Prediction of continuous variables

- Billionaire says: Wait, that’s not what I meant!
- You say: Chill out, dude.
- He says: I want to predict a continuous variable for continuous inputs: I want to predict salaries from GPA.
- You say: I can regress that...
Linear Regression

Prediction
\[ \hat{y} = w_0 + w_1 f_1(x) \]

Prediction
\[ \hat{y}_i = w_0 + w_1 f_1(x) + w_2 f_2(x) \]
Ordinary Least Squares (OLS)

Total error \( = \sum_i (y_i - \hat{y}_i)^2 = \sum_i \left( y_i - \sum_k w_k f_k(x_i) \right)^2 \)
The regression problem

- **Instances:** \(<x_j, t_j>\)
- **Learn:** Mapping from \(x\) to \(t(x)\)
- **Hypothesis space:**
  - Given, basis functions \(\{h_1, \ldots, h_k\}\)
  - \(h_i(x) \in \mathbb{R}\)
  - Find coeffs \(w=\{w_1, \ldots, w_k\}\)

- Why is this usually called *linear regression*?
  - model is linear in the parameters
  - Can we estimate functions that are not lines???
Linear Basis: 1D input

Need a bias term: \( \{ h_1(x) = x, h_2(x) = 1 \} \)
• Parabola: \{h_1(x) = x^2, h_2(x) = x, h_3(x) = 1\}

• 2D: \{h_1(x) = x_1^2, h_2(x) = x_2^2, h_3(x) = x_1 x_2, \ldots \}

• Can define any basis functions \( h_i(x) \) for n-dimensional input \( x = \langle x_1, \ldots, x_n \rangle \)
The regression problem

- **Instances:** \(<x_j, t_j>\)
- **Learn:** Mapping from x to t(x)
- **Hypothesis space:**
  - Given, basis functions \(\{h_1, ..., h_k\}\)
  - \(h_i(x) \in \mathbb{R}\)
  - Find coeffs \(w = \{w_1, ..., w_k\}\)

- Why is this usually called *linear regression*?
  - model is linear in the parameters
  - Can we estimate functions that are not lines???

- Precisely, minimize the residual squared error:

\[
\mathbf{w}^* = \arg \min_{\mathbf{w}} \sum_j \left( t(x_j) - \sum_i w_i h_i(x_j) \right)^2
\]
Regression: matrix notation

\[ w^* = \arg \min_w \sum_j \left( t(x_j) - \sum_i w_i h_i(x_j) \right)^2 \]

\[ w^* = \arg \min_w (Hw - t)^T (Hw - t) \]

residual error

\[ H = \begin{bmatrix} h_1 & \ldots & h_K \end{bmatrix} \]

\[ w = \begin{bmatrix} \vdots \end{bmatrix} \]

\[ t = \begin{bmatrix} t \end{bmatrix} \]

N data points

K basis functions

weights

N observed outputs
Regression: closed form solution

\[ w^* = \arg \min_w \sum_j \left( t(x_j) - \sum_i w_i h_i(x_j) \right)^2 \]

\[ w^* = \arg \min_w (Hw - t)^T (Hw - t) \]

\[ F(w) = (Hw - t)^T (Hw - t) \]

\[ \nabla_w F(w) = 0 \]

\[ 2H^T (Hw - t) = 0 \]

\[ H^T Hw - H^T t = 0 \]

\[ w^* = (H^T H)^{-1} H^T t \]
Regression solution: simple matrix math

\[
\mathbf{w}^* = \arg \min_{\mathbf{w}} (\mathbf{Hw} - \mathbf{t})^T (\mathbf{Hw} - \mathbf{t})
\]

residual error

solution: \( \mathbf{w}^* = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{t} = \mathbf{A}^{-1} \mathbf{b} \)

where \( \mathbf{A} = \mathbf{H}^T \mathbf{H} = \begin{bmatrix} \vdots \end{bmatrix} \) \( \mathbf{b} = \mathbf{H}^T \mathbf{t} = \begin{bmatrix} \vdots \end{bmatrix} \)

\( \text{k×k matrix for k basis functions} \)

\( \text{k×1 vector} \)
But, why?

• Billionaire (again) says: Why sum squared error???
• You say: Gaussians, Dr. Gateson, Gaussians...
• Model: prediction is linear function plus Gaussian noise
  \[ t(x) = \sum_i w_i h_i(x) + \epsilon \]
• Learn \( \mathbf{w} \) using MLE:
  \[
P(t \mid \mathbf{x}, \mathbf{w}, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{[t-\sum_i w_i h_i(x)]^2}{2\sigma^2}}
  \]
Maximizing log-likelihood

Maximize wrt \( w \):

\[
\ln P(D \mid w, \sigma) = \ln \left( \frac{1}{\sigma \sqrt{2\pi}} \right)^N \prod_{j=1}^{N} e^{-\frac{[t_j - \sum_i w_i h_i(x_j)]^2}{2\sigma^2}}
\]

\[
\arg \max_w \ln \left( \frac{1}{\sigma \sqrt{2\pi}} \right)^N + \sum_{j=1}^{N} \frac{[t_j - \sum_i w_i h_i(x_j)]^2}{2\sigma^2}
\]

\[
= \arg \max_w \sum_{j=1}^{N} \frac{[t_j - \sum_i w_i h_i(x_j)]^2}{2\sigma^2}
\]

\[
= \arg \min_w \sum_{j=1}^{N} [t_j - \sum_i w_i h_i(x_j)]^2
\]

Least-squares Linear Regression is MLE for Gaussians!!!
Regularization in Linear Regression

• One sign of overfitting: large parameter values!

• Regularized or penalized regressions modified learning object to penalize large parameters
Ridge Regression

- Introduce a new objective function:

\[
\hat{w}_{ridge} = \arg\min_w \sum_{j=1}^{N} \left( t(x_j) - (w_0 + \sum_{i=1}^{k} w_i h_i(x_j)) \right)^2
\]

- Prefer low error but also add a squared penalize for large weights
- \( \lambda \) is hyperparameter that balances tradeoff
- Explicitly writing out bias feature (essentially \( h_0=1 \)), which is not penalized
Ridge Regression: matrix notation

\[ \hat{w}_{ridge} = \arg\min_w \sum_{j=1}^{N} \left( t(x_j) - (w_0 + \sum_{i=1}^{k} w_i h_i(x_j)) \right)^2 + \lambda \sum_{i=1}^{k} w_i^2 \]

\[ = \arg\min_w (Hw - t)^T(Hw - t) + \lambda w^T I_{0+k} w \]

residual error

\[ H = \begin{bmatrix} h_1 & \cdots & h_K \\ 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} \text{N data points} \\ \text{bias column and k basis functions} \end{bmatrix} \]

\[ w = \begin{bmatrix} \text{weights} \end{bmatrix} \]

\[ t = \begin{bmatrix} \text{measurements} \end{bmatrix} \]

\[ I_{0+k} = \begin{bmatrix} \text{k+1 identity matrix, but with 0 in upper left} \end{bmatrix} \]
Ridge Regression:
closed form solution

\[
\hat{w}_{\text{ridge}} = \arg \min_w \sum_{j=1}^{N} \left( t(x_j) - (w_0 + \sum_{i=1}^{k} w_i h_i(x_j)) \right)^2 + \lambda \sum_{i=1}^{k} w_i^2
\]

\[
\hat{w}_{\text{ridge}} = \arg \min_w \left( \mathbf{H} \mathbf{w} - \mathbf{t} \right)^T \left( \mathbf{H} \mathbf{w} - \mathbf{t} \right) + \lambda \mathbf{w}^T \mathbf{I}_{0+k} \mathbf{w}
\]

\[
\mathbf{F}(\mathbf{w}) = \left( \mathbf{H} \mathbf{w} - \mathbf{t} \right)^T \left( \mathbf{H} \mathbf{w} - \mathbf{t} \right) + \lambda \mathbf{w}^T \mathbf{I}_{0+k} \mathbf{w}
\]

\[
\nabla_{\mathbf{w}} \mathbf{F}(\mathbf{w}) = \mathbf{0}
\]

\[
2\mathbf{H}^T (\mathbf{H} \mathbf{w} - \mathbf{t}) + 2\lambda \mathbf{I}_{0+k} \mathbf{w} = \mathbf{0}
\]

\[
\mathbf{w}_{\text{ridge}}^* = \left( \mathbf{H}^T \mathbf{H} + \lambda \mathbf{I}_{0+k} \right)^{-1} \mathbf{H}^T \mathbf{t}
\]
Regression solution: simple matrix math

\[ \hat{w}_{ridge} = \arg \min_w \sum_{j=1}^{N} \left( t(x_j) - (w_0 + \sum_{i=1}^{k} w_i h_i(x_j)) \right)^2 + \lambda \sum_{i=1}^{k} w_i^2 \]

\[ = \arg \min_w (Hw - t)^T (Hw - t) + \lambda w^T I_{0+k} w \]

\[ \text{residual error} \]

\[ w^*_{ridge} = (H^T H + \lambda I_{0+k})^{-1} H^T t \]

Compare to un-regularized regression:

\[ w^* = (H^T H)^{-1} H^T t \]
Ridge Regression

How does varying lambda change $w$?

$$\hat{w}_{ridge} = \arg \min_w \sum_{j=1}^{N} \left( t(x_j) - (w_0 + \sum_{i=1}^{k} w_i h_i(x_j)) \right)^2 + \lambda \sum_{i=1}^{k} w_i^2$$

- Larger $\lambda$? Smaller $\lambda$?
- As $\lambda \to 0$?
  - Becomes same as a MLE, unregularized
- As $\lambda \to \infty$?
  - All weights will be 0!
Typical approach: select $\lambda$ using cross validation, more on this later in the quarter.

Feature Weight

Larger $\lambda$ $\Leftarrow$ $\|\hat{\mathbf{w}}\|_2$ $\Rightarrow$ Smaller $\lambda$

From Kevin Murphy textbook
How to pick lambda?

- **Experimentation cycle**
  - Select a hypothesis $f$ to best match training set
  - Tune hyperparameters on held-out set
    - Try many different values of lambda, pick best one

- **Or, can do k-fold cross validation**
  - No held-out set
  - Divide training set into $k$ subsets
  - Repeatedly train on $k-1$ and test on remaining one
  - Average the results
Why squared regularization?

- Ridge:

\[ \hat{w}_{\text{ridge}} = \arg\min_w \sum_{j=1}^{N} \left( t(x_j) - (w_0 + \sum_{i=1}^{k} w_i h_i(x_j)) \right)^2 + \lambda \sum_{i=1}^{k} w_i^2 \]

- LASSO:

\[ \hat{w}_{\text{LASSO}} = \arg\min_w \sum_{j=1}^{N} \left( t(x_j) - (w_0 + \sum_{i=1}^{k} w_i h_i(x_j)) \right)^2 + \lambda \sum_{i=1}^{k} |w_i| \]

- Linear penalty pushes more weights to zero
- Allows for a type of feature selection
- But, not differentiable and no closed form solution....
Geometric Intuition

Geometric intuition of regularized objectives in 1d

LASSO solution:

\[ \hat{w}_{\text{LASSO}} = \arg \min_w \sum_{j=1}^N t(x_j) (w_0 + \sum_{i=1}^k w_i h_i(x_j))^2 + \sum_{i=1}^k |w_i| \]

Geometric Intuition for Sparsity

Lasso Ridge Regression

From Rob Tibshirani slides
Recall: Ridge Coefficient Path

Typical approach: select $\lambda$ using cross validation

LASSO Coefficient Path

Larger $\lambda$ $\leftrightarrow$ $\|\hat{\mathbf{w}}\|_1$ $\rightarrow$ Smaller $\lambda$

From Kevin Murphy textbook
Bias-Variance tradeoff – Intuition

• Model too simple: does not fit the data well
  – A *biased* solution

• Model too complex: small changes to the data, solution changes a lot
  – A *high-variance* solution
Bias-Variance Tradeoff

- Choice of hypothesis class introduces learning bias
  - More complex class $\rightarrow$ less bias
  - More complex class $\rightarrow$ more variance
Training set error

\[ w^* = \arg \min_w \sum_j \left( t(x_j) - \sum_i w_i h_i(x_j) \right)^2 \]

- Given a dataset (Training data)
- Choose a loss function
  - e.g., squared error (L₂) for regression
- Training error: For a particular set of parameters, loss function on training data:

\[
\text{error}_{\text{train}}(w) = \frac{1}{N_{\text{train}}} \sum_{j=1}^{N_{\text{train}}} \left( t(x_j) - \sum_i w_i h_i(x_j) \right)^2
\]
Training error as a function of model complexity

\[
\text{error}_{\text{train}}(w) = \frac{1}{N_{\text{train}}} \sum_{j=1}^{N_{\text{train}}} (t(x_j) - \sum_i w_i h_i(x_j))^2
\]
Prediction error

- Training set error can be a poor measure of "quality" of solution
- **Prediction error (true error):** We really care about error over all possibilities:

\[
error_{true}(w) = \mathbb{E}_x \left[ \left( t(x) - \sum_i w_i h_i(x) \right)^2 \right]
\]

\[
= \int_x \left( t(x) - \sum_i w_i h_i(x) \right)^2 p(x) dx
\]
Prediction error as a function of model complexity

\[
\text{error}_{\text{train}}(\mathbf{w}) = \frac{1}{N_{\text{train}}} \sum_{j=1}^{N_{\text{train}}} \left( t(x_j) - \sum_i w_i h_i(x_j) \right)^2
\]

\[
\text{error}_{\text{true}}(\mathbf{w}) = \int_{x} \left( t(x) - \sum_i w_i h_i(x) \right)^2 p(x) \, dx
\]
Computing prediction error

• To correctly predict error
  • Hard integral!
  • May not know \( t(\mathbf{x}) \) for every \( \mathbf{x} \), may not know \( p(\mathbf{x}) \)

\[
\text{error}_{\text{true}}(\mathbf{w}) = \int_{\mathbf{x}} \left( t(\mathbf{x}) - \sum_{i} w_i h_i(\mathbf{x}) \right)^2 p(\mathbf{x}) d\mathbf{x}
\]

• Monte Carlo integration (sampling approximation)
  • Sample a set of i.i.d. points \( \{\mathbf{x}_1, \ldots, \mathbf{x}_M\} \) from \( p(\mathbf{x}) \)
  • Approximate integral with sample average

\[
\text{error}_{\text{true}}(\mathbf{w}) \approx \frac{1}{M} \sum_{j=1}^{M} \left( t(\mathbf{x}_j) - \sum_{i} w_i h_i(\mathbf{x}_j) \right)^2
\]
Why training set error doesn’t approximate prediction error?

• Sampling approximation of prediction error:

\[ \text{error}_{\text{true}}(w) \approx \frac{1}{M} \sum_{j=1}^{M} \left( t(x_j) - \sum_i w_i h_i(x_j) \right)^2 \]

• Training error:

\[ \text{error}_{\text{train}}(w) = \frac{1}{N_{\text{train}}} \sum_{j=1}^{N_{\text{train}}} \left( t(x_j) - \sum_i w_i h_i(x_j) \right)^2 \]

• Very similar equations!!!
  – Why is training set a bad measure of prediction error???
Why training set error doesn’t approximate prediction error?

- Sampling approximation of prediction error:

  "Because you Cheated!!!"

  Training error good estimate for a single $w$, But you optimized $w$ with respect to the training error, and found $w$ that is good for this set of samples

- Training error is a (optimistically) biased estimate of prediction error

- Very similar equations!!!
  - Why is training set a bad measure of prediction error???
Test set error

\[ w^* = \arg \min_w \sum_j \left( t(x_j) - \sum_i w_i h_i(x_j) \right)^2 \]

• Given a dataset, **randomly** split it into two parts:
  – Training data – \( \{x_1, \ldots, x_{N_{\text{train}}} \} \)
  – Test data – \( \{x_1, \ldots, x_{N_{\text{test}}} \} \)

• Use training data to optimize parameters \( w \)

• **Test set error:** For the *final solution* \( w^* \), evaluate the error using:

\[
\text{error}_{\text{test}}(w) = \frac{1}{N_{\text{test}}} \sum_{j=1}^{N_{\text{test}}} \left( t(x_j) - \sum_i w_i h_i(x_j) \right)^2
\]
Test set error as a function of model complexity

\[
\text{error}_{\text{train}}(w) = \frac{1}{N_{\text{train}}} \sum_{j=1}^{N_{\text{train}}} (t(x_j) - \sum_{i} w_i h_i(x_j))^2
\]

\[
\text{error}_{\text{true}}(w) = \int_{x} \left( t(x) - \sum_{i} w_i h_i(x) \right)^2 p(x) dx
\]

\[
\text{error}_{\text{test}}(w) = \frac{1}{N_{\text{test}}} \sum_{j=1}^{N_{\text{test}}} (t(x_j) - \sum_{i} w_i h_i(x_j))^2
\]
Overfitting: this slide is so important we are looking at it again!

• **Assume:**
  – Data generated from distribution $D(X,Y)$
  – A hypothesis space $H$

• **Define:** errors for hypothesis $h \in H$
  – Training error: $error_{\text{train}}(h)$
  – Data (true) error: $error_{\text{true}}(h)$

• We say $h$ **overfits** the training data if there exists an $h' \in H$ such that:

$$error_{\text{train}}(h) < error_{\text{train}}(h')$$

and

$$error_{\text{true}}(h) > error_{\text{true}}(h')$$
Summary: error estimators

• Gold Standard:

\[ \text{error}_{true}(w) = \int_x \left( t(x) - \sum_i w_i h_i(x) \right)^2 p(x) dx \]

• Training: optimistically biased

\[ \text{error}_{train}(w) = \frac{1}{N_{train}} \sum_{j=1}^{N_{train}} \left( t(x_j) - \sum_i w_i h_i(x_j) \right)^2 \]

• Test: our final measure

\[ \text{error}_{test}(w) = \frac{1}{N_{test}} \sum_{j=1}^{N_{test}} \left( t(x_j) - \sum_i w_i h_i(x_j) \right)^2 \]
Error as a function of number of training examples for a fixed model complexity

\[
\text{error}_{\text{train}}(w) = \frac{1}{N_{\text{train}}} \sum_{j=1}^{N_{\text{train}}} \left( t(x_j) - \sum_i w_i h_i(x_j) \right)^2
\]

\[
\text{error}_{\text{test}}(w) = \frac{1}{N_{\text{test}}} \sum_{j=1}^{N_{\text{test}}} \left( t(x_j) - \sum_i w_i h_i(x_j) \right)^2
\]

little data \hspace{2cm} \text{bias} \hspace{2cm} \text{infinite data}
Error as function of regularization parameter, fixed model complexity

\[
\text{error}_{\text{train}}(w) = \frac{1}{N_{\text{train}}} \sum_{j=1}^{N_{\text{train}}} \left( t(x_j) - \sum_i w_i h_i(x_j) \right)^2
\]

\[
\text{error}_{\text{true}}(w) = \int_{x} \left( t(x) - \sum_i w_i h_i(x) \right)^2 p(x) \, dx
\]
Summary: error estimators

Be careful!!!

Test set only unbiased if you never never ever ever
never
never
ever
never
do any any any any any learning on the test data

For example, if you use the test set to select
the degree of the polynomial... no longer unbiased!!!
(We will address this problem later in the quarter)

Test: our final measure

$$error_{test}(w) = \frac{1}{N_{test}} \sum_{j=1}^{N_{test}} \left( t(x_j) - \sum_i w_i h_i(x_j) \right)^2$$
What you need to know

• Regression
  – Basis function = features
  – Optimizing sum squared error
  – Relationship between regression and Gaussians

• Regularization
  – Ridge regression math
  – LASSO Formulation
  – How to set lambda

• Bias-Variance trade-off