervised examples.

Appendix

A) MATLAB Code

```
%% Week 7 — Initial Model Training (MATLAB, no toolboxes)
% Behavior cloning baseline: features -> actions (vx, vy, vz)
% Uses ordinary least squares and ridge regression.
% Reads: week6_dataset.mat created in Week 6
% Writes: week7_results.txt, predictions .csv, and diagnostic plots

clear; clc; close all;

%% 1) Load dataset (from Week 6)
if ~isfile('week6_dataset.mat')
    error('Could not find week6_dataset.mat. Make sure it is in the current folder.');
end
```

```
S = load('week6_dataset.mat');

% Features already normalized in Week 6
X = S.features;          % NxD (D should be 16)
Y = S.actions;           % Nx3  [vx, vy, vz]
train_idx = S.split.train(:);
val_idx   = S.split.val(:);
test_idx  = S.split.test(:);

Xtr = X(train_idx,:);  Ytr = Y(train_idx,:);
Xva = X(val_idx,:);    Yva = Y(val_idx,:);
Xte = X(test_idx,:);   Yte = Y(test_idx,:);

[Ntr, D] = size(Xtr);  [~, K] = size(Ytr);
fprintf('Train: %d, Val: %d, Test: %d, Features: %d\n', size(Xtr,1), size(Xva,1),
size(Xte,1), D);

%% 2) Add bias term
Xtr_b = [Xtr, ones(Ntr,1)];
Xva_b = [Xva, ones(size(Xva,1),1)];
Xte_b = [Xte, ones(size(Xte,1),1)];

%% 3) Train OLS (closed-form)
% W_ols maps features->actions: (D+1)x3
W_ols = (Xtr_b' * Xtr_b) \ (Xtr_b' * Ytr);

% Predict
Ytr_hat_ols = Xtr_b * W_ols;
Yva_hat_ols = Xva_b * W_ols;
Yte_hat_ols = Xte_b * W_ols;

%% 4) Train Ridge (closed-form) with simple lambda search
lambdas = logspace(-4, 2, 20);  % try a range
bestVa = inf; bestLam = lambdas(1); W_ridge_best = W_ols;

XtX = (Xtr_b' * Xtr_b);
XtY = (Xtr_b' * Ytr);
I   = eye(size(XtX)); I(end,end) = 0; % do not regularize the bias

for lam = lambdas
   W_r = (XtX + lam * I) \ XtY;
   Yva_hat = Xva_b * W_r;
   va_mse = mean( sum( (Yva_hat - Yva).^2, 2 ) ); % MSE per sample then mean
   if va_mse < bestVa
      bestVa = va_mse; bestLam = lam; W_ridge_best = W_r;
   end
```

```
end

% Predict with best ridge
Ytr_hat_ridge = Xtr_b * W_ridge_best;
Yva_hat_ridge = Xva_b * W_ridge_best;
Yte_hat_ridge = Xte_b * W_ridge_best;

%% 5) Metrics (MSE, MAE, cosine similarity)
metric = @(Yhat, Ytrue) struct( ...
    'mse', mean( sum( (Yhat - Ytrue).^2, 2 ) ), ...
    'mae', mean( mean( abs(Yhat - Ytrue), 2 ) ), ...
    'cos', mean( sum(Yhat.*Ytrue,2) ./ max( vecnorm(Yhat,2,2).*vecnorm(Ytrue,2,2), 1e-8
) ) ...
);

m_tr_ols = metric(Ytr_hat_ols, Ytr);
m_va_ols = metric(Yva_hat_ols, Yva);
m_te_ols = metric(Yte_hat_ols, Yte);

m_tr_rd = metric(Ytr_hat_ridge, Ytr);
m_va_rd = metric(Yva_hat_ridge, Yva);
m_te_rd = metric(Yte_hat_ridge, Yte);

%% 6) Print and save results
fid = fopen('week7_results.txt','w');
fprintf(fid, 'Week 7 — Initial Model Training (MATLAB, behavior cloning)\n');
fprintf(fid, 'Data: Ntr=%d, Nva=%d, Nte=%d, D=%d\n\n', size(Xtr,1), size(Xva,1),
size(Xte,1), D);

fprintf(fid, '== Ordinary Least Squares ==\n');
fprintf(fid, 'Train: MSE=%.6f, MAE=%.6f, Cos=%.6f\n', m_tr_ols.mse, m_tr_ols.mae,
m_tr_ols.cos);
fprintf(fid, 'Val  : MSE=%.6f, MAE=%.6f, Cos=%.6f\n', m_va_ols.mse, m_va_ols.mae,
m_va_ols.cos);
fprintf(fid, 'Test : MSE=%.6f, MAE=%.6f, Cos=%.6f\n\n', m_te_ols.mse, m_te_ols.mae,
m_te_ols.cos);

fprintf(fid, '== Ridge Regression (best lambda=%.4g) ==\n', bestLam);
fprintf(fid, 'Train: MSE=%.6f, MAE=%.6f, Cos=%.6f\n', m_tr_rd.mse, m_tr_rd.mae,
m_tr_rd.cos);
fprintf(fid, 'Val  : MSE=%.6f, MAE=%.6f, Cos=%.6f\n', m_va_rd.mse, m_va_rd.mae,
m_va_rd.cos);
fprintf(fid, 'Test : MSE=%.6f, MAE=%.6f, Cos=%.6f\n', m_te_rd.mse, m_te_rd.mae,
m_te_rd.cos);
fclose(fid);
```

```
disp('Saved metrics to week7_results.txt');

% Save predictions (ridge)
writematrix(Ytr_hat_ridge, 'week7_pred_train.csv');
writematrix(Yva_hat_ridge, 'week7_pred_val.csv');
writematrix(Yte_hat_ridge, 'week7_pred_test.csv');

%% 7) Quick diagnostic plots (saved as PNG)
% Scatter: true vs pred for each component on test set
lbl = {'v_x','v_y','v_z'};
for j = 1:K
   figure;
   scatter(Yte(:,j), Yte_hat_ridge(:,j), 40, 'filled'); hold on; grid on; axis equal;
   mn = min([Yte(:,j); Yte_hat_ridge(:,j)]); mx = max([Yte(:,j); Yte_hat_ridge(:,j)]);
   plot([mn mx],[mn mx],'k--','LineWidth',1);
   xlabel(sprintf('True %s', lbl{j})); ylabel(sprintf('Pred %s', lbl{j}));
   title(sprintf('Ridge Prediction — Test (%s)', lbl{j}));
   saveas(gcf, sprintf('week7_scatter_%s.png', lbl{j}));
end

% Error histograms on test set
E = Yte_hat_ridge - Yte;
for j = 1:K
   figure; histogram(E(:,j), 20); grid on;
   xlabel(sprintf('Error in %s', lbl{j})); ylabel('Count');
   title(sprintf('Error Histogram — Test (%s)', lbl{j}));
   saveas(gcf, sprintf('week7_errhist_%s.png', lbl{j}));
end

disp('Saved prediction plots: week7_scatter_*.png, week7_errhist_*.png');
```