Recap and Review – Exam 1

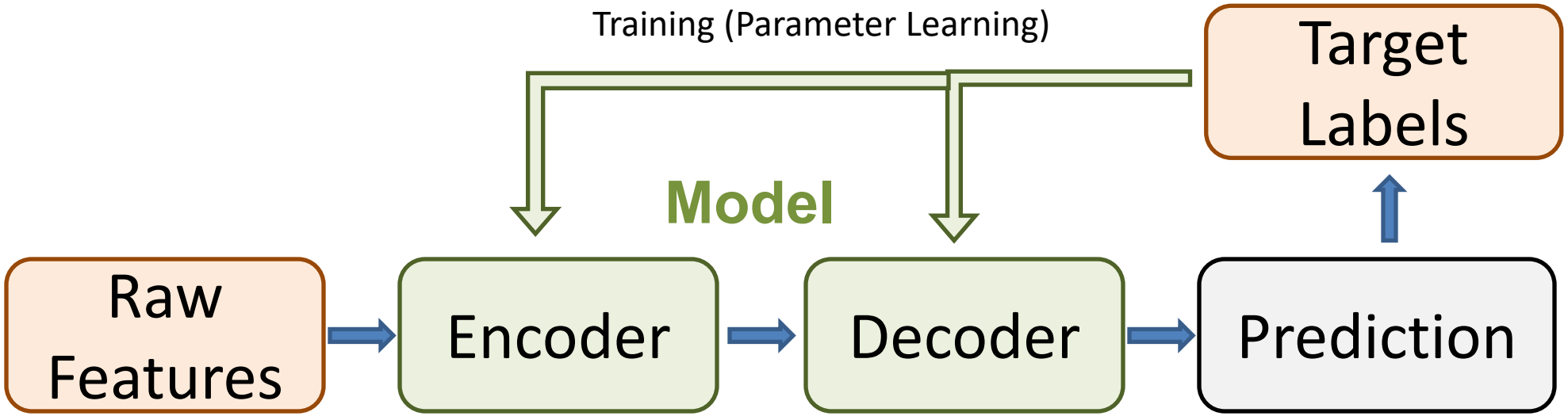
Applied Machine Learning
Derek Hoiem

Midterm Exam Logistics

- Reserve a time at CBTF during exam period
- Exam is 50 min
- Multiple choice or multiple select
- Closed book, except for this document
- You will not have time to look up all the answers, so do prepare by reviewing slides, lectures, AML book, and practice questions

Midterm Exam Central Topics

- How does **train/test error** depend on
  - Number of training samples
  - Complexity of model
- **Bias-variance trade-off**, including meaning of “bias” and “variance” for ML models and “overfitting”
- **Basic function/form/assumptions of classification/regression** models (KNN, NB, linear/logistic regression, SVMs)
- **Search/retrieval**, including brute force and LSH
- Data organization and transformation: **clustering, PCA**
- **Latent variables and robustness**: EM, density estimation, robust estimation and fitting



Discrete/continuous values

Text
Images
Audio
Structured/unstructured
Few/many features
Clean/noisy labels

Trees
Feature selection
Clustering
Kernels
Density estimation

Manual feature design
Deep networks

Linear regressor
Logistic regressor
Nearest Neighbor
Probabilistic model
SVM

Category
Continuous value
Clusters
Low dimensional embedding

Pixel labels
Generated text, image, audio
Positions

Learning a model

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

- $f(\boldsymbol{X};\theta)$: the model, e.g. $y = \boldsymbol{w}^T\boldsymbol{x}$
- θ: parameters of the model (e.g. $\boldsymbol{w}$)
- $(\boldsymbol{X}, \boldsymbol{y})$: pairs of training samples
- $Loss()$: defines what makes a good model
  - Good predictions, e.g. minimize $-\sum_n \log P(y_n|\boldsymbol{x}_n)$
  - Likely parameters, e.g. minimize $\boldsymbol{w}^T\boldsymbol{w}$
    - Regularization and priors indicate preference for particular solutions, which tends to improve generalization (for well chosen parameters) and can be necessary to obtain a unique solution

Prediction using a model

$$y_t = f(\boldsymbol{x_t}; \theta)$$

- Given some new set of input features $\boldsymbol{x}_t$, model predicts y_t
  - Regression: output y_t directly, possibly with some variance estimate
  - Classification
    - Output most likely y_t directly, as in nearest neighbor
    - Output $P(y_t|\boldsymbol{x}_t)$, as in logistic regression

Model evaluation process

1. Collect/define training, validation, and test sets
2. Decide on some candidate models and hyperparameters
3. For each candidate:
   a. Learn parameters with training set
   b. Evaluate trained model on the validation set
4. Select best model
5. Evaluate best model's performance on the test set
   - Cross-validation can be used as an alternative
   - Common measures include error or accuracy, root mean squared error, precision-recall

How to think about ML algorithms

- What is the model?
  - What kinds of functions can it represent?
  - What functions does it prefer? (regularization/prior)
- What is the objective function?
  - What “values” are implied?
  - The objective function does not always match the final evaluation metric
  - Objectives are designed to be optimizable and improve generalization
- How do I optimize the model?
  - How long does it take to train, and how does it depend on the amount of training data or number of features?
  - Can I reach a global optimum?
- How does the prediction work?
  - How accurate is the prediction?
  - How fast can I make a prediction for a new sample?
  - Does my algorithm provide a confidence on its prediction?

Bias-Variance Trade-off

$$\underbrace{E_{\mathbf{x},y,D}\left[(h_D(\mathbf{x}) - y)^2\right]}_{\text{Expected Test Error}} = \underbrace{E_{\mathbf{x},D}\left[(h_D(\mathbf{x}) - \bar{h}(\mathbf{x}))^2\right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y}\left[(\bar{y}(\mathbf{x}) - y)^2\right]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}}\left[(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x}))^2\right]}_{\text{Bias}^2}$$

Variance: due to limited data
Different training samples will give different models that vary in predictions for the same test sample

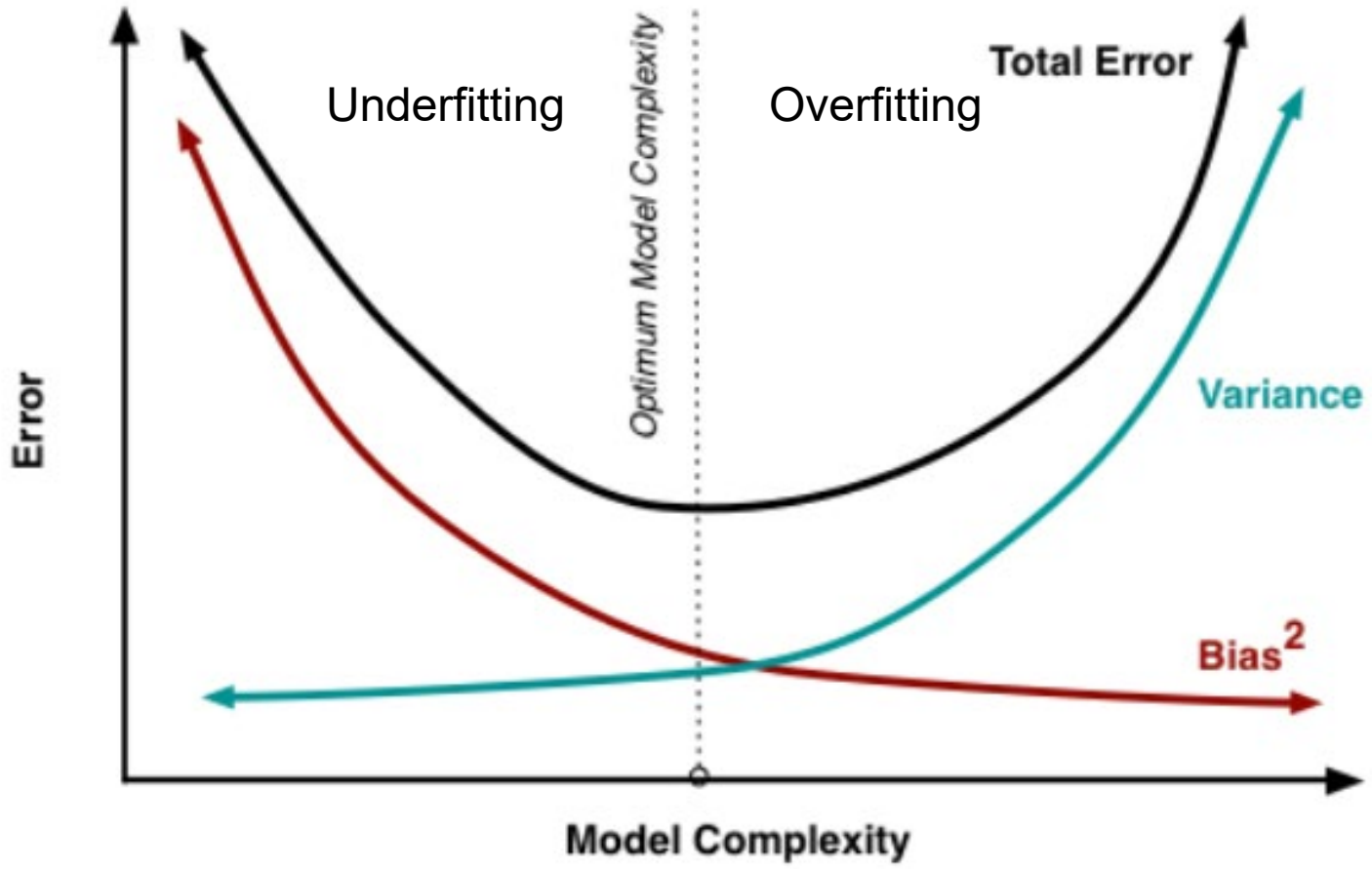
"**Noise**": irreducible error due to data/problem

Bias: error when optimal model is learned from infinite data

Above is for regression.
But same error = variance + noise + bias^2 holds for classification error and logistic regression.

See this for derivation

Fig Sources



How to detect high variance:

- Test error is much higher than training error

How to detect high bias or noise:

- The training error is high

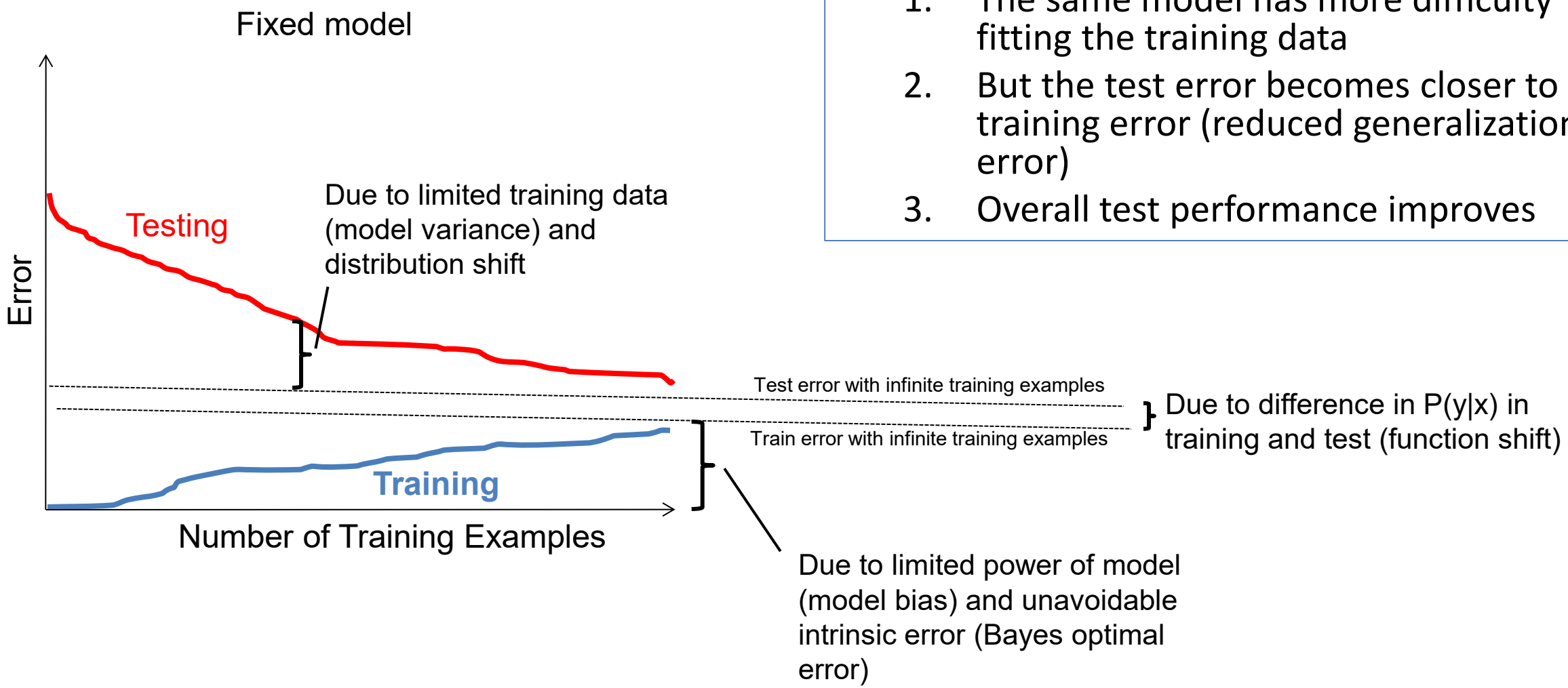
As you increase model complexity:

- Training error will decrease
- Test error may decrease (if you are currently “underfitting”) or increase (if you are “overfitting”)

What does “model complexity” mean?

- More parameters in the same structure, e.g. a deeper tree is more complex than a shallow tree
- Less regularization, e.g. smaller regularization penalty

Performance vs training size



As we get more training data:

1. The same model has more difficulty fitting the training data
2. But the test error becomes closer to training error (reduced generalization error)
3. Overall test performance improves

Classification methods

	Nearest Neighbor	Naïve Bayes	Logistic Regression
Type	Instance-Based	Probabilistic	Probabilistic
Decision Boundary	Partition by example distance	Usually linear	Usually linear
Model / Prediction	$i^* = \underset{i}{\operatorname{argmin}}\, dist(X_{trn}[i], x)$ $y^* = y_{trn}[i^*]$	$y^* = \underset{y}{\operatorname{argmax}} \prod_i P(x_i \mid y) P(y)$	$w^T x + b \approx \log \frac{P(y=1 \mid x)}{P(y=0 \mid x)}$ $y^* = \underset{y}{\operatorname{argmax}}\, P(y \mid x)$
Strengths	* Low bias * No training time * Widely applicable * Simple	* Estimate from limited data * Simple * Fast training/prediction	* Powerful in high dimensions * Widely applicable * Good confidence estimates * Fast prediction
Limitations	* Relies on good input features * Slow prediction (in basic implementation)	* Limited modeling power	* Relies on good input features

Classification methods (extended)

assuming x in {0 1}

	Learning Objective	Training	Inference
Naïve Bayes	$\text{maximize} \sum_i \left[\sum_j \log P(x_{ij} \mid y_i; \theta_j) + \log P(y_i; \theta_0) \right]$	$\theta_{kj} = \frac{\sum_i \delta(x_{ij} = 1 \wedge y_i = k) + r}{\sum_i \delta(y_i = k) + Kr}$	$\theta_1^T \mathbf{x} + \theta_0^T (1 - \mathbf{x}) > 0$ where $\theta_{1j} = \log \frac{P(x_j = 1 \mid y = 1)}{P(x_j = 1 \mid y = 0)}$, $\theta_{0j} = \log \frac{P(x_j = 0 \mid y = 1)}{P(x_j = 0 \mid y = 0)}$
Logistic Regression	$\text{minimize} \sum_i -\log(P(y_i \mid \mathbf{x}, \boldsymbol{\theta})) + \lambda \lVert \boldsymbol{\theta} \rVert$ where $P(y_i \mid \mathbf{x}, \boldsymbol{\theta}) = 1/(1 + \exp(-y_i \boldsymbol{\theta}^T \mathbf{x}))$	Gradient descent	$\boldsymbol{\theta}^T \mathbf{x} > t$
Linear SVM	$\text{minimize } \lambda \sum_i \xi_i + \frac{1}{2} \lVert \boldsymbol{\theta} \rVert$ such that $y_i \boldsymbol{\theta}^T \mathbf{x} \geq 1 - \xi_i \ \ \forall i, \ \xi_i \geq 0$	Quadratic programming or subgradient opt.	$\boldsymbol{\theta}^T \mathbf{x} > t$
Kernelized SVM	complicated to write	Quadratic programming	$\sum_i y_i \alpha_i K(\hat{\mathbf{x}}_i, \mathbf{x}) > 0$
Nearest Neighbor	most similar features → same label	Record data	y_i where $i = \underset{i}{\text{argmin}} \ K(\hat{\mathbf{x}}_i, \mathbf{x})$

* Notation may differ from previous slide

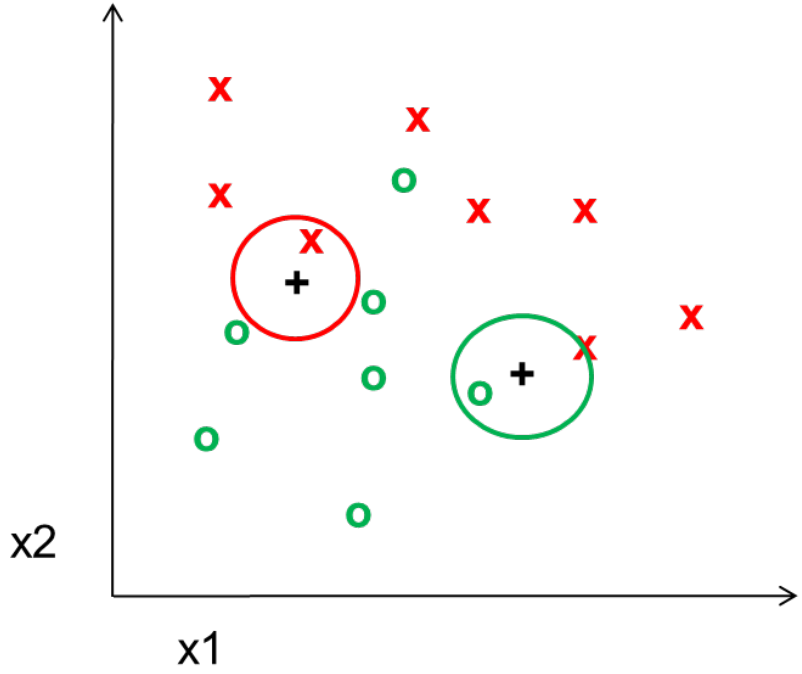
Regression methods

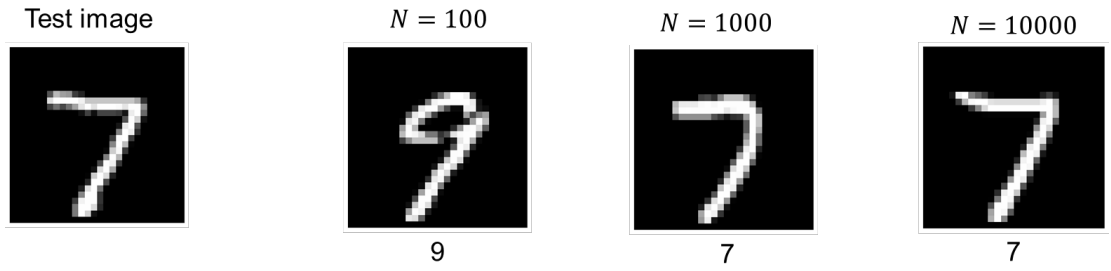
	Nearest Neighbor	Naïve Bayes	Linear Regression
Type	Instance-Based	Probabilistic	Data fit
Decision Boundary	Partition by example distance	Usually linear	Linear
Model / Prediction	$i^* = \underset{i}{\operatorname{argmin}}\, dist(X_{trn}[i], x)$ $y^* = y_{trn}[i^*]$	$y^* = \underset{y}{\operatorname{argmax}} \prod_i P(x_i \mid y) P(y)$	$y^* = w^T x$
Strengths	* Low bias * No training time * Widely applicable * Simple	* Estimate from limited data * Simple * Fast training/prediction	* Powerful in high dimensions * Widely applicable * Fast prediction * Coefficients may be interpretable
Limitations	* Relies on good input features * Slow prediction (in basic implementation)	* Limited modeling power	* Relies on good input features

Summaries

KNN Classification (L2)

- Foundation of ML: similar features predict similar labels
  - Hard part: How to represent inputs with vectors that reflect the similarity
- KNN is a simple but effective classifier that predicts the label of the most similar training example(s)
  - Accuracy depends on quality of features and number of training samples
- Larger K gives a smoother prediction function
- Measure classification performance with error and confusion matrices





Working with Data (L2)

- **Data** is a set of numbers that contains information. Images, audio, signals, tabular data and everything else must be represented as a vector of numbers to be used in ML.

- **Information** is the power to predict something – a lot of the challenge in ML is in transforming the data to make the desired information more obvious

- In machine learning, we have
  Sample: a data point, such as a feature vector and label corresponding to the input and desired output of the model
  Dataset: a collection of samples
  Training set: a dataset used to train the model
  Validation set: a dataset used to select which model to use or compare variants and manually set parameters
  Test set: a dataset used to evaluate the final model

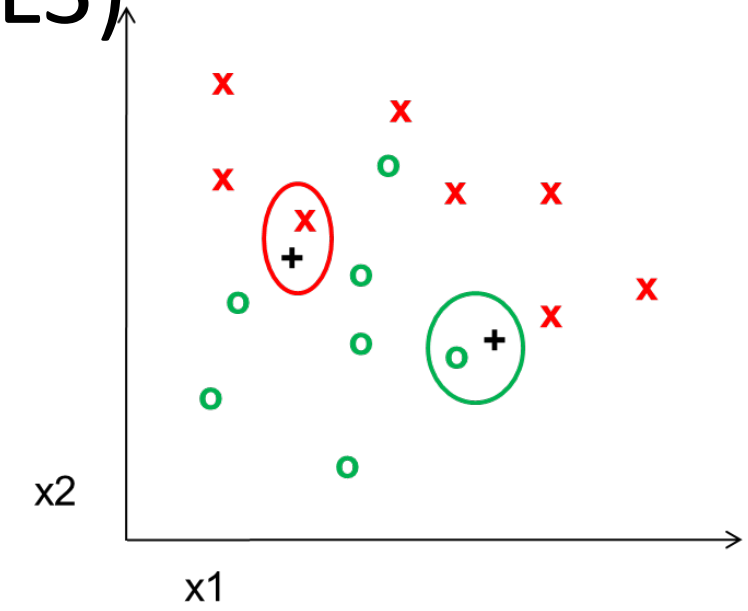
- In a **classification** problem, the goal is to map from features to a categorical label (or “class”)

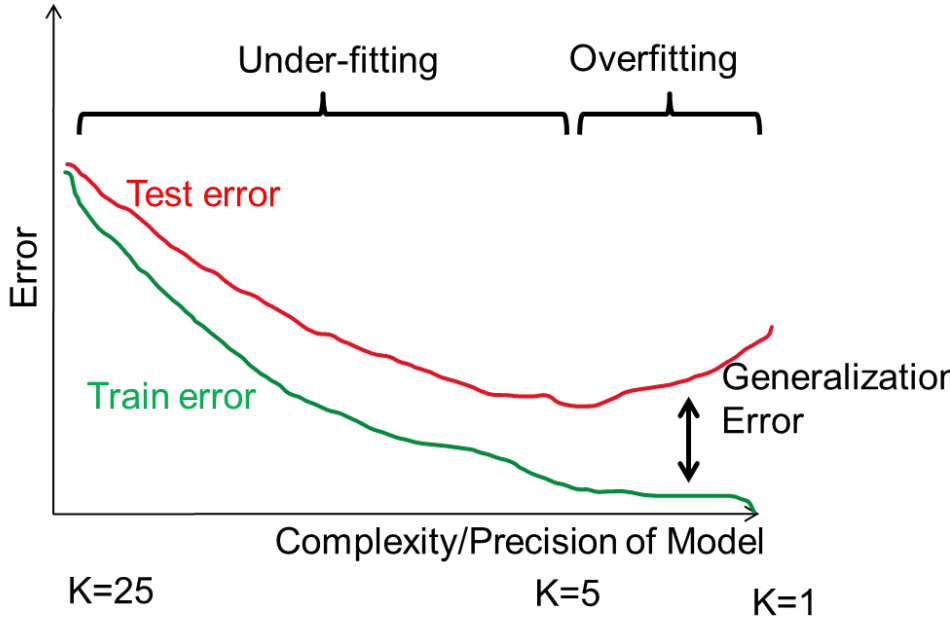
- Nearest neighbor (or **K-NN**) algorithm can perform classification by retrieving the K nearest neighbors to the query features and assigning their most common label

- We can measure **error** and **confusion matrices** to show the fraction of mistakes and what kinds of mistakes are made

KNN Regression & Generalization (L3)

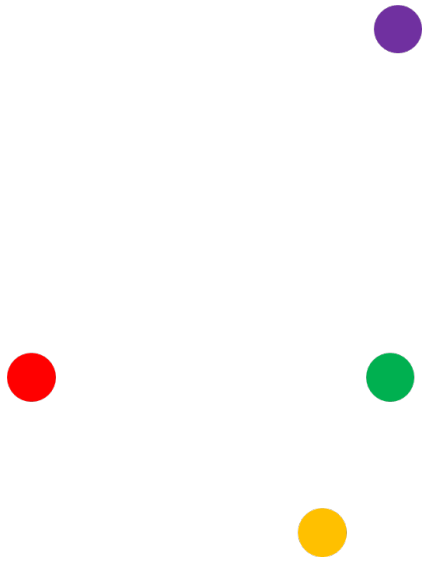
- Similarity/distance measures: L1, L2, cosine
- KNN can be used for either classification (return most common label) or regression (return average target value)
- Regression error measures
  - Root mean squared error(RMSE)
    $\sqrt{\frac{1}{N}\sum_i (f(X_i) - y_i)^2}$
  - R^2: $1 - \frac{\sum_i (f(X_i) - y_i)^2}{\sum_i (y_i - \bar{y})^2}$
- Test error is composed of
  - **Irreducible error** (perfect prediction not possible given features)
  - **Bias** (model cannot perfectly fit the true function)
  - **Variance** (parameters cannot be perfectly learned from training data)

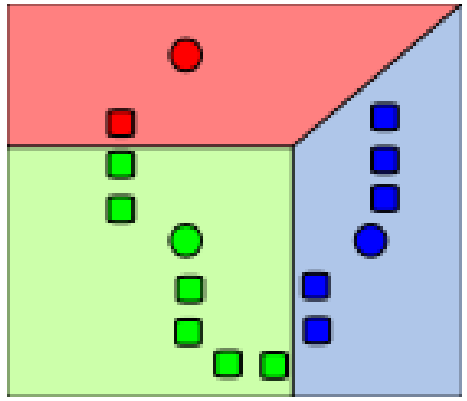




Clustering (L4)

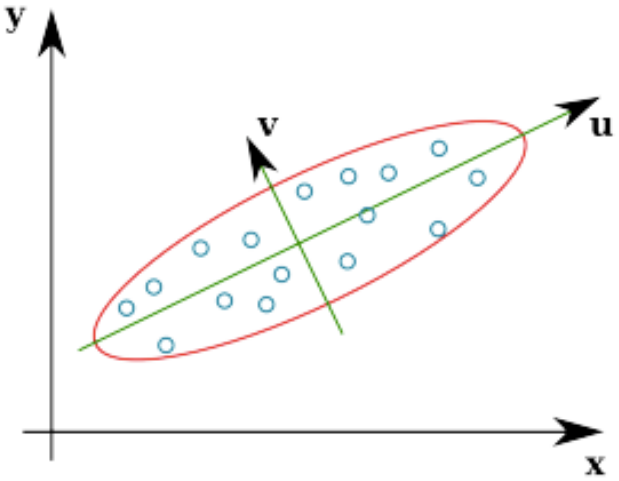
- Use highly optimized libraries like FAISS for search/retrieval
- Approximate search methods like LSH can be used to find similar points quickly
- Clustering groups similar data points
- K-means is the must-know method, but there are many others



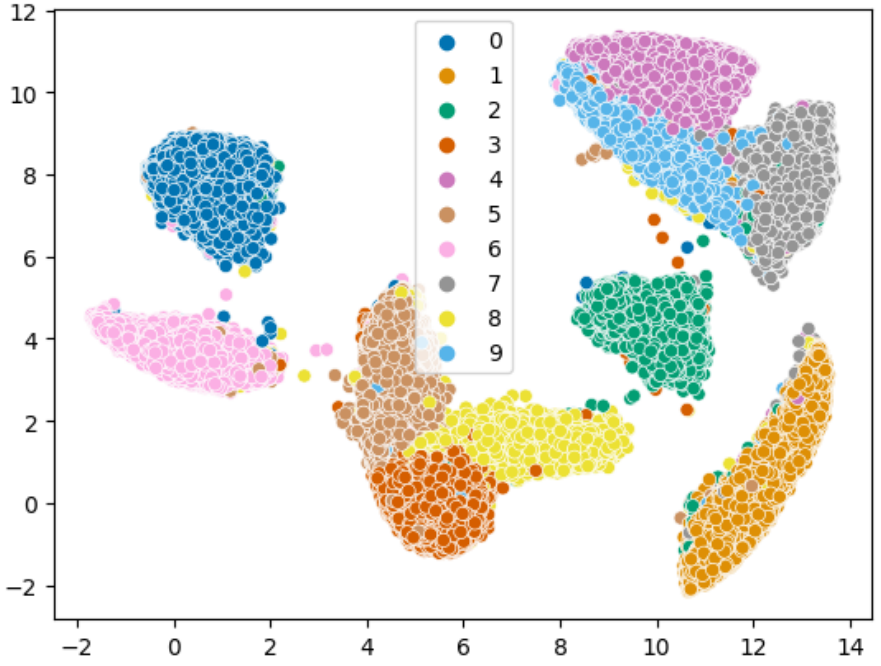


PCA/Embedding (L5)

- PCA reduces dimensions by linear projection
  - Preserves variance to reproduce data as well as possible, according to mean squared error
  - May not preserve local connectivity structure or discriminative information



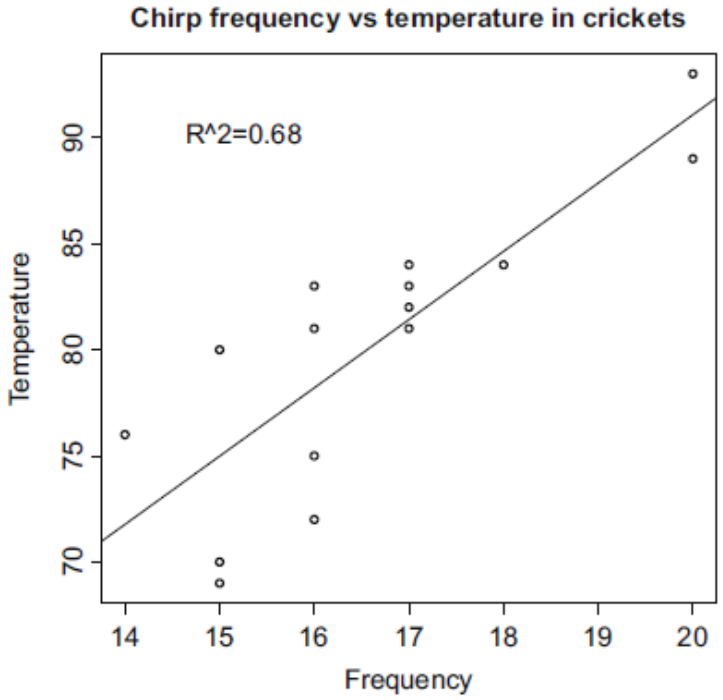
- Other methods try to preserve relationships between points
  - MDS: preserve pairwise distances
  - IsoMap: MDS but using a graph-based distance
  - t-SNE: preserve a probabilistic distribution of neighbors for each point (also focusing on closest points)
  - UMAP: incorporates k-nn structure, spectral embedding, and more to achieve good embeddings relatively quickly

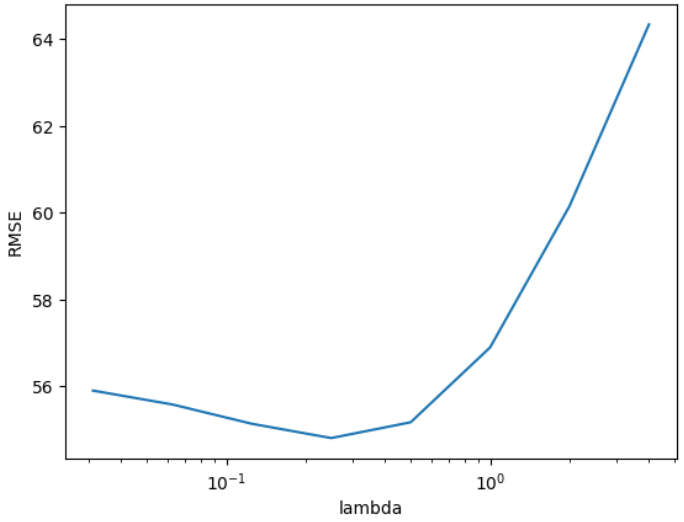


Linear Regression (L6)

- Linear regression fits a linear model to a set of feature points to predict a continuous value
  - Explain relationships
  - Predict values
  - Extrapolate observations

- Regularization prevents overfitting by restricting the magnitude of feature weights
  - L1: prefers to assign a lot of weight to the most useful features
  - L2: prefers to assign smaller weight to everything





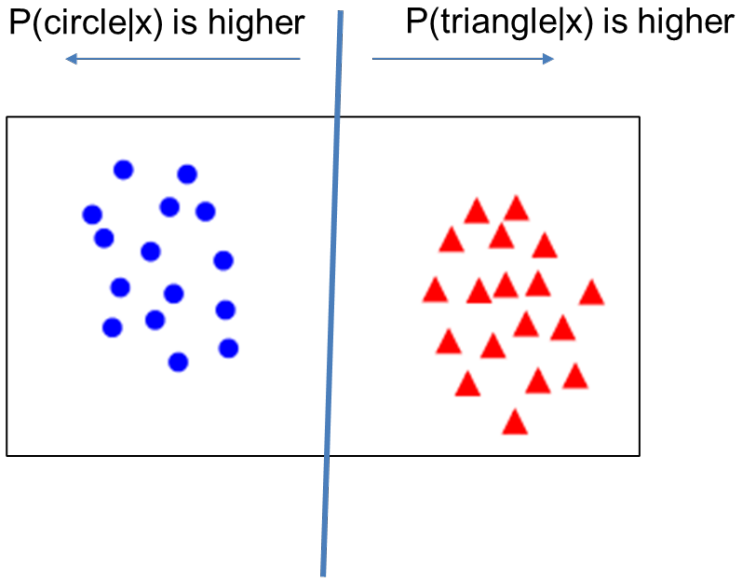
Linear Classifiers (L7)

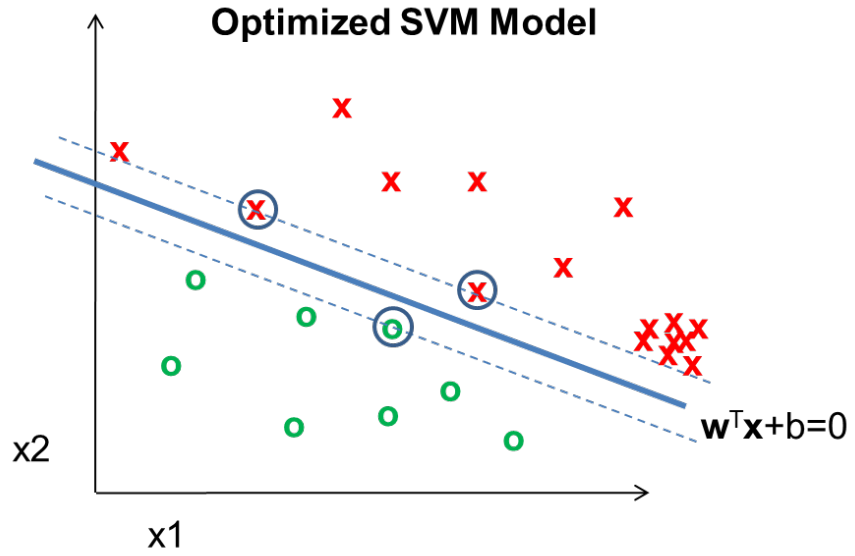
- Linear logistic regression and linear SVM are classification techniques that aims to split features between two classes with a linear model
  - Predict categorical values with confidence

- Logistic regression maximizes confidence in the correct label, while SVM just tries to be confident enough

- Non-linear versions of SVMs can also work well and were once popular (but almost entirely replaced by deep networks)

- Nearest neighbor and linear models are the final predictors of most ML algorithms – the complexity lies in finding features that work well with NN or linear models





Probability / Naïve Bayes (L8)

- Probabilistic models are a large class of machine learning methods

- Naïve Bayes assumes that features are independent given the label
  - Easy/fast to estimate parameters
  - Less risk of overfitting when data is limited

$$P(\boldsymbol{x}, y) = \prod_i P(x_i|y)P(y)$$

- You can look up how to estimate parameters for most common probability models
  - Or take partial derivative of total data/label likelihood given parameter

- Prediction involves finding y that maximizes $P(x, y)$, either by trying all y or solving partial derivative

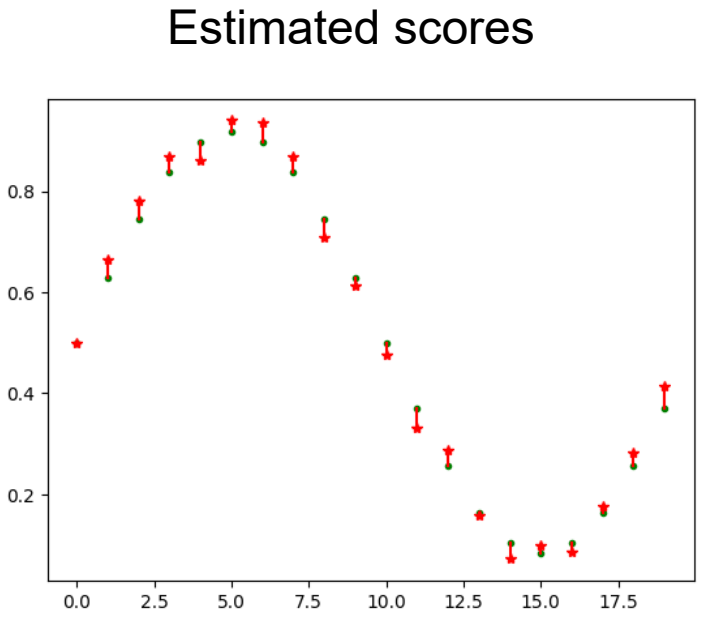
- Maximizing $\log P(x, y)$ is equivalent to maximizing $P(x, y)$ and often much easier

$$y^* = \operatorname{argmax}_y \prod_i P(x_i|y)\,P(y)$$

$$= \operatorname{argmax}_y \sum_i \log P(x_i|y) + \log P(y)$$

EM (L9)

- EM is a widely applicable algorithm to solve for latent variables and parameters that make the observed data likely
  - E-step: compute the likelihoods of the values of the latent variables
  - M-step: solve for most likely model parameters, using the likelihoods from the E-step as weights
- While derivation is long and somewhat complicated, the application is simple
- EM is used, for example, in mixture of Gaussian and topic models



Good annotators: 0, 1, 3

PDF Estimation (L10)

	Parametric Models	**Semi-Parametric**	**Non-Parametric**
Description	Assumes a fixed form for density	Can fit a broad range of functions with limited parameters	Can fit any distribution
Examples	Gaussian, exponential	Mixture of Gaussians	Discretization, kernel density estimation
Good when	Model is able to approximately fit the distribution	Low dimensional or smooth distribution	1-D data
Not good when	Model cannot approximate the distribution	Distribution is not smooth, challenging in high dimensions	Data is high dimensional

Robust Estimation (L11)

Median and quantiles are robust to outliers, while mean/min/max aren't

Outliers can be detected as low probability points, low density points, poorly compressible points, or through 2D visualizations

Least squares is not robust to outliers. Use RANSAC or IRLS or robust loss function instead.