COMP 551 - Applied Machine Learning
Lecture 6 — Evaluation and Regularization

William L. Hamilton

(with slides and content from Joelle Pineau)

Quizzes

- Quiz 1 – Attempt 1
  - 246 students attempted.
  - 74% average.
- Quiz 1 – Attempt 2
  - 20 completed so far
  - 87% average so far

Learning in LDA: Covariance clarification

- Estimating Σ, from the training data:
  - Maximum Likelihood (ML) Estimate:

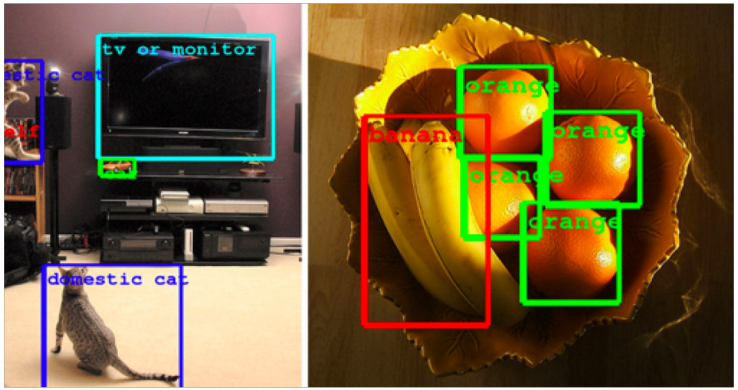
$$\Sigma = \sum_{k=0:1} \sum_{i=1:n} I(y_i = k)\ (x_i - \mu_k)(x_i - \mu_k)^T / (N_0 + N_1)$$

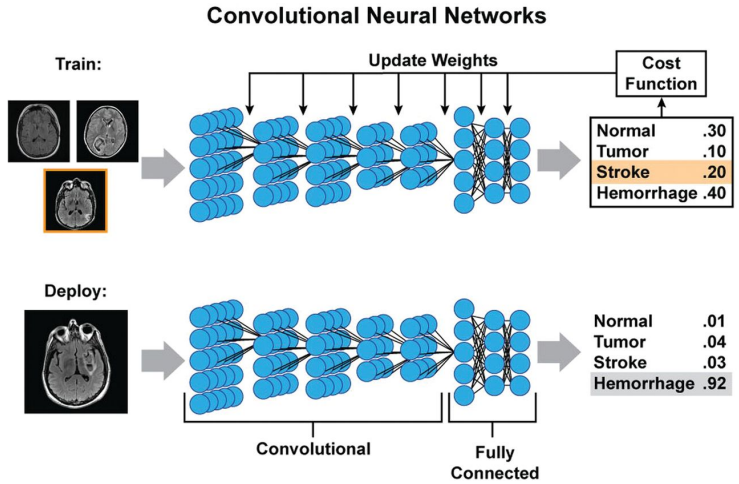
  - Unbiased Estimate:

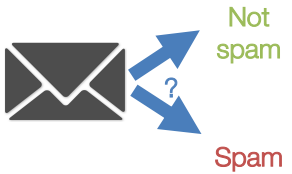
$$\Sigma = \sum_{k=0:1} \sum_{i=1:n} I(y_i = k)\ (x_i - \mu_k)(x_i - \mu_k)^T / (N_0 + N_1 - 2)$$

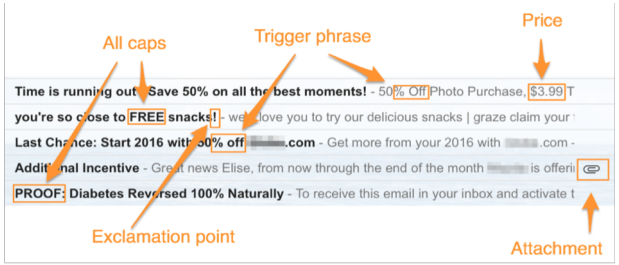
- The unbiased estimate involves a correction of the ML estimate (subtracting the number of classes in the denominator) and is from the Hastie et al. textbook (page 109).

Classification in the real world







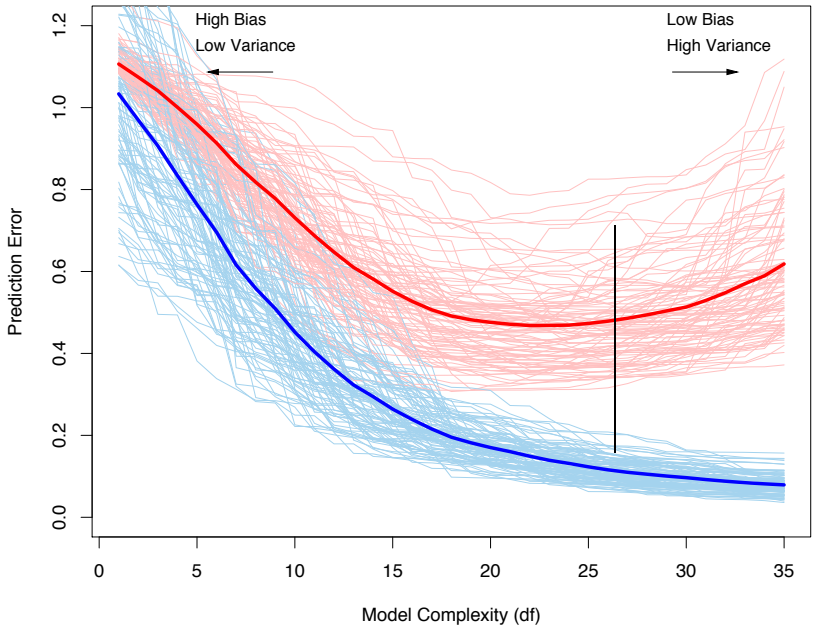


Classification in the real world

- Different objectives:
  - Selecting the right model for a problem.
  - Testing performance of a new algorithm.
  - Evaluating impact on a new application.

Minimizing the error

- Find the low point in the validation error:

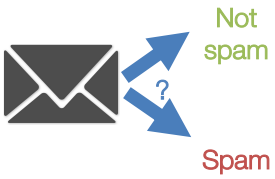


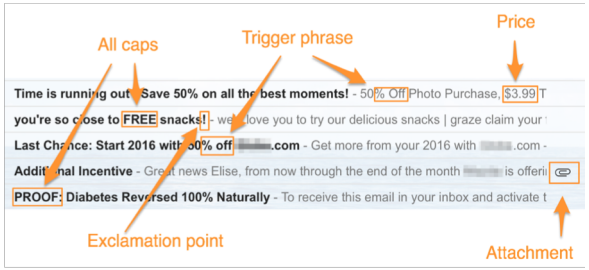
Validation error

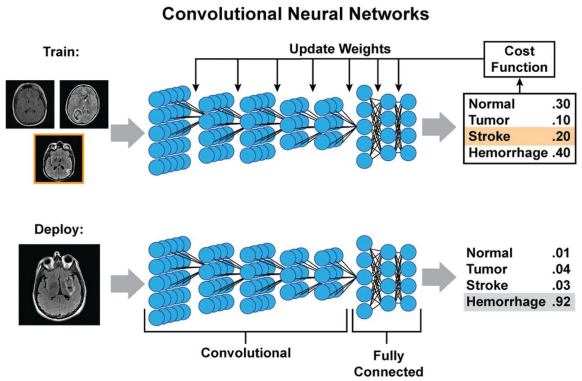
Train error

Not all errors are created equal

- Not all errors have equal impact!
- Different types of mistakes, particularly with classification tasks.
  - Incorrectly classifying a tumor is very different from accidently allowing a spam email though the filter!

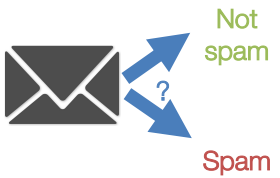


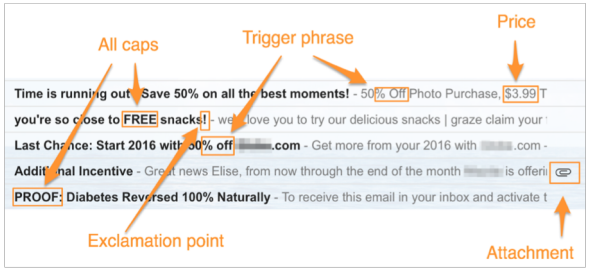


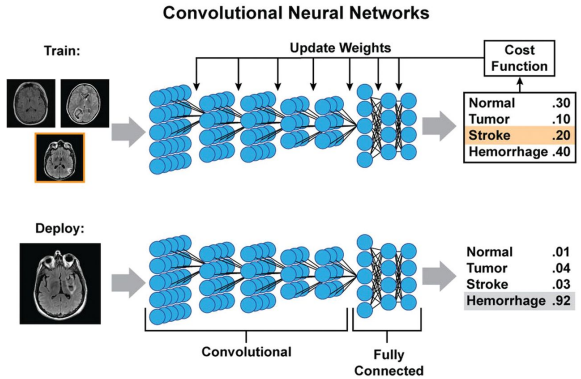


Datasets can be imbalanced

- Most real world data is not “balanced”, meaning that there classes are not equally distributed.
- It is easy to get high accuracy on imbalanced data!
  - E.g., if 99% of emails are not spam, then we can get 99% accuracy by simply predicting that everything is not spam!







Example 1

Int J Neural Syst. 2013 Oct;23(5):1350023. doi: 10.1142/S0129065713500238. Epub 2013 Jul 21.

Application of intrinsic time-scale decomposition (ITD) to EEG signals for automated seizure prediction.

Martis RJ[1], Acharya UR, Tan JH, Petznick A, Tong L, Chua CK, Ng EY.

Author information

Abstract

Intrinsic time-scale decomposition (ITD) is a new nonlinear method of time-frequency representation which can decipher the minute changes in the nonlinear EEG signals. In this work, we have automatically classified normal, interictal and ictal EEG signals using the features derived from the ITD representation. The energy, fractal dimension and sample entropy features computed on ITD representation coupled with decision tree classifier has yielded an average classification accuracy of 95.67%, sensitivity and specificity of 99% and 99.5%, respectively using 10-fold cross validation scheme. With application of the nonlinear ITD representation, along with conceptual advancement and improvement of the accuracy, the developed system is clinically ready for mass screening in resource constrained and emerging economy scenarios.

- Real-world study on predicting seizures for EEG signals.
- Claim the technique is “clinically ready for mass screening”

Example 1

Int J Neural Syst. 2013 Oct;23(5):1350023. doi: 10.1142/S0129065713500238. Epub 2013 Jul 21.

Application of intrinsic time-scale decomposition (ITD) to EEG signals for automated seizure prediction.

Martis RJ[1], Acharya UR, Tan JH, Petznick A, Tong L, Chua CK, Ng EY.

Author information

Abstract

Intrinsic time-scale decomposition (ITD) is a new nonlinear method of time-frequency representation which can decipher the minute changes in the nonlinear EEG signals. In this work, we have automatically classified normal, interictal and ictal EEG signals using the features derived from the ITD representation. The energy, fractal dimension and sample entropy features computed on ITD representation coupled with decision tree classifier has yielded an average classification accuracy of 95.67%, sensitivity and specificity of 99% and 99.5%, respectively using 10-fold cross validation scheme. With application of the nonlinear ITD representation, along with conceptual advancement and improvement of the accuracy, the developed system is clinically ready for mass screening in resource constrained and emerging economy scenarios.

- Real-world study on predicting seizures for EEG signals.
- Claim the technique is "clinically ready for mass screening"

Accuracy = True positives + True Negatives / Total number of examples
Sensitivity = True positives / Total number of actual positives
Specificity = True negatives / Total number of actual negatives

Performance metrics for classification

- Not all errors have equal impact!
- There are different types of mistakes, particularly in the classification setting.
  - E.g., Consider the diagnostic of a disease. Two types of mis-diagnostics:
    - Patient does not have disease but received positive diagnostic (Type I error);
    - Patient has disease but it was not detected (Type II error).
  - E.g. Consider the problem of spam classification:
    - A message that is not spam is assigned to the spam folder (Type I error);
    - A message that is spam appears in the regular folder (Type II error).
- Question: How many Type I errors are you willing to tolerate, for a reasonable rate of Type II errors?

Example 2

Clin Neurophysiol. 2013 Sep;124(9):1745-54. doi: 10.1016/j.clinph.2013.04.006. Epub 2013 May 3.

Seizure prediction in patients with mesial temporal lobe epilepsy using EEG measures of state similarity.

Gadhoumi K[1], Lina JM, Gotman J.

⊕ Author information

Abstract

OBJECTIVES: In patients with intractable epilepsy, predicting seizures above chance and with clinically acceptable performance has yet to be demonstrated. In this study, an intracranial EEG-based seizure prediction method using measures of similarity with a reference state is proposed.

METHODS: 1565 h of continuous intracranial EEG data from 17 patients with mesial temporal lobe epilepsy were investigated. The recordings included 175 seizures. In each patient the data was split into a training set and a testing set. EEG segments were analyzed using continuous wavelet transform. During training, a reference state was defined in the immediate preictal data and used to derive three features quantifying the discrimination between preictal and interictal states. A classifier was then trained in the feature space. Its performance was assessed using testing set and compared with a random predictor for statistical validation.

RESULTS: Better than random prediction performance was achieved in 7 patients. The sensitivity was higher than 85%, the warning rate was less than 0.35/h and the proportion of time under warning was less than 30%.

CONCLUSION: Seizures are predicted above chance in 41% of patients using measures of state similarity.

SIGNIFICANCE: Sensitivity and specificity levels are potentially interesting for closed-loop seizure control applications.

- Another real-world study on predicting seizures for EEG signals.
- >85% sensitivity.
- “Warning rate” of 0.35/hr.

Example 3

Conf Proc IEEE Eng Med Biol Soc. 2012;2012:1061-4. doi: 10.1109/EMBC.2012.6346117.

Low complexity algorithm for seizure prediction using Adaboost.

Ayinala M[1], Parhi KK.

⊕ Author information

Abstract

This paper presents a novel low-complexity patient-specific algorithm for seizure prediction. Adaboost algorithm is used in two stages of the algorithm: feature selection and classification. The algorithm extracts spectral power features in 9 different sub-bands from the electroencephalogram (EEG) recordings. We have proposed a new feature ranking method to rank the features. The key (top ranked) features are used to make a prediction on the seizure event. Further, to reduce the complexity of classification stage, a non-linear classifier is built based on the Adaboost algorithm using decision stumps (linear classifier) as the base classifier. The proposed algorithm achieves a sensitivity of 94.375% for a total of 71 seizure events with a low false alarm rate of 0.13 per hour and 6.5% of time spent in false alarms using an average of 5 features for the Freiburg database. The low computational complexity of the proposed algorithm makes it suitable for an implantable device.

PMID: 23366078 [PubMed - indexed for MEDLINE]

- Another real-world study on predicting seizures for EEG signals.
- 95% sensitivity.
- “False alarm rate” of 0.15/hr.

Terminology

- Type of classification outputs:
  - True positive (m11): Example of class 1 predicted as class 1.
  - False positive (m01): Example of class 0 predicted as class 1. Type 1 error.
  - True negative (m00): Example of class 0 predicted as class 0.
  - False negative (m10): Example of class 1 predicted as class 0. Type II error.
- Total number of instances: $m = m00 + m01 + m10 + m11$

Terminology

- Type of classification outputs:
  - True positive (m11): Example of class 1 predicted as class 1.
  - False positive (m01): Example of class 0 predicted as class 1. Type 1 error.
  - True negative (m00): Example of class 0 predicted as class 0.
  - False negative (m10): Example of class 1 predicted as class 0. Type II error.
- Total number of instances: m = m00 + m01 + m10 + m11
- Error rate: (m01 + m10) / m
  - If the classes are imbalanced (e.g. 10% from class 1, 90% from class 0), one can achieve low error (e.g. 10%) by classifying everything as coming from class 0!

Confusion matrix

- Many software packages output a **confusion matrix**:

	actual class (observation)	
predicted class (expectation)	**tp** (true positive) Correct result	**fp** (false positive) Unexpected result
	fn (false negative) Missing result	**tn** (true negative) Correct absence of result

Confusion matrix

- Many software packages output this matrix:

	actual class (observation)	
predicted class (expectation)	**tp** (true positive) Correct result	**fp** (false positive) Unexpected result
	fn (false negative) Missing result	**tn** (true negative) Correct absence of result

- Be careful! Sometimes the format is slightly different, e.g.,

$$\begin{bmatrix} m_{00} & m_{01} \\ m_{10} & m_{11} \end{bmatrix}$$

Common measures

- Accuracy = (TP+ TN) / (TP + FP + FN + TN)
- Precision = True positives / Total number of declared positives
  = TP / (TP+ FP)
- Recall = True positives / Total number of actual positives
  = TP / (TP + FN)

Common measures

- Accuracy = (TP+ TN) / (TP + FP + FN + TN)
- Precision = True positives / Total number of declared positives
  = TP / (TP+ FP)
- Recall = True positives / Total number of actual positives
  = TP / (TP + FN)

Text classification

- Sensitivity is the same as recall.
- Specificity = True negatives / Total number of actual negatives
  = TN / (FP + TN)

Medicine

Common measures

- Accuracy = (TP+ TN) / (TP + FP + FN + TN)
- Precision = True positives / Total number of declared positives
  = TP / (TP+ FP)
- Recall = True positives / Total number of actual positives
  = TP / (TP + FN)

Text classification

- Sensitivity is the same as recall.
- Specificity = True negatives / Total number of actual negatives
  = TN / (FP + TN)

Medicine

- False positive rate = FP / (FP + TN)

Common measures

- Accuracy = (TP+ TN) / (TP + FP + FN + TN)
- Precision = True positives / Total number of declared positives
  = TP / (TP+ FP)

Text classification

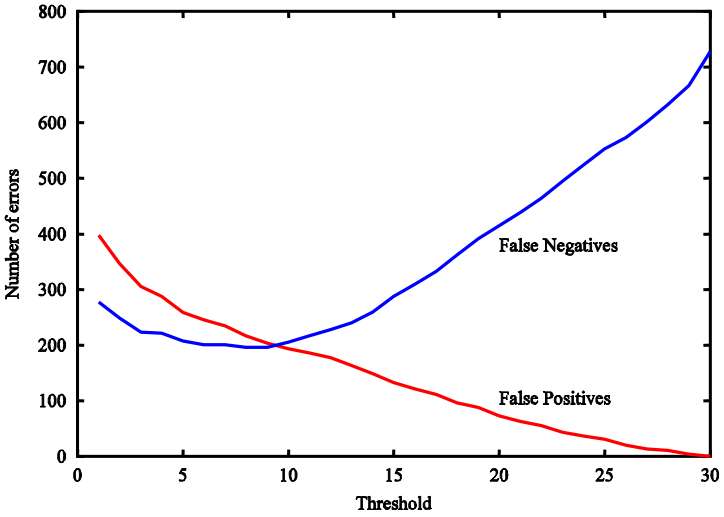
- Recall = True positives / Total number of actual positives
  = TP / (TP + FN)

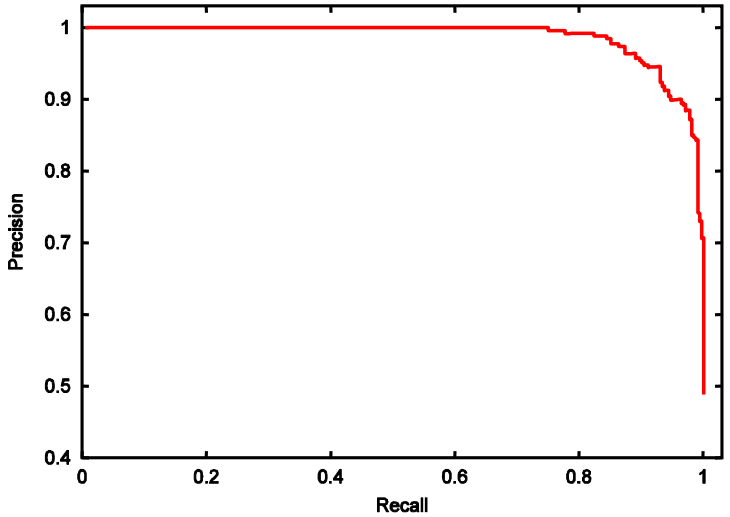
Medicine

- Sensitivity is the same as recall.
- Specificity = True negatives / Total number of actual negatives
  = TN / (FP + TN)
- False positive rate = FP / (FP + TN)
- F1 measure $F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$

Trade-off: False positives and false negatives

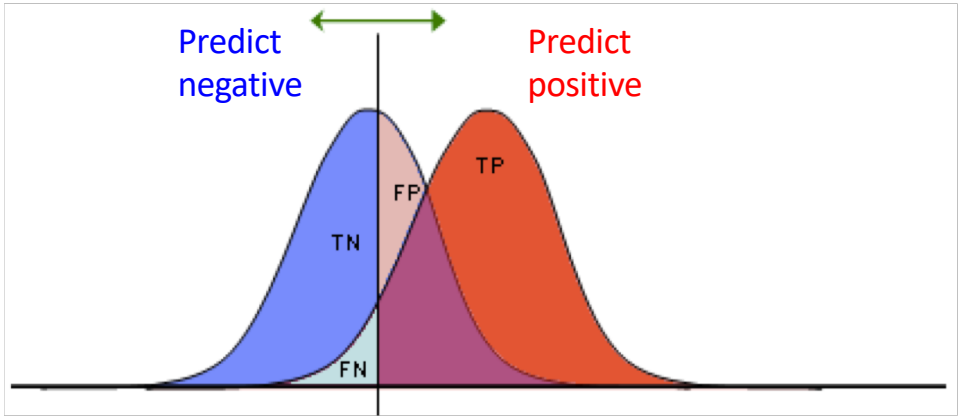
- Often have a trade-off between false positives and false negatives.
  - E.g. Consider 30 different classifiers trained on a class. Classify a new sample as positive if K classifiers output positive. Vary K between 0 and 30.

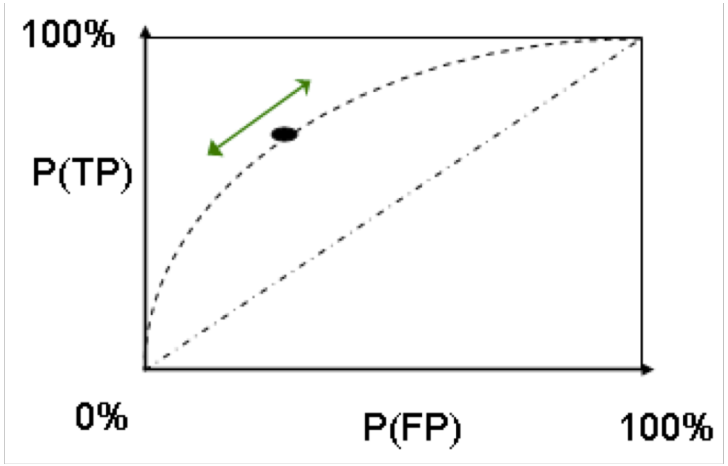




Receiver operating characteristic (ROC)

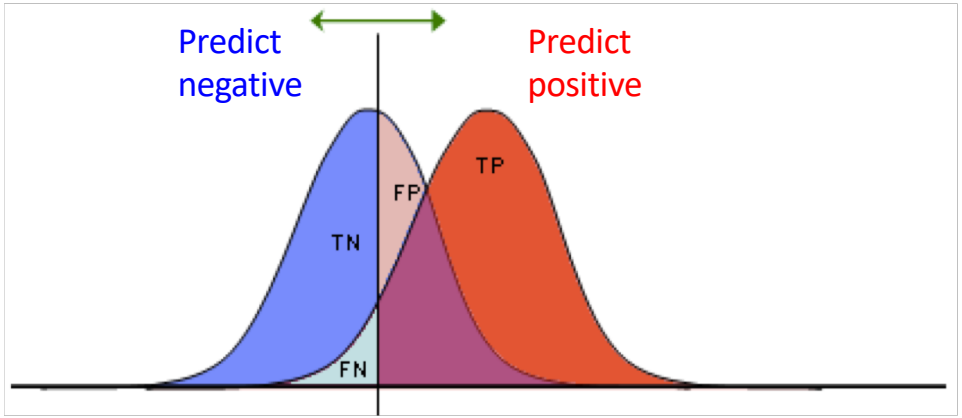
- Consider simple 2-class classification (blue = negative class; red = positive class).
- Consider the decision boundary (vertical line inthe left figure), where you predict Negative on the left of the boundary and predict Positive on the right .
- Changing that boundary defines the ROC curve on the right.

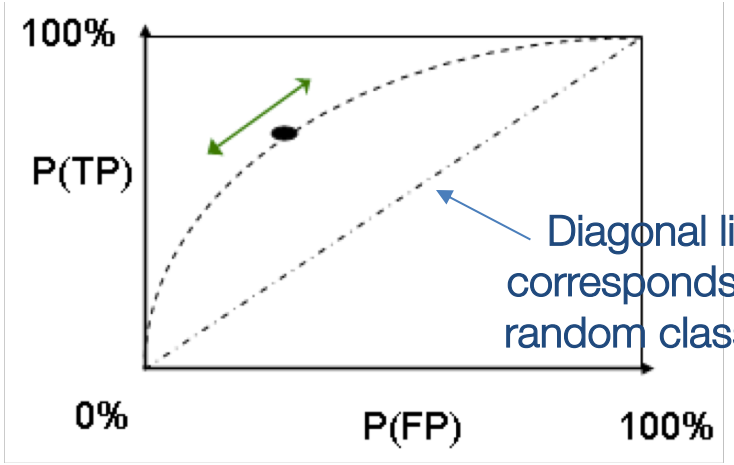




Understanding the ROC curve

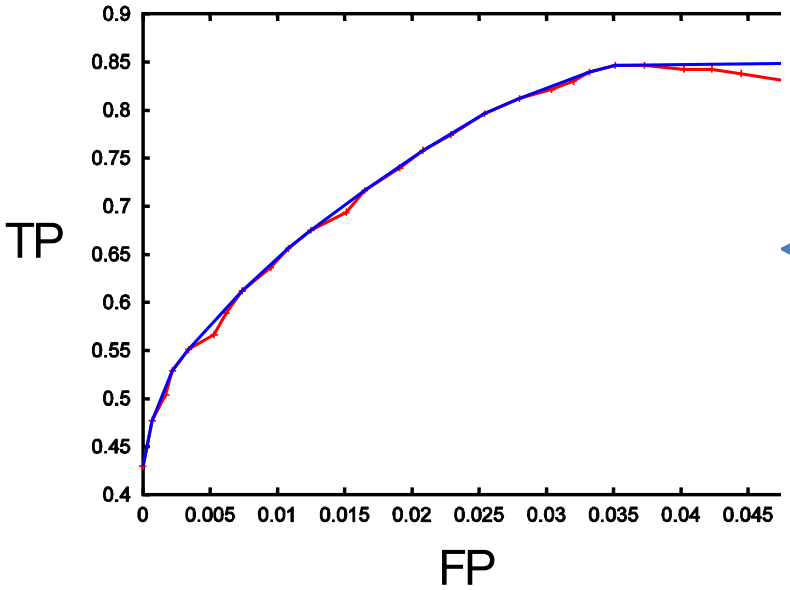
- Consider simple 2-class classification (blue = negative class; red = positive class).
- Consider the decision boundary (vertical line inthe left figure), where you predict Negative on the left of the boundary and predict Positive on the right .
- Changing that boundary defines the ROC curve on the right.





Building the ROC curve

- In many domains, the empirical ROC curve will be non-convex (red line). Take the convex hull of the points (blue line).

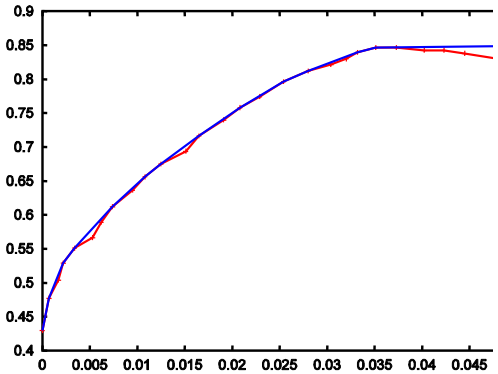


To build the ROC curve:

1. Train a classifier.
2. Vary the decision boundary.
3. Compute FP and TP rate for each different decision boundary.

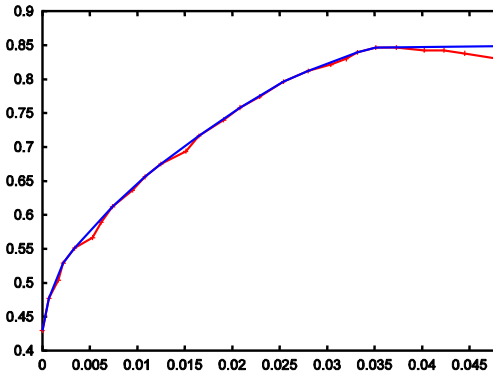
Using the ROC curve

- To compare 2 algorithms over a range of classification thresholds, consider the Area Under the (ROC) Curve (AUC).
  - A perfect algorithm has AUC=1.
  - A random algorithm has AUC=0.5.
  - Higher AUC doesn't mean all performance measures are better.



Using the ROC curve

- To compare 2 algorithms over a range of classification thresholds, consider the Area Under the (ROC) Curve (AUC).
  - A perfect algorithm has AUC=1.
  - A random algorithm has AUC=0.5.
  - Higher AUC doesn't mean all performance measures are better.



Intuition: AUC is equivalent to the probability of ranking a random positive example higher than a random negative example!

What about regression?

- Evaluation for regression is generally more straightforward than classification, but there are still choices!
- Common examples:
  - Mean Squared Error: $\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$
  - Root Mean Squared Error: $\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)}$
  - Mean Absolute Error: $\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$
- (R)MSE penalizes larger errors more (e.g., being off by 10 is *more* than twice as bad as being off by 5).

Our goal: Minimize validation error

- Overly simple model:
  - High training error and high test error.
- Overly complex model:
  - Low training error but high test error.

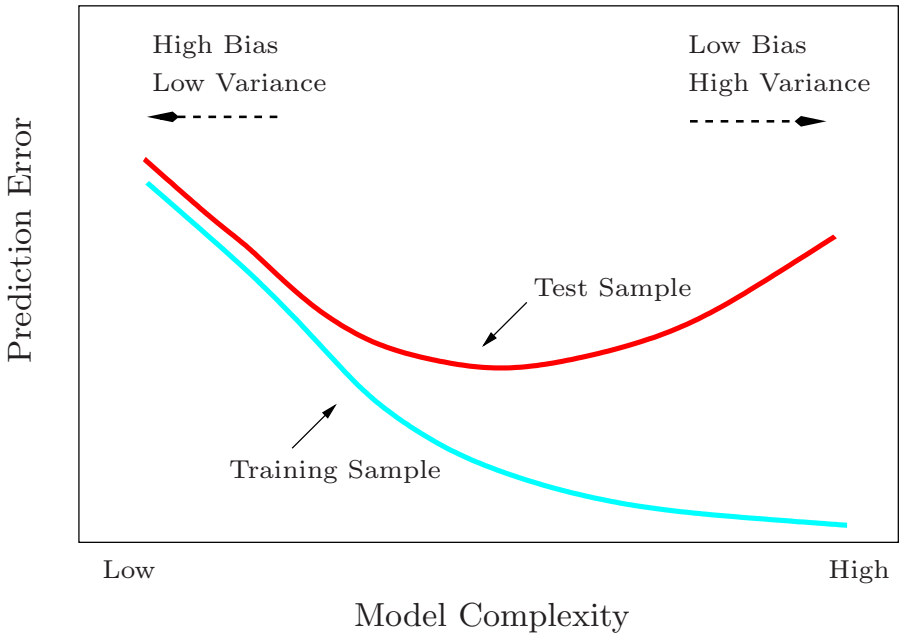


FIGURE 2.11. *Test and training error as a function of model complexity.*

Our goal: Minimize validation error

- Overly simple model:
  - High bias
- Overly complex model:
  - High variance

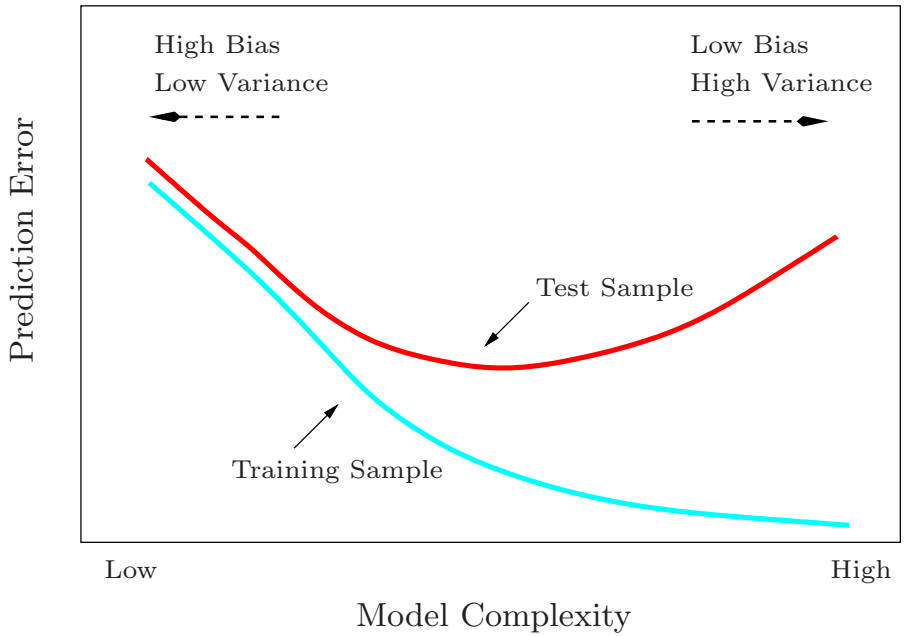


FIGURE 2.11. *Test and training error as a function of model complexity.*

Bias vs variance in machine learning

- Assume that $y = f(x) + \varepsilon$ where ε is zero mean and finite variance σ^2
- Then for **any** learned model we can decompose the error as:

$$\mathbb{E}[(y - \hat{f}(\mathbf{x}))^2] = (\text{Bias}[\hat{f}(\mathbf{x})])^2 + \text{Var}[\hat{f}(\mathbf{x})] + \sigma^2$$

- where

$$\text{Bias}[\hat{f}(\mathbf{x})] = \mathbb{E}[\hat{f}(x)] - f(x)$$

$$\text{Var}[\hat{f}(\mathbf{x})] = \mathbb{E}[\hat{f}(x)^2] - \mathbb{E}[\hat{f}(x)]^2$$

Bias vs variance in machine learning

- Assume that $y = f(x) + \varepsilon$ where ε is zero mean and finite variance σ^2
- Then for **any** learned model we can decompose the error as:

$$\mathbb{E}[(y - \hat{f}(\mathbf{x}))^2] = (\text{Bias}[\hat{f}(\mathbf{x})])^2 + \text{Var}[\hat{f}(\mathbf{x})] + \sigma^2$$

- where

$$\text{Bias}[\hat{f}(\mathbf{x})] = \mathbb{E}[\hat{f}(x)] - f(x)$$

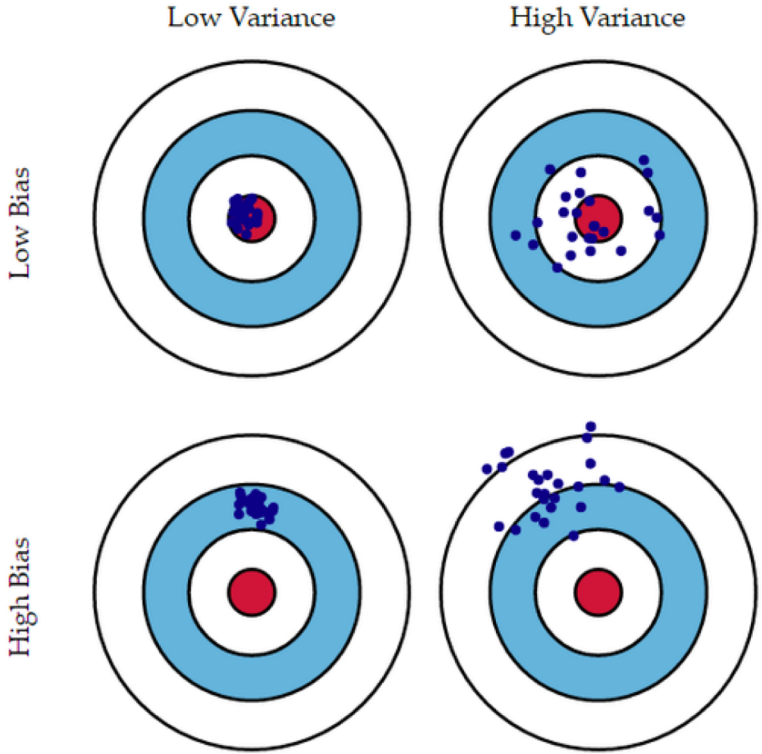
How far off is our mean prediction?

$$\text{Var}[\hat{f}(\mathbf{x})] = \mathbb{E}[\hat{f}(x)^2] - \mathbb{E}[\hat{f}(x)]^2$$

How wildly do our predictions vary?

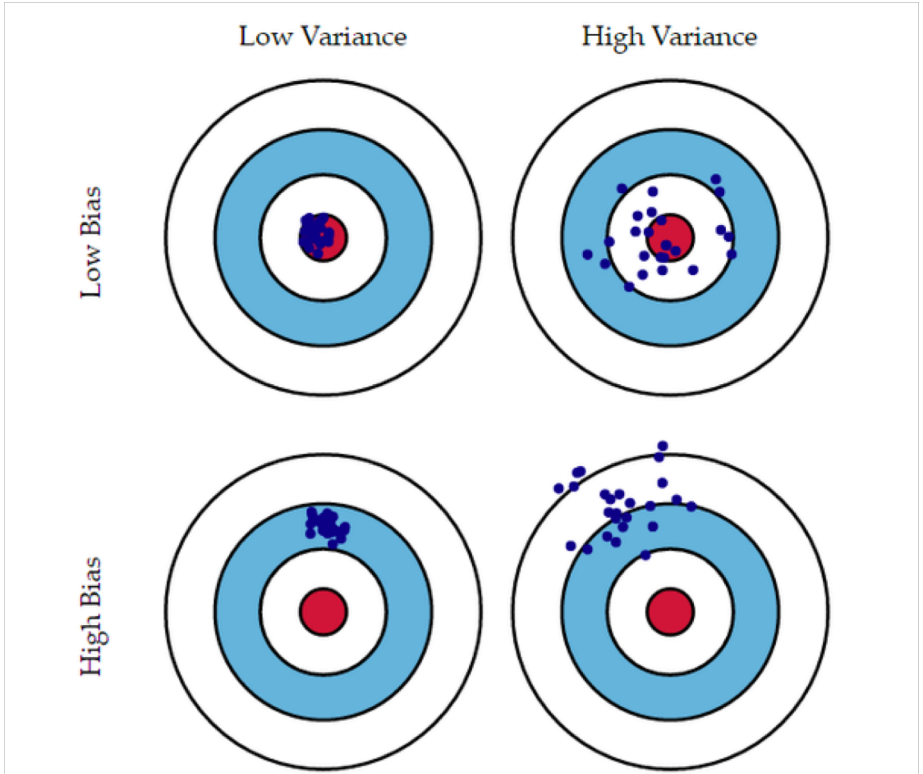
Bias vs variance in machine learning

- Overly simple model:
  - High bias
  - Underfitting
- Overly complex model:
  - High variance
  - Overfitting



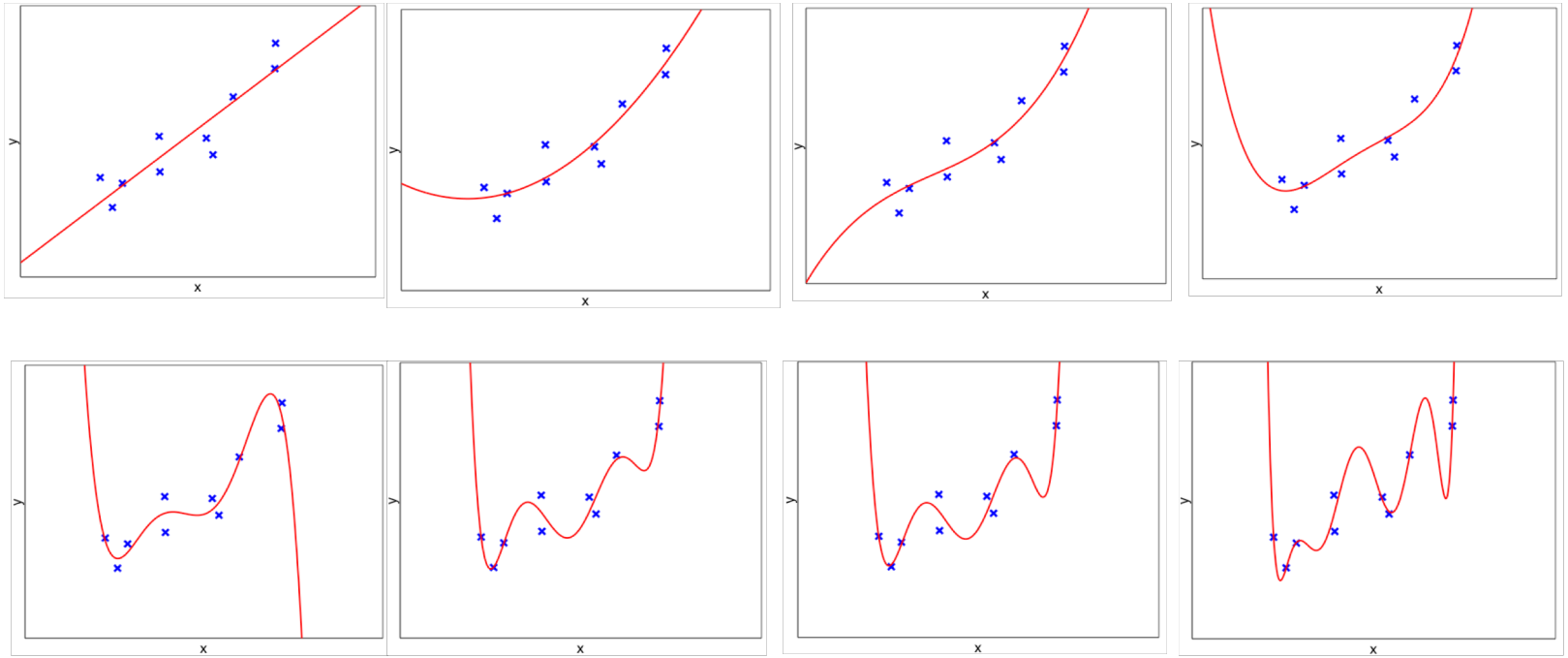
Overfitting, bias, and variance

- What to do when we are overfitting?
- Remove some features.
- Simplify the model.
- Reduce the variance at the cost of some bias!
- This is called **regularization.**



Overfitting, bias, and variance

Which models have higher bias, which have higher variance?



Ridge regression (aka L2-regularization)

- Constrains the weights by imposing a penalty on their size:

$$\hat{\mathbf{w}}^{\text{ridge}} = \text{argmin}_{\mathbf{w}} \{ \sum_{i=1:n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \lambda \sum_{j=0:m} w_j^2 \}$$

  where λ can be selected manually, or by cross-validation.

- Do a little algebra to get the solution: $\hat{\mathbf{w}}^{\text{ridge}} = (X^T X + \lambda I)^{-1} X^T Y$
  - The ridge solution is not equivariant under scaling of the data, so typically need to normalize the inputs first.
  - Ridge gives a smooth solution, effectively shrinking the weights, but drives few weights to 0.

Ridge regression (aka L2-regularization)

- Constrains the weights by imposing a penalty on their size:

$$\hat{\mathbf{w}}^{\text{ridge}} = \text{argmin}_{\mathbf{w}} \{ \sum_{i=1:n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \lambda \sum_{j=0:m} w_j^2 \}$$

  where λ can be selected manually, or by cross-validation.

- I.e., $\text{Err}(\mathbf{w}) = \sum_{i=1:n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \lambda ||\mathbf{w}||_2$
- Can also simply add the penalty to gradient descent:

$$\partial \text{Err}(\mathbf{w}) / \partial \mathbf{w} = 2(\mathbf{X}^T \mathbf{X} \mathbf{w} - \mathbf{X}^T \mathbf{Y}) + 2 \lambda \mathbf{w}$$

Usual gradient for linear regression

Gradient from the L2-penalty term

Lasso regression (aka L1-regularization)

- Constrains the weights by penalizing the absolute value of their size:

$$\hat{\mathbf{w}}^{\text{lasso}} = \text{argmin}_{\text{W}} \{ \sum_{\text{i=1:n}} (\text{y}_\text{i} - \mathbf{w}^\text{T}\mathbf{x}_\text{i})^2 + \lambda \sum_{\text{j=1:m}} |\text{w}_\text{j}| \}$$

- Now the objective is non-linear in the output y, and there is no closed-form solution. Need to solve a quadratic programming problem instead.
  - More computationally expensive than Ridge regression.
  - Effectively sets the weights of less relevant input features to zero.

Lasso regression (aka L1-regularization)

- Constrains the weights by penalizing the absolute value of their size:

$$\hat{\mathbf{w}}^{\text{lasso}} = \text{argmin}_{\text{W}} \{ \sum_{\text{i=1:n}} (\text{y}_\text{i} - \mathbf{w}^\text{T}\mathbf{x}_\text{i})^2 + \lambda \sum_{\text{j=1:m}} |\text{w}_\text{j}| \}$$

- I.e., $\text{Err}(\text{w}) = \sum_{\text{i=1:n}} (\text{y}_\text{i} - \mathbf{w}^\text{T}\mathbf{x}_\text{i})^2 + \lambda ||\mathbf{w}||_1$
- Can also simply add the penalty to gradient descent:

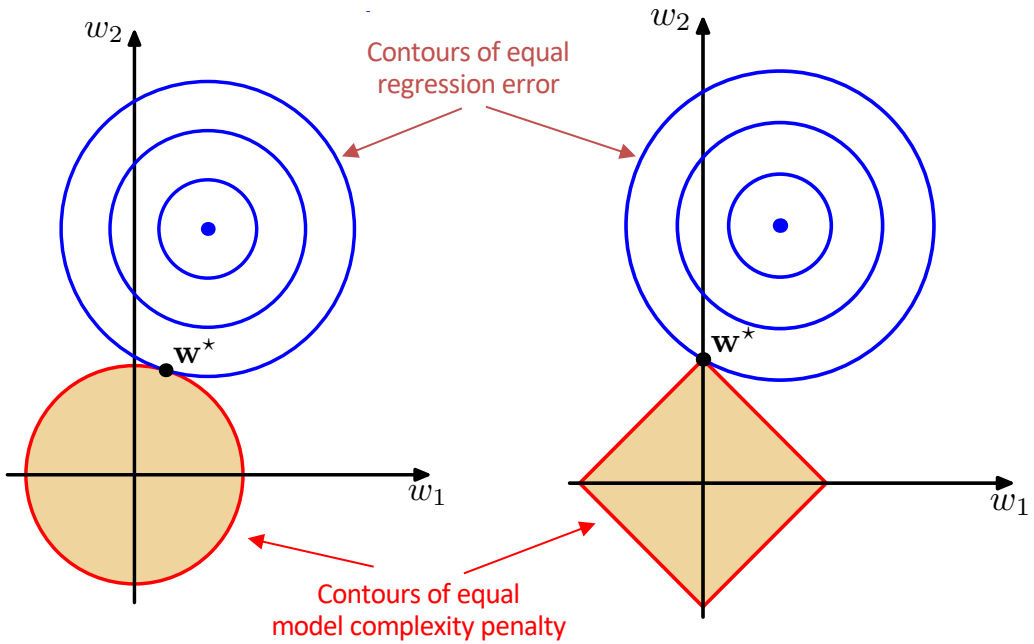
$$\partial \text{Err}(\mathbf{w}) / \partial \mathbf{w} = 2(\text{X}^\text{T}\text{X}\mathbf{w} - \text{X}^\text{T}\text{Y}) + \lambda \, \text{sign}(\mathbf{w})$$

Usual gradient for linear regression

(Sub)gradient of the L1-penalty term. sign(x)=1 if x >0, sign(x)=-1 if x < 0, and sign(x)=0 if x=0.

Comparing Ridge and Lasso

- Ridge regression will tend to lower all weights.
- Lasso will tend to set some weights to 0.
- Combining Lasso and Ridge regression is called the "elastic" net.



Bias, variance, and regularization

- Regularization (e.g, Ridge and Lass) decrease variance at the expense of some bias.

- Regularization reduces overfitting but can cause underfitting.

- The L2 and L1 penalties can be added to any model (e.g., logistic regression). Easy to incorporate with gradient descent.

The concept of inductive bias

- Statistical “bias” is not a bad thing in ML!
- It is all about having the right **inductive bias.**
- ML is all about coming up with different models that have different inductive biases.
- Optional reading: “The Need for Biases in Learning Generalizations"