Dall-E

K-Nearest Neighbor and ML Basics

Applied Machine Learning
Derek Hoiem

Today's Lecture

- Overview of machine learning process
- K-Nearest Neighbor Algorithm
- Measuring and understanding error
- Example applications
  - HW 1 overview
  - Deep Face

Machine learning model maps from features to prediction

$$f(x) \rightarrow y$$

Features — Prediction

Examples

- Classification
  - Is this a dog or a cat?
  - Is this email spam or not?

- Regression
  - What will the stock price be tomorrow?
  - What will be the high temperature tomorrow?

- Structured prediction
  - What is the pose of this person?

Learning has three stages

- Training: optimize model parameters
- Validation: intermediate evaluations to design/select model
- Test: final performance evaluation

Training: the model is fit to data to minimize a loss or maximize an objective function

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, Loss(f(X;\theta), Y)$$

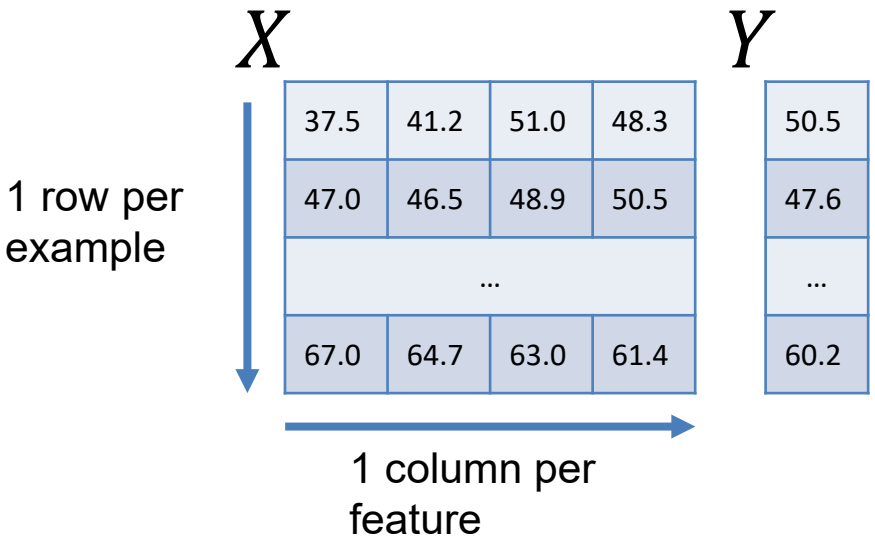
Model parameters that minimize loss

Features of all training examples

"Ground Truth" predictions of all training examples

Example

Learn to predict next day's temperature given preceding days' temperatures



Loss: sum squared error

$$Loss(f(X;\theta), Y) = \sum_i (f(X_i;\theta) - y_i)^2$$

Model: linear

$$f(X_i;\theta) = AX_i + b$$

Optimization via ordinary least squares regression

Model design and “hyper parameter” tuning is performed using a validation set

- Select model ~~linear regression~~ neural network
- Set training parameters
  - Feature selection
  - Learning rate, regularization parameters, …

- Sometimes, there are clear “train”, “val”, and “test” sets. Other times, you need to split “train” into a train set for learning parameters and a val set for checking model performance

Testing: The effectiveness of the model is evaluated on a held out test set

- "Held out": not used in training; ideally not viewed by developers, e.g. in a private test server

- Common performance measures

  Classification error: $\frac{1}{\mathrm{N}}\sum_i f(X_i) \neq y_i$ (for classification model, y_i is target/true label)

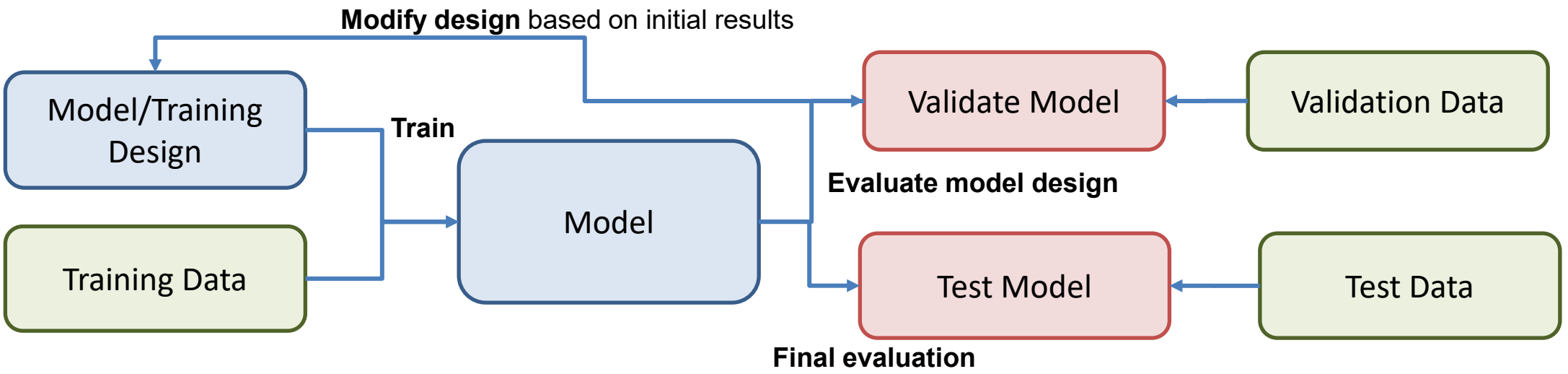
  Cross-entropy: $-\frac{1}{\mathrm{N}}\sum_i \log f(y = y_i|X_i)$ (for probabilistic model)

  RMSE: $\sqrt{\frac{1}{\mathrm{N}}\sum_i (f(X_i) - y_i)^2}$ (regression measure)

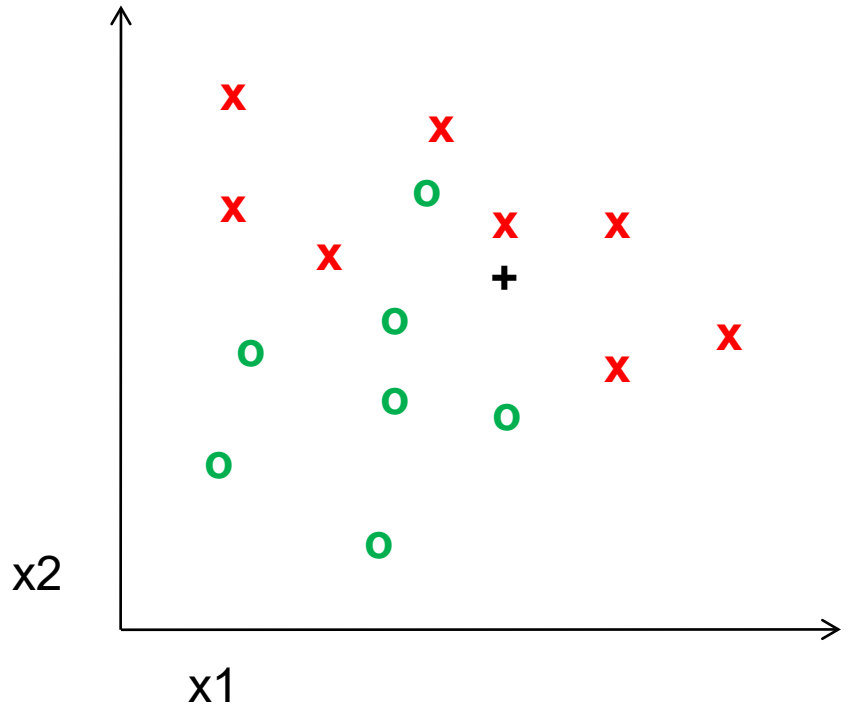
  R^2: $1 - \sum_i (f(X_i) - y_i)^2 \,/\, \sum_i (y_i - \overline{\boldsymbol{y}})^2$ (unitless regression measure; $\overline{\boldsymbol{y}}$ is expectation/mean/avg)

- In machine learning research, usually data is collected once and then randomly sampled into train and test partitions
  - Train and test samples are "i.i.d.", independent and identically distributed
  - In many real-world applications, the input to the model in deployment comes from a different distribution than training

Recap of training and evaluation



What class do you think the ‘+’ belongs to?



Key principle of machine learning

Given feature/target pairs $(X_1, y_1), \ldots, (X_n, y_n)$:

if X_i is similar to X_j, then y_i is probably similar to y_j

With variations on how you define similarity and make predictions based on multiple similar examples, this principle underlies virtually all ML algorithms

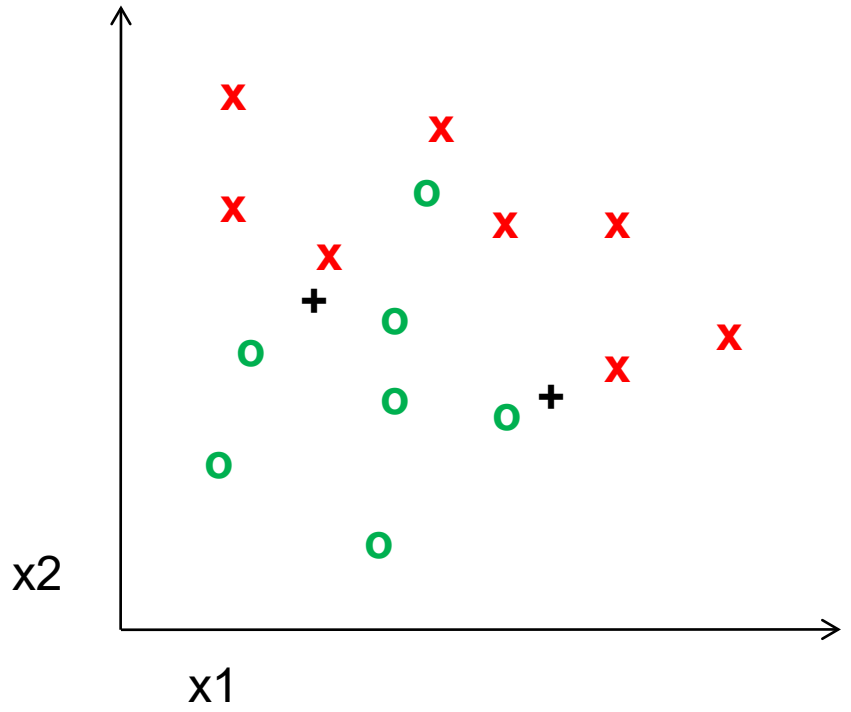
Nearest neighbor algorithm

For given test features, assign the label / target value of the most similar training features

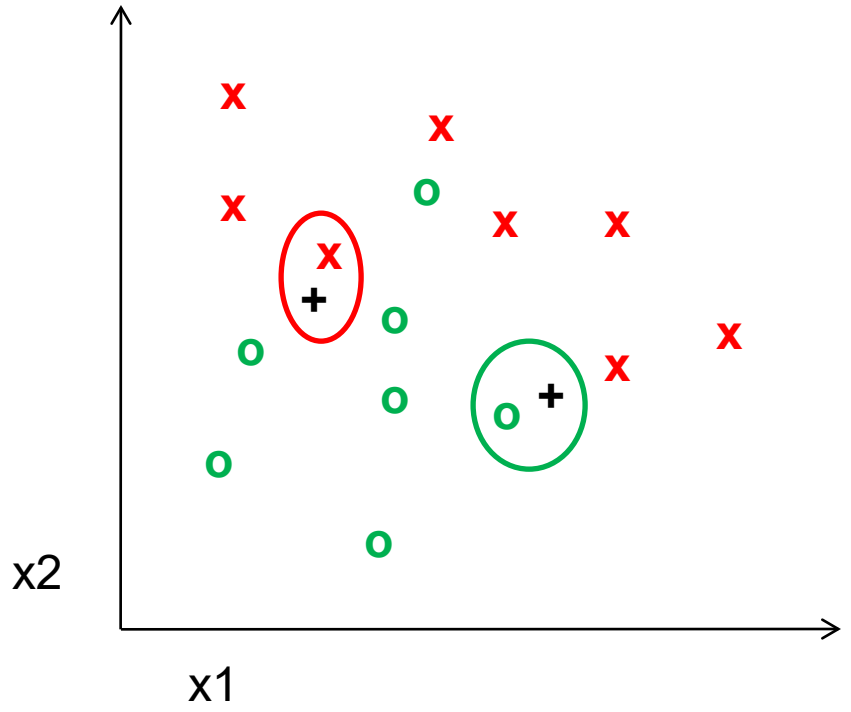
1. $i^* = \underset{i}{\operatorname{argmin}}\, distance(X_{train}[i], X_{test})$
2. $y_{test} = y_{train}[i^*]$

Distance function is up to designer. Simplest is L2 distance.

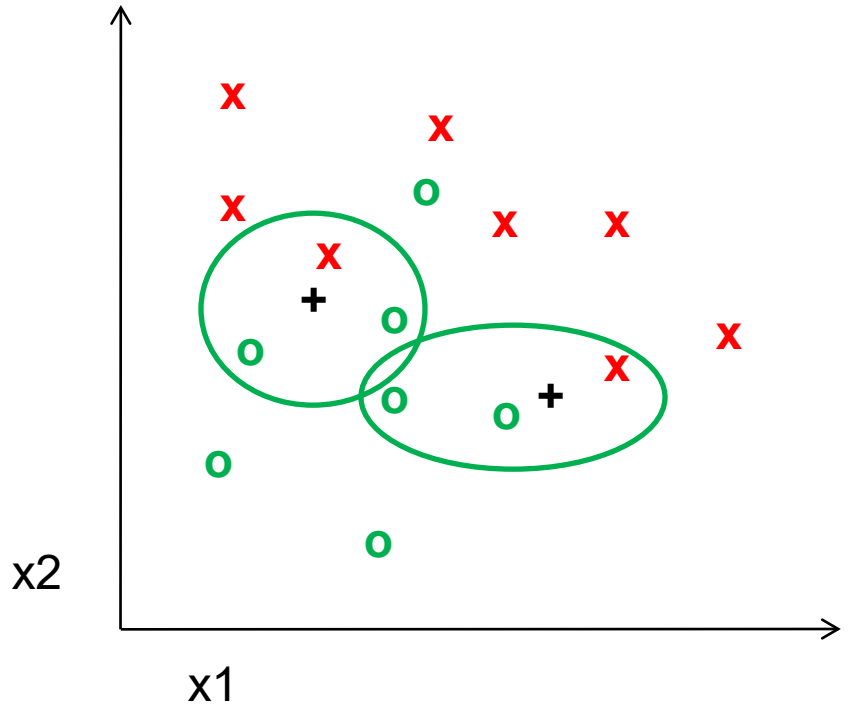
K-nearest neighbor: predict based on K closest training samples



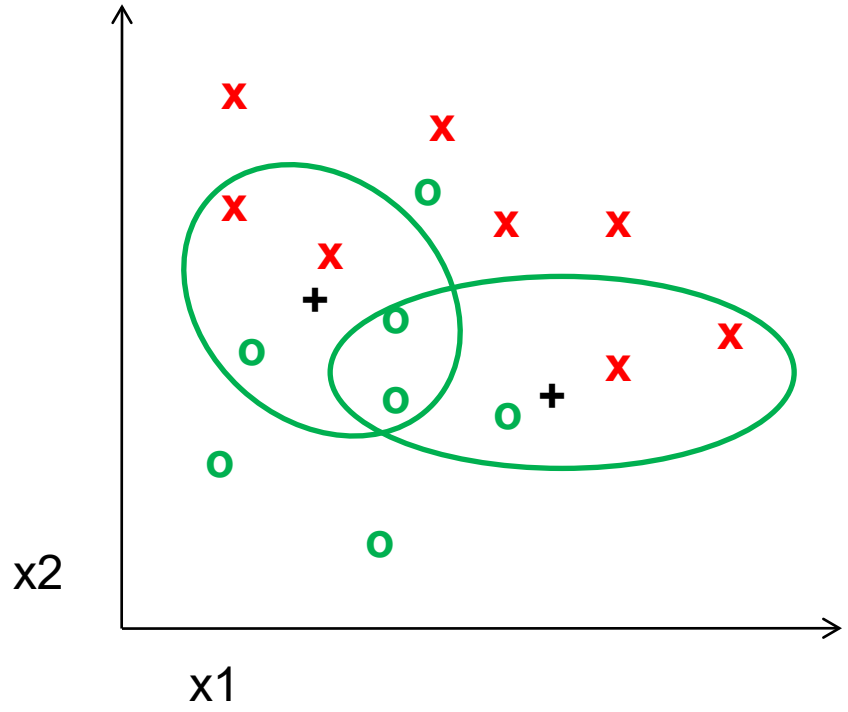
1-nearest neighbor



3-nearest neighbor



5-nearest neighbor



KNN Distance Function

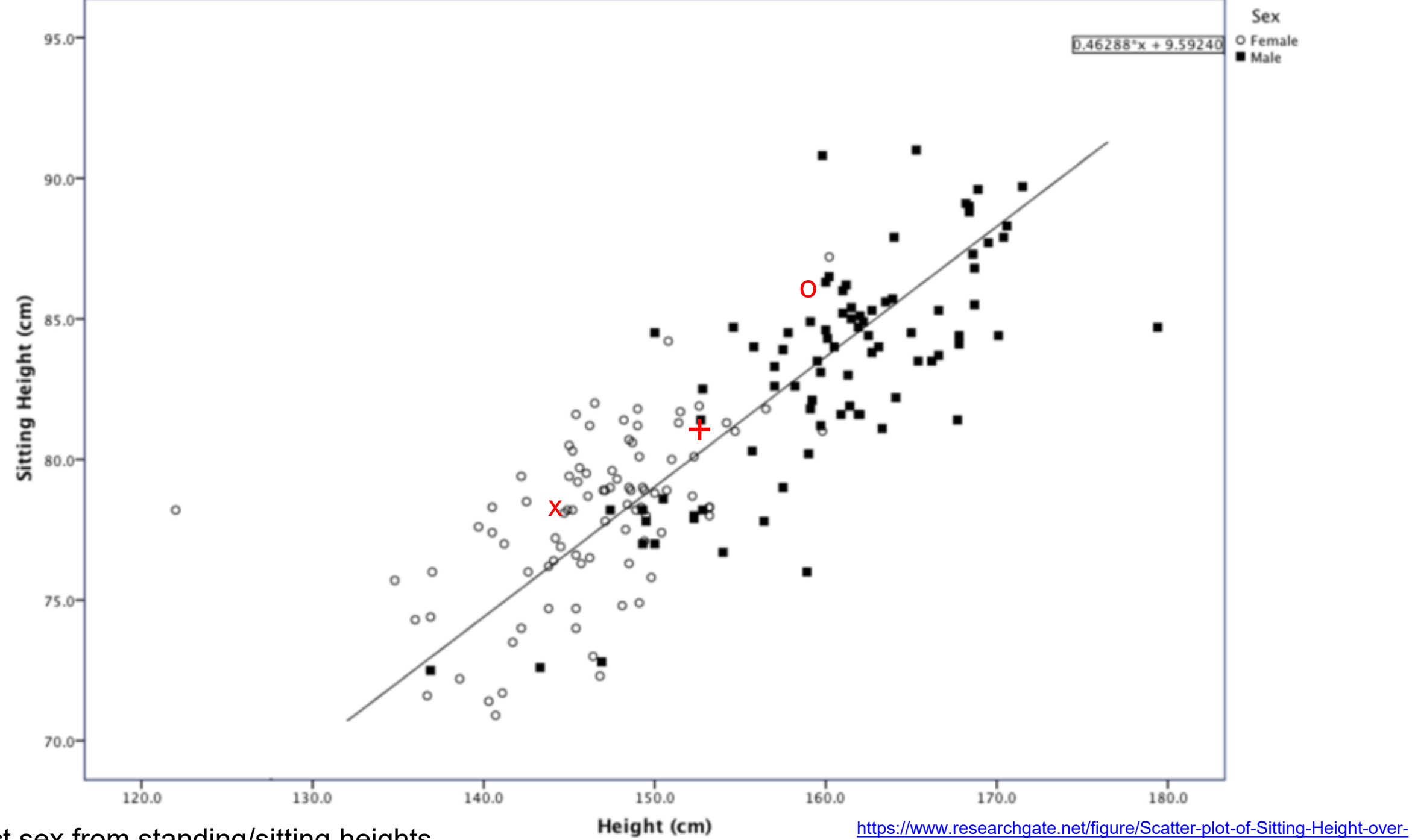
- Euclidean or L2 norm: $\|\boldsymbol{x} - \boldsymbol{t}\|_2 = \sqrt{\sum_k (x_k - t_k)^2}$
  - Assumes all dimensions are equally scaled
  - Dominated by biggest differences
- Citi Block or L1 norm: $\|\boldsymbol{x} - \boldsymbol{t}\|_1 = \sum_k |x_k - t_k|$
  - Assumes all dimensions are equally scaled
  - Less sensitive to very large differences along one dimension
- Mahalanobis distance: $d_M(\boldsymbol{x}, \boldsymbol{t}) = \sqrt{(\boldsymbol{x} - \boldsymbol{t})^T \sum^{-1} (\boldsymbol{x} - \boldsymbol{t})}$
  - Normalized by inverse feature covariance matrix: “whitening”
  - When diagonal covariance is assumed, this is equivalent to scaling each dimension by $1/\sigma_k$

$\boldsymbol{x}$ and $\boldsymbol{t}$ are training and test sample feature vectors

KNN Classification vs Regression

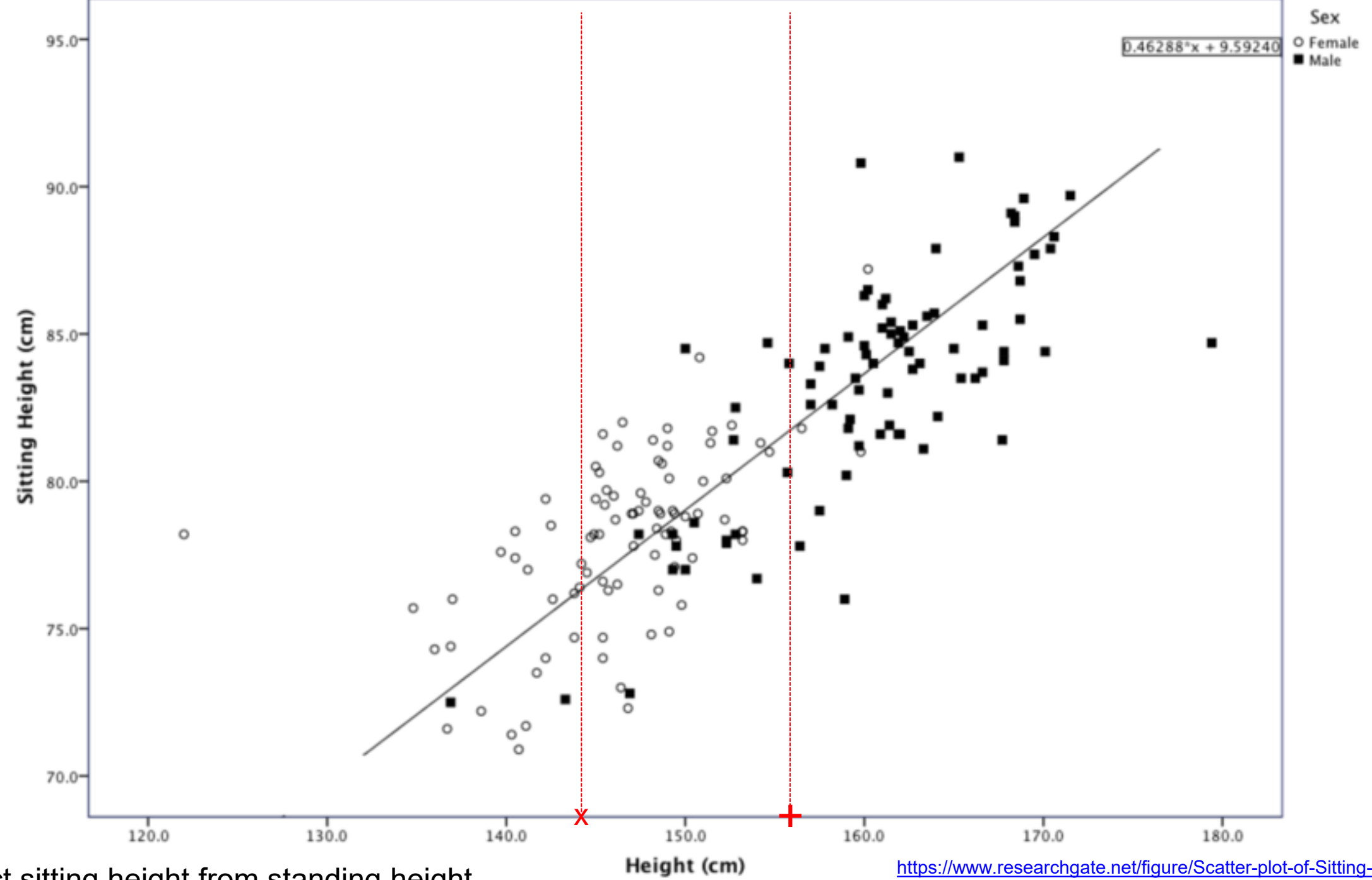
- For classification, prediction is usually the mode or most common class of the returned labels

- For regression, prediction is usually the arithmetic mean (average, informally) of the returned values



Predict sex from standing/sitting heights

https://www.researchgate.net/figure/Scatter-plot-of-Sitting-Height-over-Height-for-males-and-females_fig3_301724988



Predict sitting height from standing height

https://www.researchgate.net/figure/Scatter-plot-of-Sitting-Height-over-Height-for-males-and-females_fig3_301724988

KNN Classification Demos

- https://lecture-demo.ira.uka.de/knn-demo/#
- http://vision.stanford.edu/teaching/cs231n-demos/knn/

Comments on K-NN

- Simple: an excellent baseline and sometimes hard to beat
  - Naturally scales with data: it may be the only choice when you have one example per class, and is still often achieves good performance when you have many
  - Higher K gives smoother functions

- Sloww… but there are tricks to speed it up, e.g.

can be precomputed

  - $\underset{i}{\operatorname{argmin}} \|x_i - x_t\|_2 = \underset{i}{\operatorname{argmin}} \left(x_i^T x_i - 2 x_i x_t + x_t^T x_t \right) = \underset{i}{\operatorname{argmin}} \left(x_i^T x_i - 2 x_i x_t \right)$
  - Can use approximate nearest neighbor methods like FLANN (will come to those later)

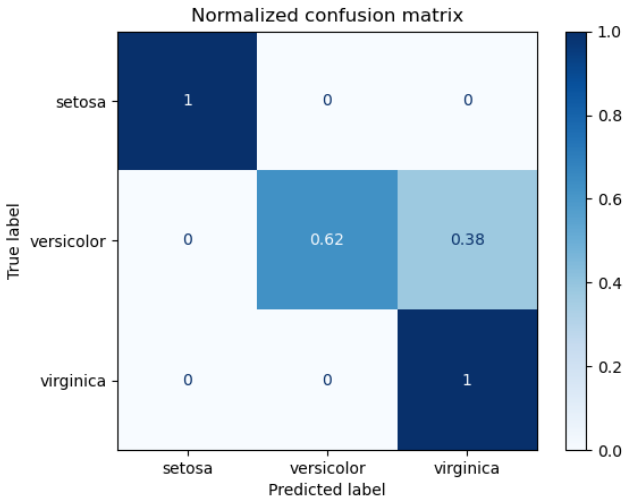
- No training time (unless you learn a distance function)

- With infinite examples, 1-NN provably has error that is at most twice Bayes optimal error (but we never have infinite examples)

How do we measure and analyze classification error?

- Classification error: $\frac{1}{\mathrm{N}}\sum_i f(X_i) \neq y_i$
  - Percent of examples that are wrongly predicted

- Confusion matrix: joint/conditional distribution of predicted and true labels
  - Can be a count or probability
  - Practice varies whether “Predict” or “True” is the y-axis. Need to label.



https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html

Measuring error example

True	Predicted
Y	Y
N	Y
Y	Y
Y	N
N	N
N	Y
N	N

Classification Error:

Confusion Matrix:

Count

		Predicted	
		N	Y
True	N		
	Y		

P(predicted | true)

		Predicted	
		N	Y
True	N		
	Y		

How do we measure and analyze regression error?

- Root mean squared error
$$\sqrt{\frac{1}{\mathrm{N}}\sum_i (f(X_i) - y_i)^2}$$

- Mean absolute error $\frac{1}{\mathrm{N}}\sum_i |f(X_i) - y_i|$

- R^2: $1 - \frac{\sum_i (f(X_i) - y_i)^2}{\sum_i (y_i - \bar{y})^2}$ (unexplained variance) / (total variance)

- RMSE/MAE are unit-dependent measures of accuracy, while R^2 is a unitless measure of the fraction of explained variance

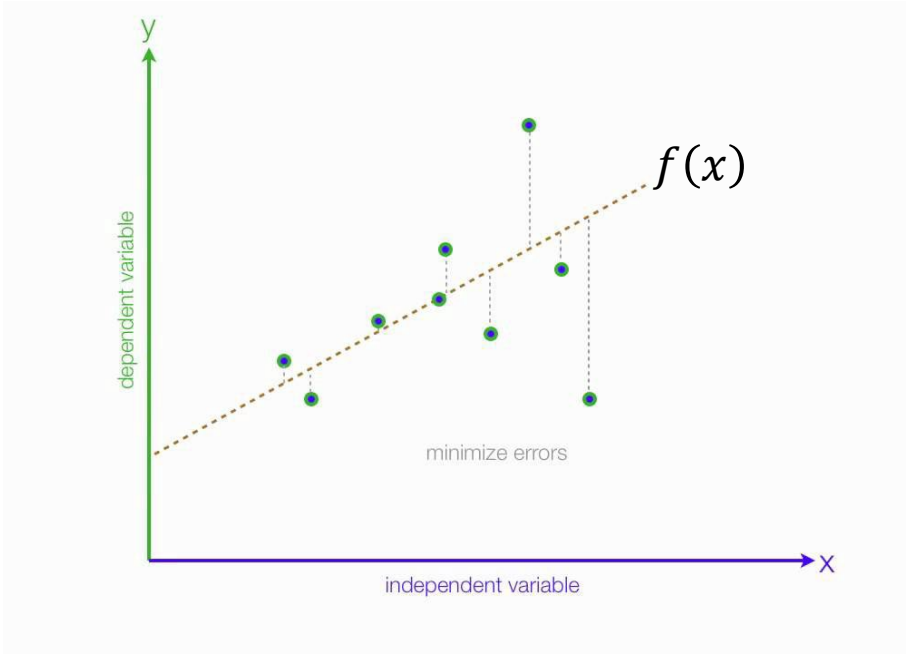


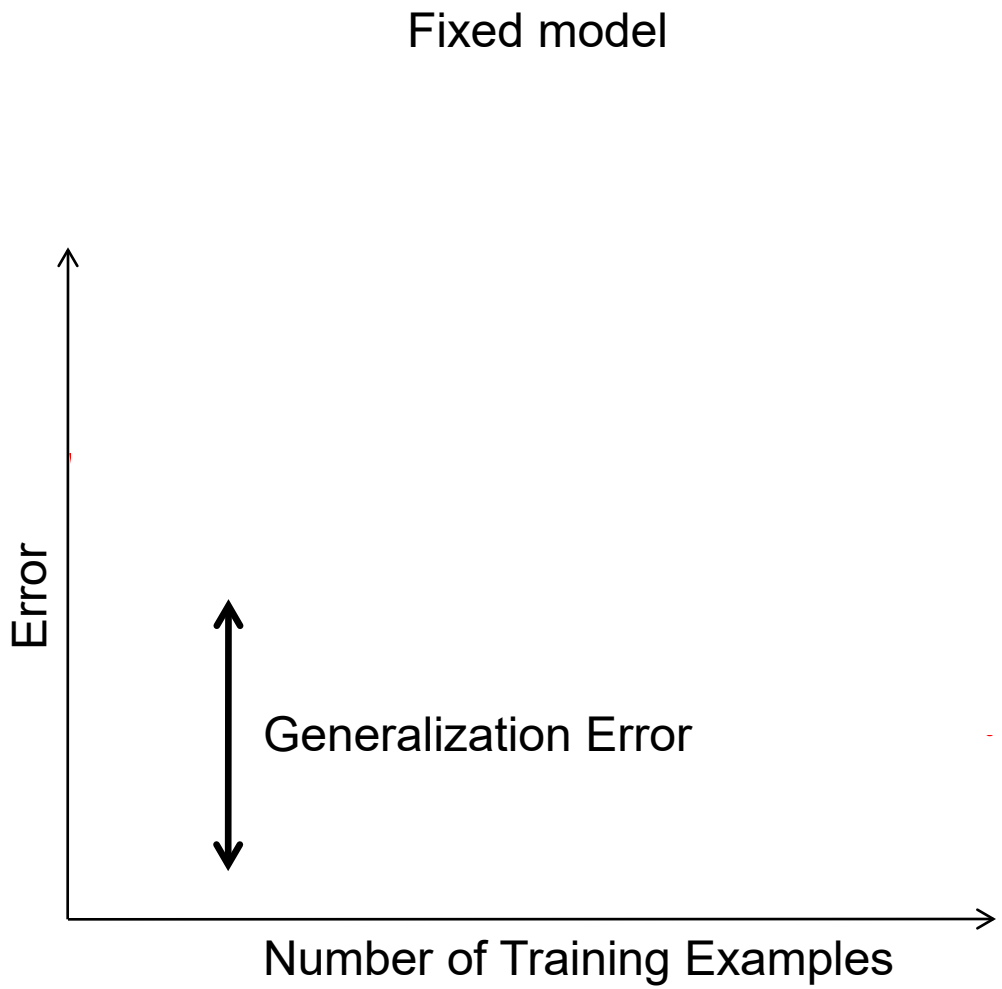
Fig: https://medium.com/analytics-vidhya/mae-mse-rmse-coefficient-of-determination-adjusted-r-squared-which-metric-is-better-cd0326a5697e

Sources of test error for a trained model

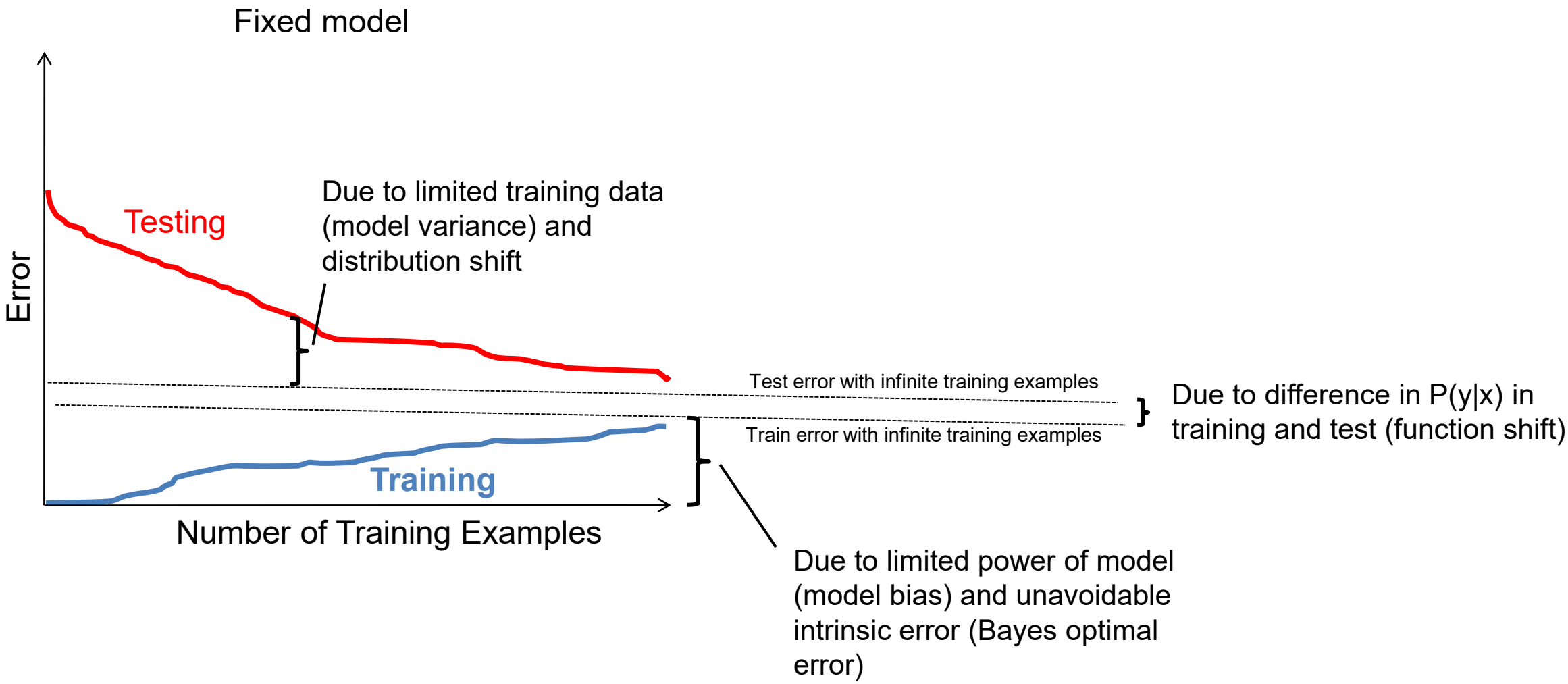
- **Intrinsic**: sometimes it is not possible to achieve zero error given available features (e.g. handwriting, weather prediction)
  - Bayes optimal error: The error if the true function P(y|x) is known
- **Model Bias**: the model is limited so that Bayes optimal error cannot be achieved for an infinite training set
- **Model Variance**: given finite training data, different parameters and predictions would result from different samplings of data
- **Distribution Shift**: some examples are more likely in test than training, i.e. true P(x) is different for train and test
  - E.g., datasets are collected at different times and frequency of content has changed
- **Function Shift**: P(y|x) is changed between train and test
  - E.g., the predicted answer to “What is your favorite TV show?” changes over time

Others: imperfect optimization, final performance measure is different than training loss

Effect of Training Size



Sources of error and training size



Something to think about…

Why is it important to have a validation set? Why not simply evaluate all your trained models on the test set and then choose the best?

HW 1 Preview

KNN Usage Example: Deep Face

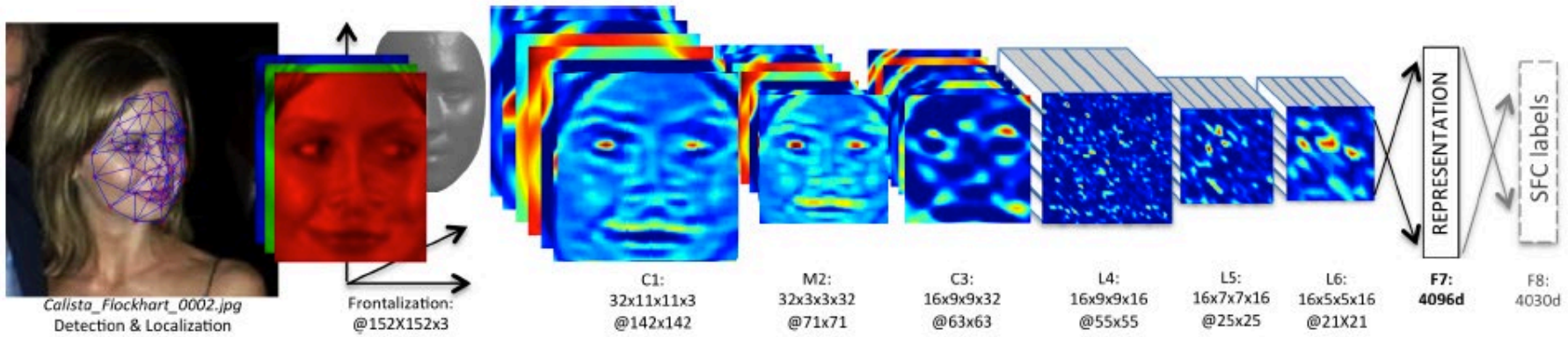
DeepFace: Closing the Gap to Human-Level Performance in Face Verification

Yaniv Taigman Ming Yang Marc'Aurelio Ranzato Lior Wolf

Facebook AI Research
Menlo Park, CA, USA
{yaniv, mingyang, ranzato}@fb.com

Tel Aviv University
Tel Aviv, Israel
wolf@cs.tau.ac.il

CVPR 2014



1. Detect facial features
2. Align faces to be frontal
3. Extract features using deep network while training classifier to label image into person (dataset based on employee faces)
4. In testing, extract features from deep network and use nearest neighbor classifier to assign identity

- Performs similarly to humans in the LFW dataset (labeled faces in the wild)
- Can be used to organize photo albums, identifying celebrities, or alert user when someone posts an image of them
- If this is used in a commercial deployment, what might be some unintended consequences?
- This algorithm is used by Facebook (though with expanded training data)

KNN Summary

- Key Assumptions
  - Samples with similar input features will have similar output predictions
  - Depending on distance measure, may assume all dimensions are equally important
- Model Parameters
  - Features and predictions of the training set
- Designs
  - K (number of nearest neighbors to use for prediction)
  - How to combine multiple predictions if K > 1
  - Feature design (selection, transformations)
  - Distance function (e.g. L2, L1, Mahalanobis)
- When to Use
  - Few examples per class, many classes
  - Features are all roughly equally important
  - Training data available for prediction changes frequently
  - Can be applied to classification or regression, with discrete or continuous features
  - Most powerful when combined with feature learning
- When Not to Use
  - Many examples are available per class (feature learning with linear classifier may be better)
  - Limited storage (cannot store many training examples)
  - Limited computation (linear model may be faster to evaluate)

Things to remember

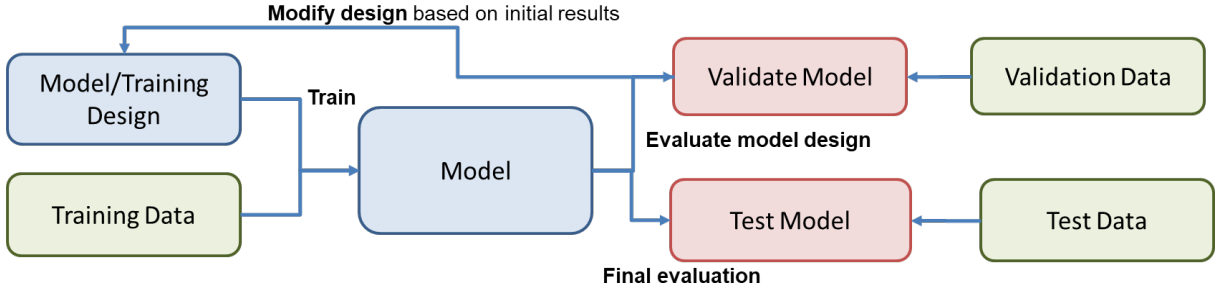
- Supervised machine learning involves:
  1. Fitting parameters to a model using training data
  2. Refining the model based on validation performance
  3. Evaluating the final model on a held out test set

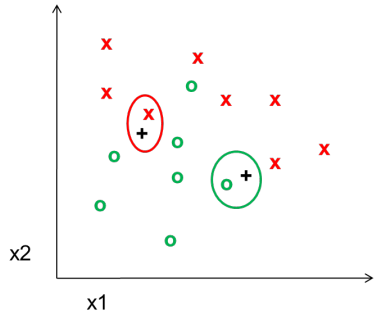
- KNN is a simple but effective classifier/regressor that predicts the label of the most similar training example(s)

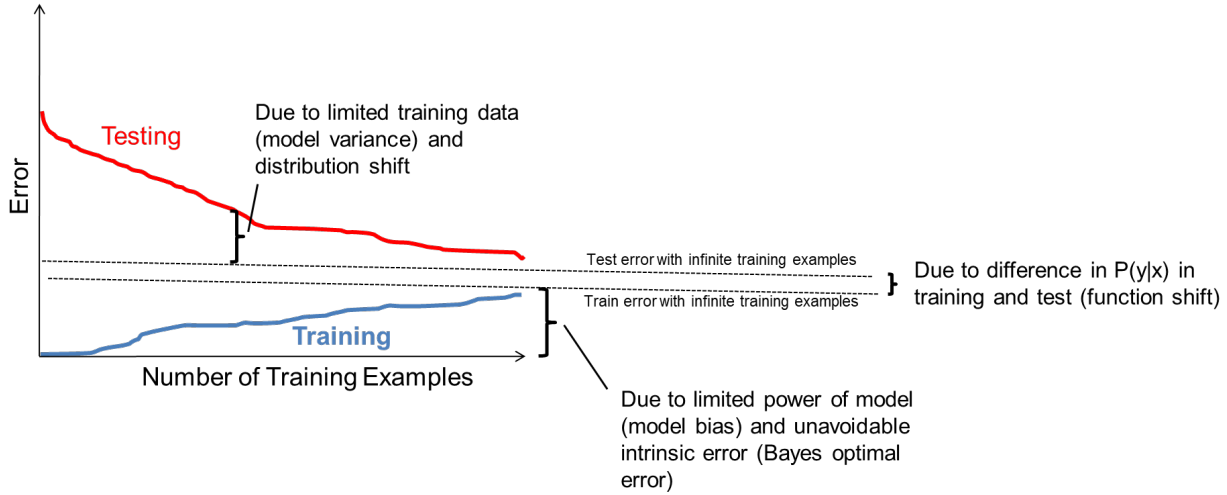
- With more samples, fitting the training data becomes harder, but test error is expected to decrease

- Test errors have many sources
  - intrinsic to problem
  - model bias / limited power
  - model variance / limited training data
  - differences in training and test distributions

- Model design and fitting is just one part of a larger process in collecting data, developing, and deploying models







Next week

- Probabilistic models and Naïve Bayes
- Linear and Logistic Regression