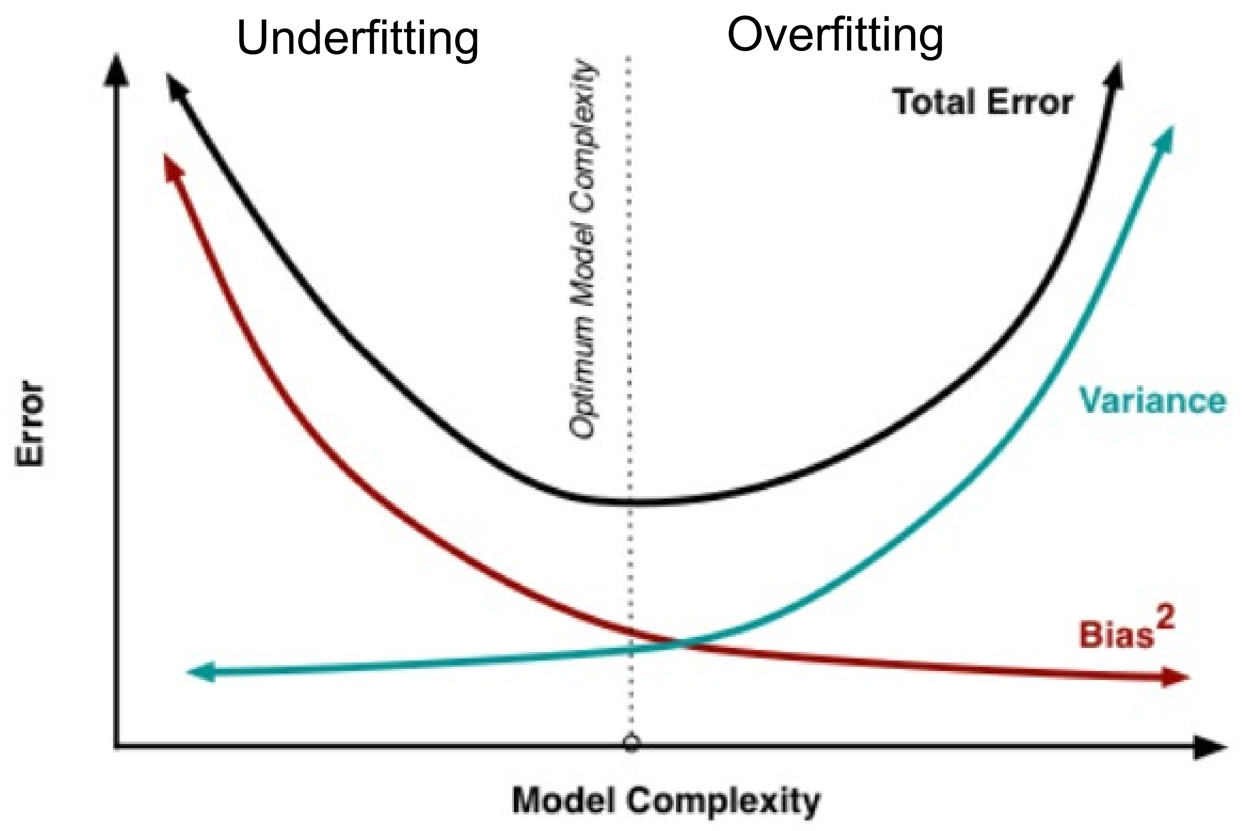
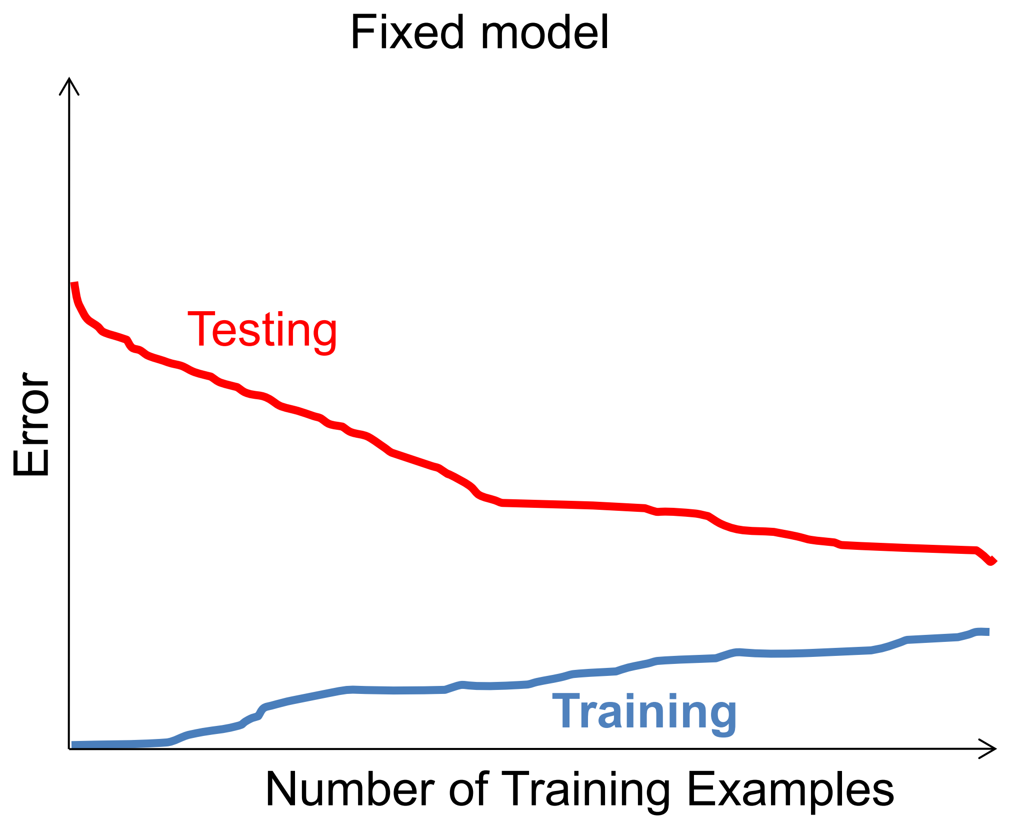
Exam 1 Reference

Applied Machine Learning
Derek Hoiem

Bias-Variance / Generalization





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	$\text{maximize} \sum_i \left[\sum_j \log P(x_{ij} \mid y_i; \theta_j) + \log P(y_i; \theta_0) \right]$	$\theta_{kj} = \dfrac{\sum_i \delta(x_{ij} = 1 \wedge y_i = k) + r}{\sum_i \delta(y_i = k) + Kr}$	$\boldsymbol{\theta}_1{}^T \mathbf{x} + \boldsymbol{\theta}_0{}^T (1 - \mathbf{x}) > 0$ where $\theta_{1j} = \log \dfrac{P(x_j = 1 \mid y = 1)}{P(x_j = 1 \mid y = 0)}$, $\theta_{0j} = \log \dfrac{P(x_j = 0 \mid y = 1)}{P(x_j = 0 \mid y = 0)}$
Logistic Regression	$\textit{minimize} \sum_i -\log(P(y_i \mid \mathbf{x}, \boldsymbol{\theta})) + \lambda \lVert \boldsymbol{\theta} \rVert$ 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 $y_i \boldsymbol{\theta}^T \mathbf{x} \ge 1 - \xi_i \;\; \forall i, \; \xi_i \ge 0$	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	$i^* = \underset{i}{\operatorname{argmin}}\, dist(X_{trn}[i], x)$ $y^* = y_{trn}[i^*]$	$y^* = \underset{y}{\operatorname{argmax}} \prod_i P(x_i \vert y) P(y)$	$y^* = w^T x$
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

KNN Classification & Datasets (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

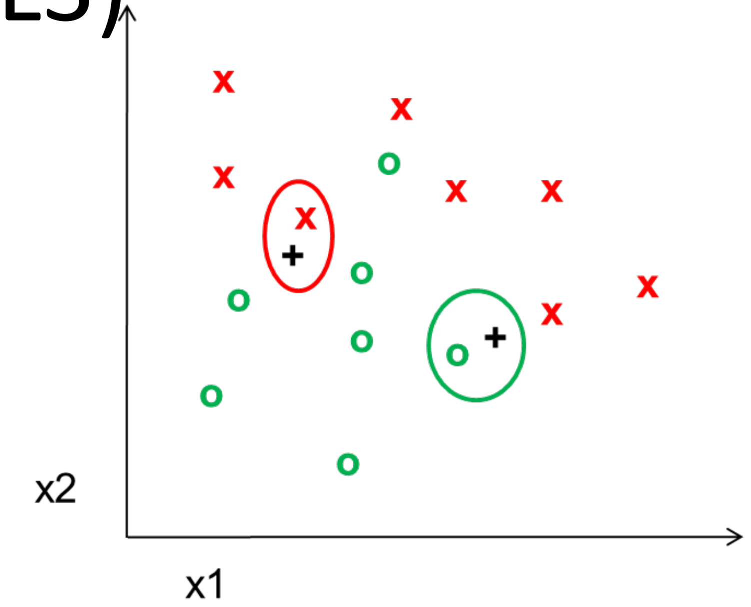
KNN Regression & Generalization (L3)

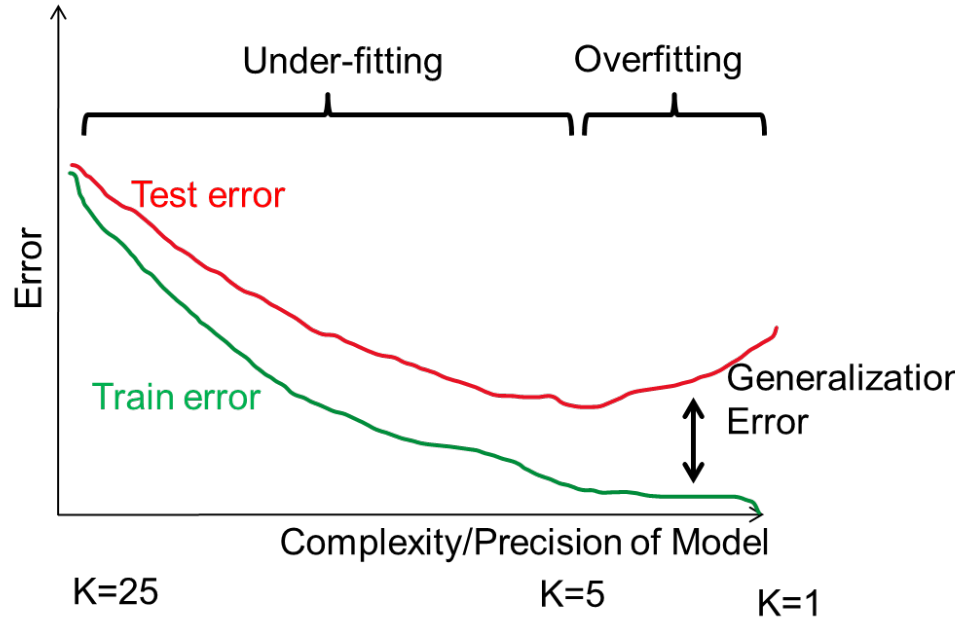
- Similarity/distance measures: L1, L2, cosine

- KNN can be used for either classification (return most common label) or regression (return average target value)

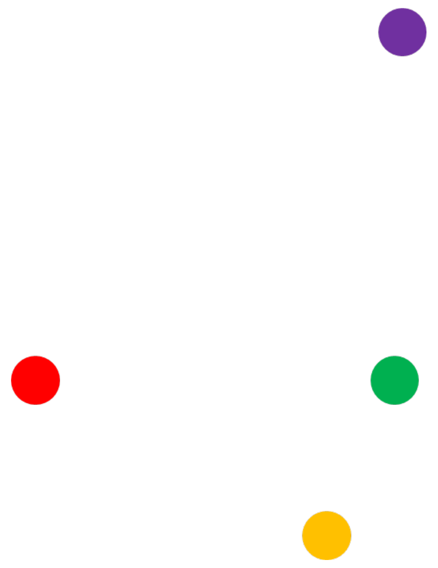
- Regression error measures
  - Root mean squared error(RMSE)
    $\sqrt{\frac{1}{N}\sum_i (f(X_i) - y_i)^2}$
  - R^2: $1 - \frac{\sum_i (f(X_i)-y_i)^2}{\sum_i (y_i-\bar{y})^2}$

- 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)

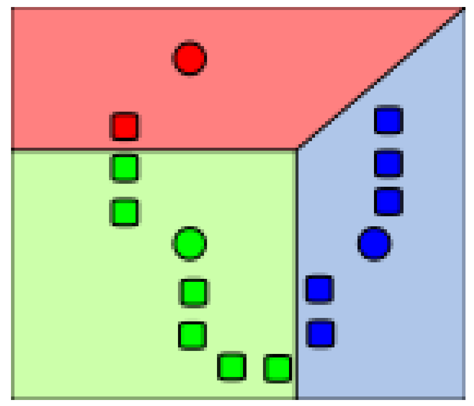




Clustering (L4)



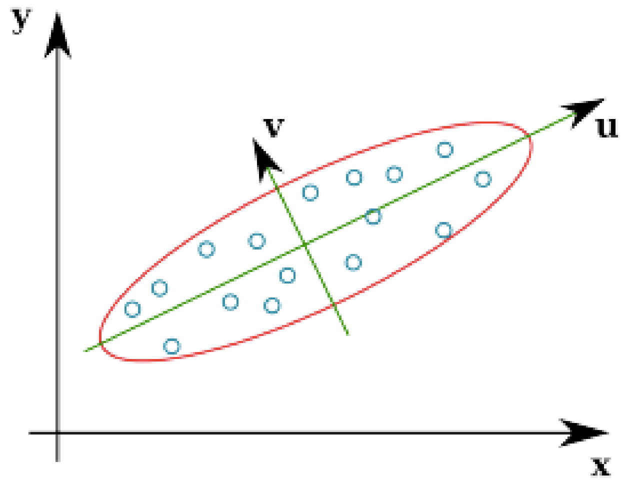
- Use highly optimized libraries like FAISS for search/retrieval
- Approximate search methods like LSH can be used to find similar points quickly
- Clustering groups similar data points
- K-means is the must-know method, but there are many others

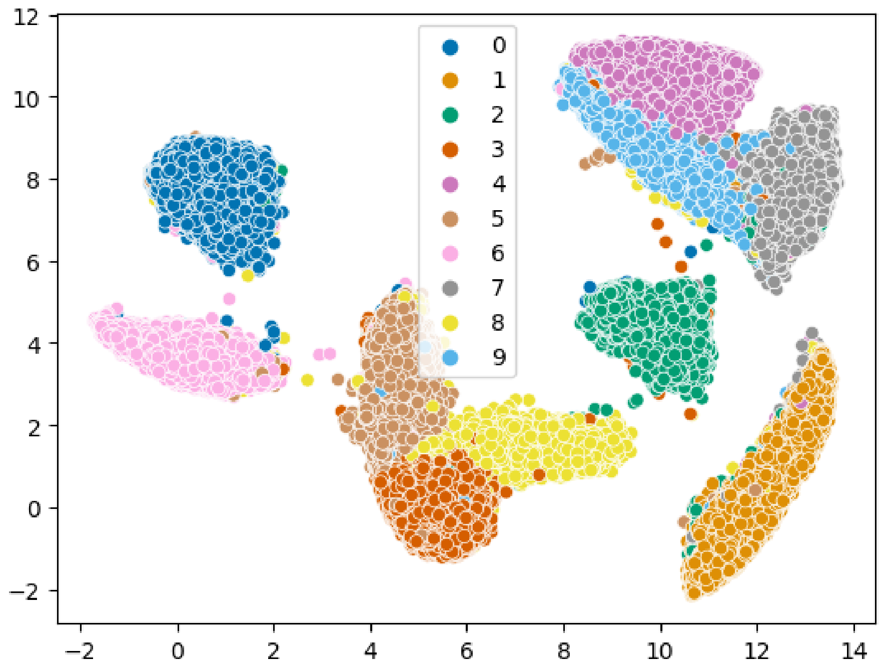


PCA/Embedding (L5)

- 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

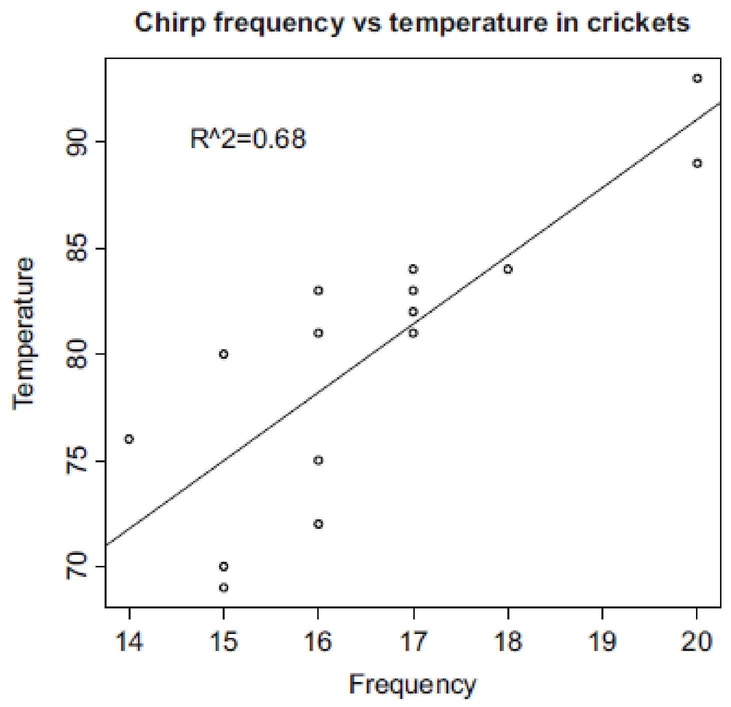


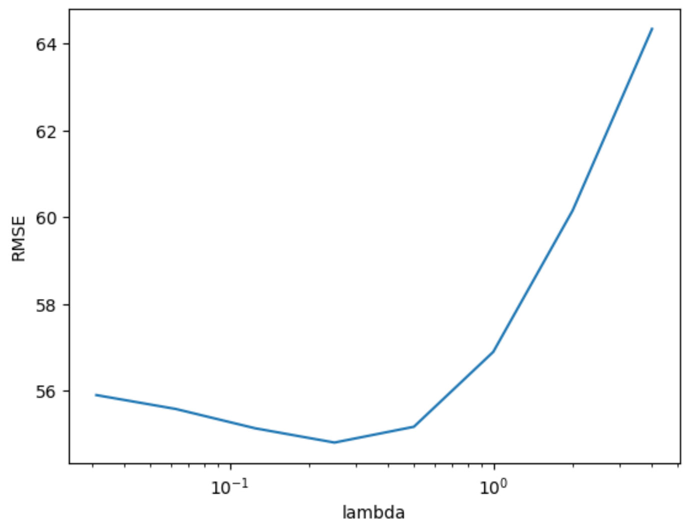


Linear Regression (L6)

- 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





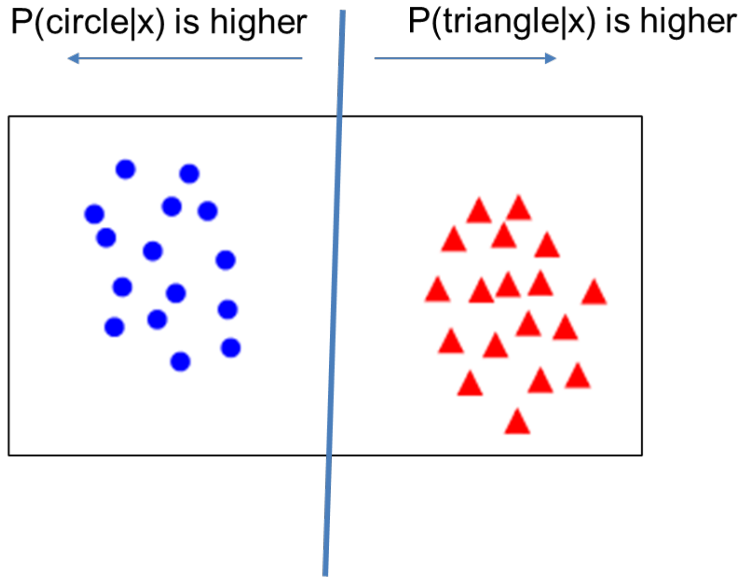
Linear Classifiers (L7)

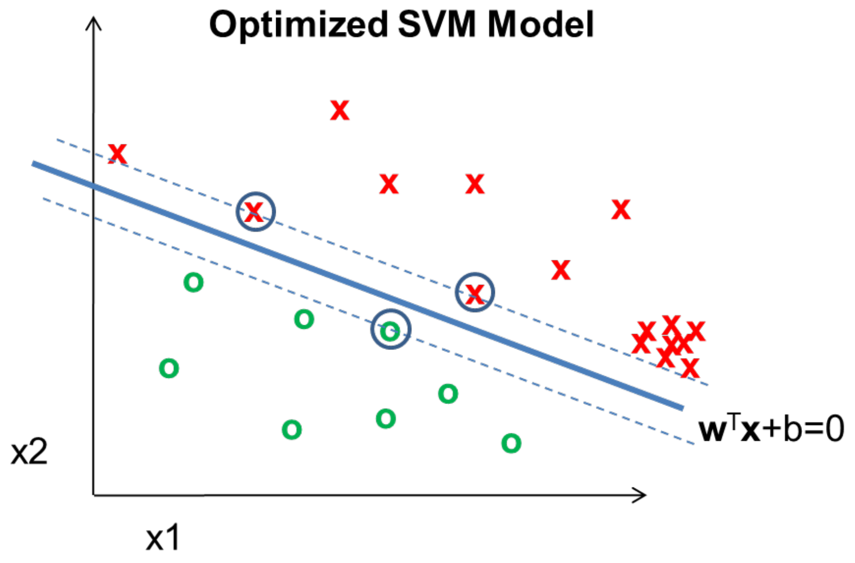
- Linear logistic regression and linear SVM are classification techniques that aims to split features between two classes with a linear model
  - Predict categorical values with confidence

- 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
- 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

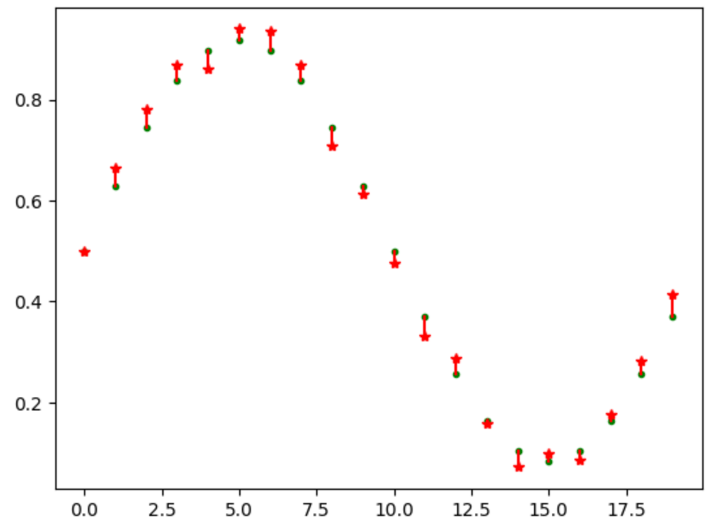
$$P(\boldsymbol{x}, y) = \prod_i P(x_i|y)P(y)$$

$$y^* = \arg\max_y \prod_i P(x_i|y)\, P(y)$$
$$= \arg\max_y \sum_i \log P(x_i|y) + \log P(y)$$

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

Estimated scores



(Green = true; red = prediction)

Good annotators: 0, 1, 3

PDF Estimation (L10)

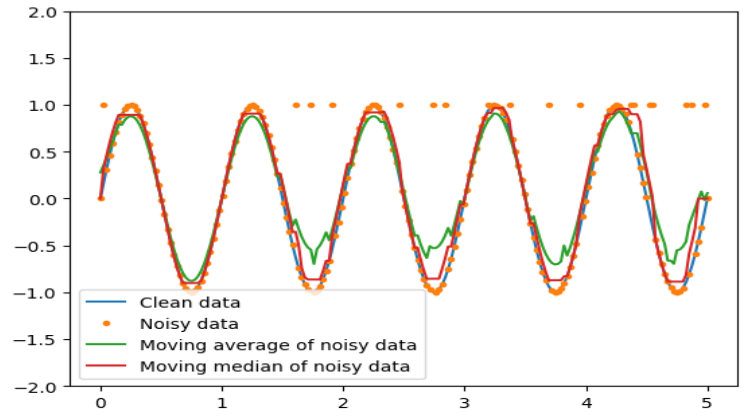
	Parametric Models	**Semi-Parametric**	**Non-Parametric**
Description	Assumes a fixed form for density	Can fit a broad range of functions with limited parameters	Can fit any distribution
Examples	Gaussian, exponential	Mixture of Gaussians	Discretization, kernel density estimation
Good when	Model is able 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

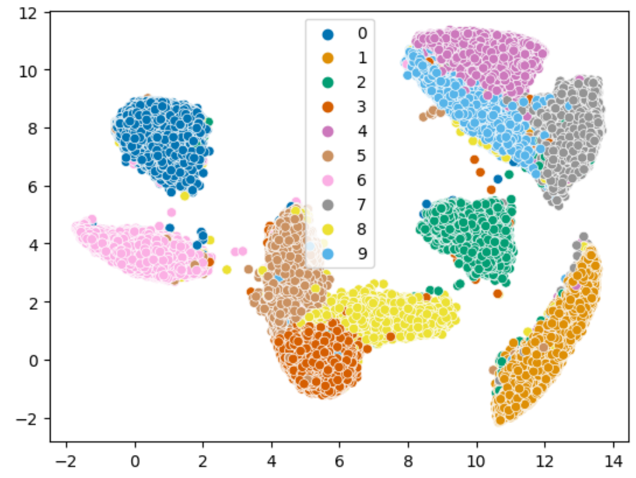
Robust Estimation (L11)

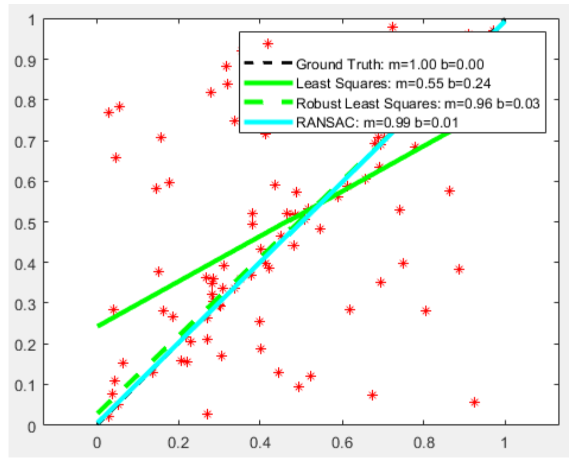
Median and quantiles are robust to outliers, while mean/min/max aren't

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.


Clean data
Noisy data
Moving average of noisy data
Moving median of noisy data


0
1
2
3
4
5
6
7
8
9


Ground Truth: m=1.00 b=0.00
Least Squares: m=0.55 b=0.24
Robust Least Squares: m=0.96 b=0.03
RANSAC: m=0.99 b=0.01