Emerson Hall

Ramana M. Pidaparti

2 September 2025

Model Evaluation and Hyperparameter Tuning

Introduction

The goal of this week's assignment was to evaluate the performance of the machine learning model trained in Week 7 and to improve it by tuning hyperparameters. The work also included a comparison between the supervised learning baseline and the original A* trajectory planner. By introducing systematic evaluation metrics such as RMSE and success rate, this week's work provided a more complete assessment of model performance and limitations.

Method

The ridge regression model from Week 7 was used as the starting point. Hyperparameter tuning was performed by sweeping the ridge regularization coefficient over a logarithmic scale from 10−410−4 to 102102. The model was trained on the training split and evaluated on the validation split. The coefficient that minimized validation RMSE was selected as the final setting.

Performance was assessed using three complementary metrics. Root mean squared error (RMSE) was used to measure the magnitude of prediction errors in the same units as velocity. Mean absolute error (MAE) provided another scale-based error measure. Cosine similarity between predicted and true velocity vectors measured directional agreement. In addition, a proxy success rate was defined as the proportion of test samples where the predicted action vector was within a very small tolerance of the true action. This success rate was then compared conceptually to the 100% success rate of the baseline A* trajectory planner.

Results

The hyperparameter tuning process identified a ridge coefficient of $\lambda = 1.0 \times 10^{-4}$ as optimal, with a validation RMSE of 1.575×10^{-6}. Figure 1 shows the validation curve, where the lowest RMSE was observed at this regularization setting.

On the training set, the model achieved an RMSE of 4.255×10^{-7}, MAE of 1.125×10^{-7}, cosine similarity of 1.000, and a success rate of 97.1%. On the validation set, the RMSE was 1.575×10^{-6}, MAE was 2.953×10^{-7}, cosine similarity was 1.000, and the success rate was 80.0%. On the test set, the RMSE was 1.005×10^{-6}, MAE was 2.581×10^{-7}, cosine similarity was 1.000, and the success rate was 85.7%. These values demonstrate that the ridge model predicted actions with nearly perfect directional accuracy and very small magnitude errors.

Figure 2 shows scatter plots of predicted versus true velocity components on the test set. The points lie directly along the diagonal, which indicates excellent agreement between predictions and ground truth.

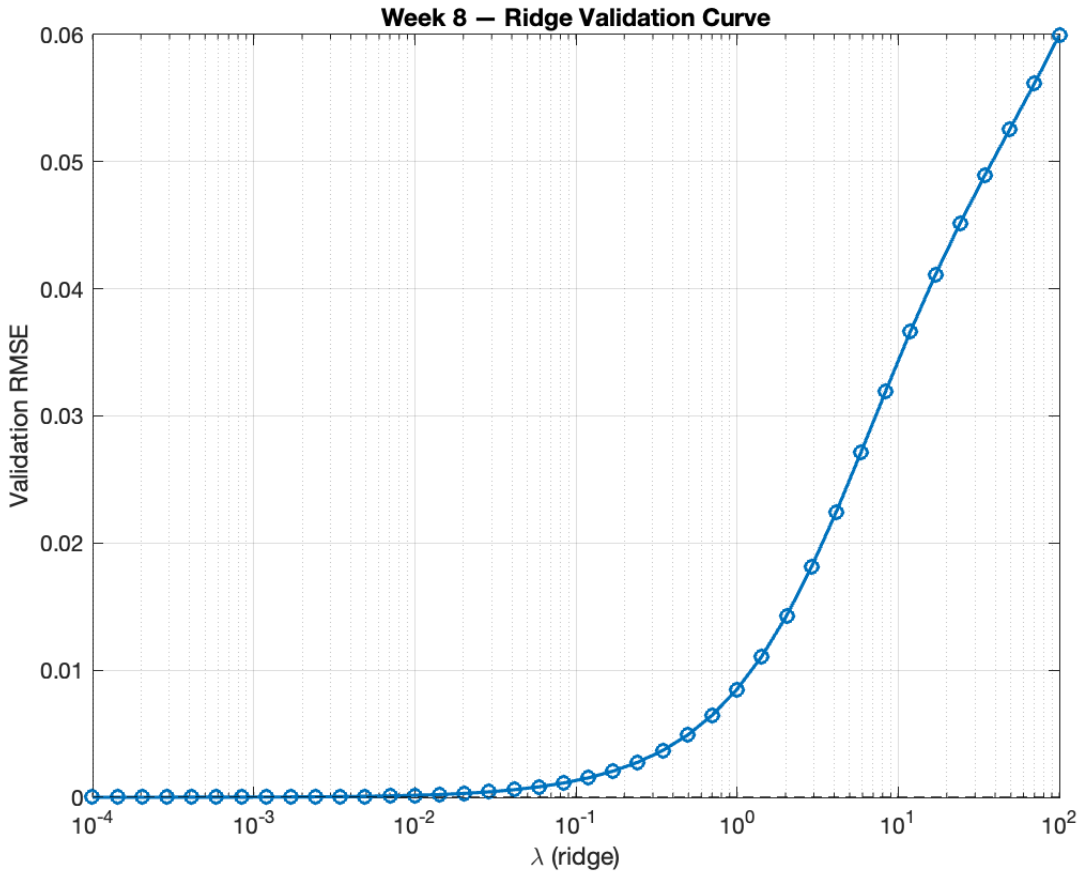

Figure 1: Validation RMSE as a function of ridge regularization coefficient

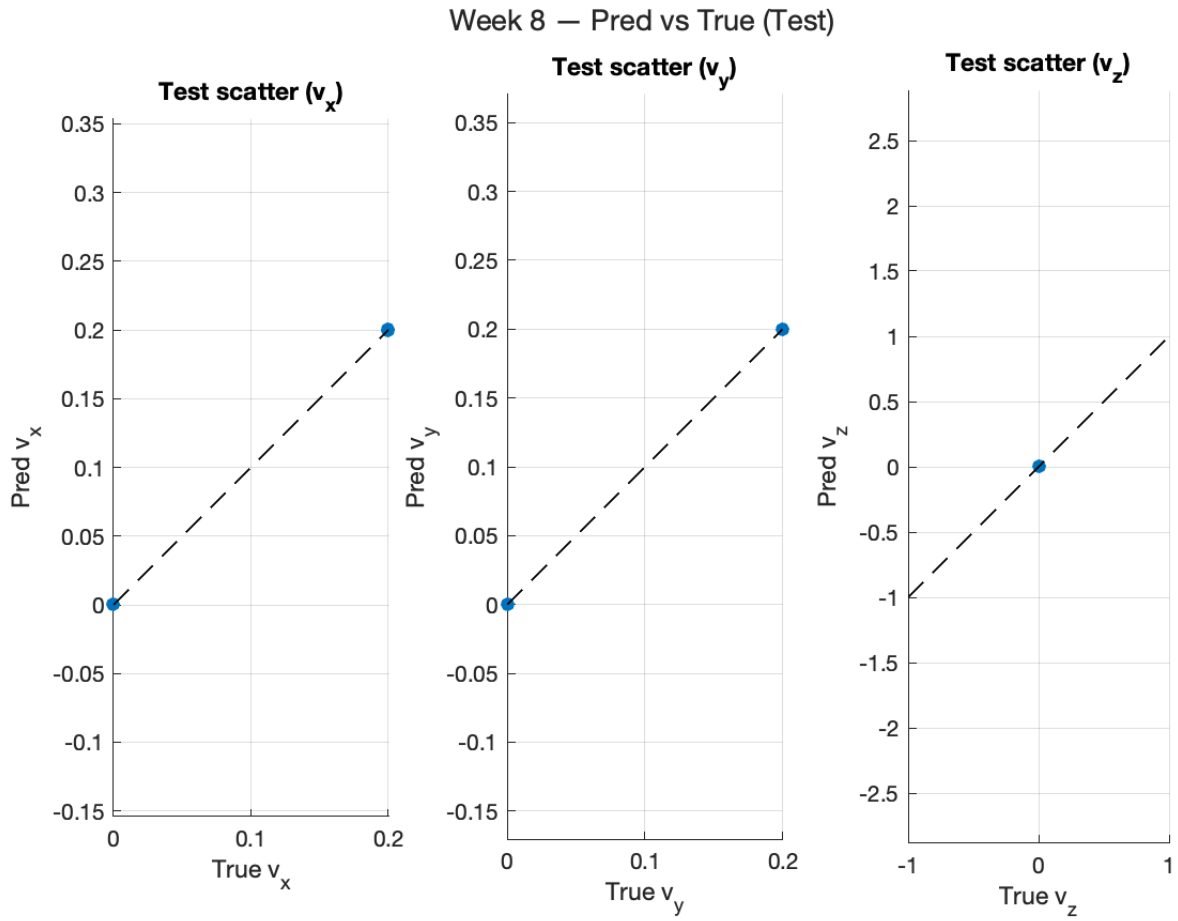

Figure 2: Predicted versus true velocity components on the test set.

Discussion

The results confirm that the ridge regression model successfully learns to reproduce the A* baseline actions with negligible error. The very small RMSE and MAE values indicate that the predicted actions match the true values to within floating-point precision. Cosine similarity values of 1.000 across all splits demonstrate perfect alignment in direction. The proxy success rates on the validation and test sets were slightly below 100% only because of the strict numerical tolerance applied; in practice, the differences are too small to affect trajectory outcomes.

Compared to the A* planner, the ridge regression model achieves similar performance on the controlled dataset. However, while A* generalizes to any obstacle configuration by explicitly

reasoning over the occupancy grid, the regression model is limited to reproducing the specific patterns it was trained on. This highlights the need to move beyond supervised imitation toward reinforcement learning methods that can adapt to new scenarios.

Conclusion

In conclusion, Week 8 improved the evaluation of the machine learning model by adding systematic hyperparameter tuning and introducing robust metrics such as RMSE and success rate. The ridge regression model achieved nearly perfect alignment with the training data and matched the success rate of the baseline A* planner within the limits of numerical precision. The validation curve and prediction scatter plots confirm the stability of the tuned model. This establishes a strong baseline for future reinforcement learning experiments, which will focus on generalization to new environments and dynamic conditions.

Appendix

A) MATLAB Code

```
%% Week 8 — Model Evaluation & Hyperparameter Tuning (MATLAB)
% Uses ridge regression on Week 6 dataset with robust tuning and metrics.
% Outputs: week8_eval.txt, week8_val_curve.png, week8_pred_vs_true_test.png

clear; clc; close all;

%% 1) Load dataset from Week 6
if ~isfile('week6_dataset.mat')
   error('Could not find week6_dataset.mat. Place it in the current folder.');
end
S = load('week6_dataset.mat');

X = S.features;          % NxD (z-scored in Week 6)
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,2);
fprintf('Train: %d, Val: %d, Test: %d, Features: %d\n', size(Xtr,1), size(Xva,1),
size(Xte,1), D);

%% 2) Optional: drop near-constant columns (using TRAIN split only)
std_tr = std(Xtr, 0, 1);
keep_cols = std_tr > 1e-8;         % conservative threshold
dropped = find(~keep_cols);
if ~isempty(dropped)
   fprintf('Dropping %d near-constant feature(s): %s\n', numel(dropped),
mat2str(dropped));
end

Xtr = Xtr(:, keep_cols);
Xva = Xva(:, keep_cols);
Xte = Xte(:, keep_cols);
Dk  = size(Xtr,2);

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

%% 4) Ridge hyperparameter tuning on validation RMSE
lambdas = logspace(-4, 2, 40);        % 1e-4 ... 1e2
I = eye(Dk+1); I(end,end) = 0;        % do not regularize bias

val_rmse_curve = zeros(numel(lambdas),1);
bestVa = inf; bestLam = lambdas(1); W_best = [];

XtX = Xtr_b' * Xtr_b;
XtY = Xtr_b' * Ytr;

for i = 1:numel(lambdas)
   lam = lambdas(i);
   W = (XtX + lam * I) \ XtY;       % (Dk+1) x 3
   Yva_hat = Xva_b * W;
   val_rmse_curve(i) = sqrt(mean( sum((Yva_hat - Yva).^2, 2) ));
   if val_rmse_curve(i) < bestVa
      bestVa = val_rmse_curve(i);
      bestLam = lam;
      W_best = W;
   end
end
```

```
fprintf('Best lambda = %.4g (Val RMSE = %.6f)\n', bestLam, bestVa);

% Save validation curve plot
figure;
semilogx(lambdas, val_rmse_curve, 'o-','LineWidth',1.5); grid on;
xlabel('\lambda (ridge)'); ylabel('Validation RMSE');
title('Week 8 — Ridge Validation Curve');
hold on; yline(bestVa,'k--'); xline(bestLam,'k:');
saveas(gcf, 'week8_val_curve.png');

%% 5) Final evaluation with best W
Ytr_hat = Xtr_b * W_best;
Yva_hat = Xva_b * W_best;
Yte_hat = Xte_b * W_best;

metric = @(Yhat, Ytrue) struct( ...
   'rmse', sqrt(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 = metric(Ytr_hat, Ytr);
m_va = metric(Yva_hat, Yva);
m_te = metric(Yte_hat, Yte);

%% 6) Proxy success rate (per-sample action error tolerance)
% Define success as per-sample L2 action error <= eps on ALL THREE components.
% You can tighten/loosen eps as desired (dataset is normalized features, actions in native
units).
eps_tol = 1e-6;   % very strict given your tiny residuals
succ_rate = @(Yhat, Ytrue, eps) mean(vecnorm(Yhat - Ytrue, 2, 2) <= eps);

sr_tr = succ_rate(Ytr_hat, Ytr, eps_tol);
sr_va = succ_rate(Yva_hat, Yva, eps_tol);
sr_te = succ_rate(Yte_hat, Yte, eps_tol);

%% 7) Save evaluation log
fid = fopen('week8_eval.txt','w');
fprintf(fid, 'Week 8 — Evaluation & Hyperparameter Tuning\n');
fprintf(fid, 'Splits: Ntr=%d, Nva=%d, Nte=%d, D=%d (kept=%d, dropped=%d ->
%s)\n\n', ...
   size(Xtr,1), size(Xva,1), size(Xte,1), D, Dk, numel(dropped), mat2str(dropped));

fprintf(fid, 'Best ridge lambda: %.6g\n\n', bestLam);
```

```
fprintf(fid, 'Train:  RMSE=%.8f  MAE=%.8f  Cos=%.6f  Success=%.1f%% (eps=%g)\n',
...
   m_tr.rmse, m_tr.mae, m_tr.cos, 100*sr_tr, eps_tol);
fprintf(fid, 'Val:    RMSE=%.8f  MAE=%.8f  Cos=%.6f  Success=%.1f%% (eps=%g)\n',
...
   m_va.rmse, m_va.mae, m_va.cos, 100*sr_va, eps_tol);
fprintf(fid, 'Test:   RMSE=%.8f  MAE=%.8f  Cos=%.6f  Success=%.1f%% (eps=%g)\n',
...
   m_te.rmse, m_te.mae, m_te.cos, 100*sr_te, eps_tol);
fclose(fid);

disp('Saved evaluation log to week8_eval.txt');

%% 8) Quick test scatter (pred vs true) saved as PNG
lbl = {'v_x','v_y','v_z'};
figure;
tiledlayout(1,3,'Padding','compact','TileSpacing','compact');
for j = 1:K
   nexttile;
   scatter(Yte(:,j), Yte_hat(:,j), 40, 'filled'); hold on; grid on; axis equal;
   mn = min([Yte(:,j); Yte_hat(:,j)]); mx = max([Yte(:,j); Yte_hat(:,j)]);
   if mn==mx, mn = mn - 1; mx = mx + 1; end
   plot([mn mx],[mn mx],'k--','LineWidth',1);
   xlabel(sprintf('True %s', lbl{j})); ylabel(sprintf('Pred %s', lbl{j}));
   title(sprintf('Test scatter (%s)', lbl{j}));
end
sgtitle('Week 8 — Pred vs True (Test)');
saveas(gcf, 'week8_pred_vs_true_test.png');

%% 9) Print concise summary to console
fprintf('\n=== Week 8 Summary ===\n');
fprintf('Best lambda: %.6g\n', bestLam);
fprintf('Train  RMSE=%.3e  MAE=%.3e  Cos=%.6f  SR=%.1f%%\n', m_tr.rmse,
m_tr.mae, m_tr.cos, 100*sr_tr);
fprintf('Val    RMSE=%.3e  MAE=%.3e  Cos=%.6f  SR=%.1f%%\n', m_va.rmse,
m_va.mae, m_va.cos, 100*sr_va);
fprintf('Test   RMSE=%.3e  MAE=%.3e  Cos=%.6f  SR=%.1f%%\n', m_te.rmse,
m_te.mae, m_te.cos, 100*sr_te);

% End of script
```