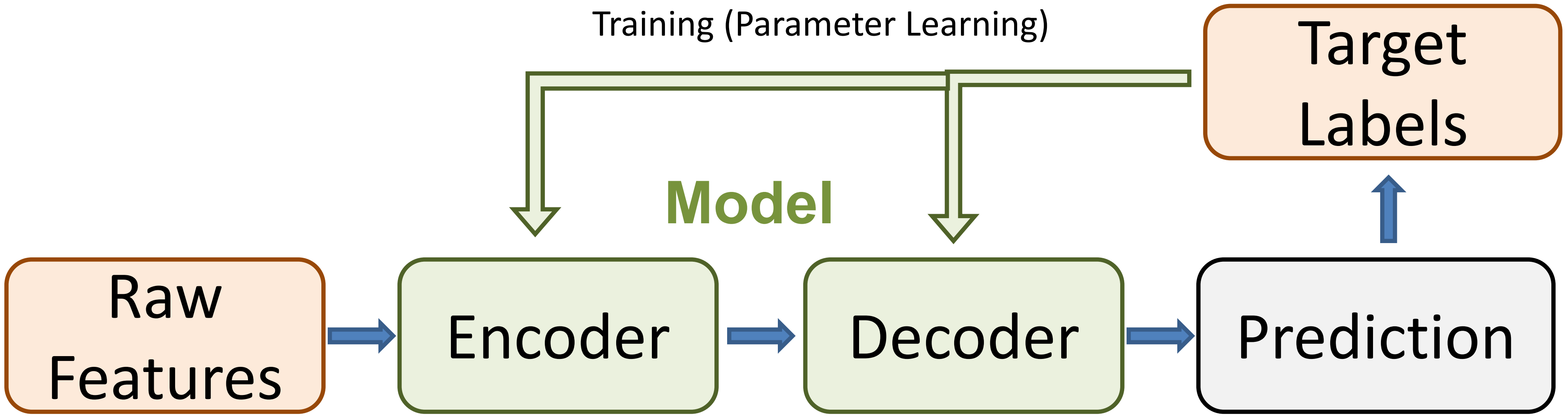
Consolidation and Review

Applied Machine Learning
Derek Hoiem

Dall-E: A cauldron full of books and math equations and plots in a fire, cartoon style



Discrete/continuous values
Images

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

Trees
Feature selection

Manual feature design
Deep networks
Clustering
Kernels
Density estimation

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
- $(\boldsymbol{X}, \boldsymbol{y})$: 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
  - Classification
    - Output most likely y_t directly, as in nearest neighbor or Naïve Bayes
    - 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 parameters
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?
  - Note that the objective function often does not 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 fast can I make a prediction for a new sample?
  - Can I find the most likely prediction according to my model?
  - Does my algorithm provide a confidence on its prediction?

Classification methods

	Nearest Neighbor	Naïve Bayes	Logistic Regression	Decision Tree
Type	Instance-Based	Probabilistic	Probabilistic	Probabilistic
Decision Boundary	Partition by example distance	Usually linear	Usually linear	Partition by selected boundaries
Model / Prediction	$i^* = \underset{i}{\operatorname{argmin}}\, dist(X_{trn}[i], x)$ $y^* = y_{trn}[i^*]$	$y^* = \underset{y}{\operatorname{argmax}} \prod_i P(x_i\|y)P(y)$	$w^Tx + b \approx \log \frac{P(y=1\|x)}{P(y=0\|x)})$ $y^* = \underset{y}{\operatorname{argmax}}\, P(y\|x)$	Conjunctive rules $y^* = leaf(x)$
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	* Explainable decision function * Widely applicable * Does not require feature scaling
Limitations	* Relies on good input features * Slow prediction (in basic implementation)	* Limited modeling power	* Relies on good input features	* One tree tends to either generalize poorly or underfit the data

Classification methods (extended)

assuming x in {0 1}

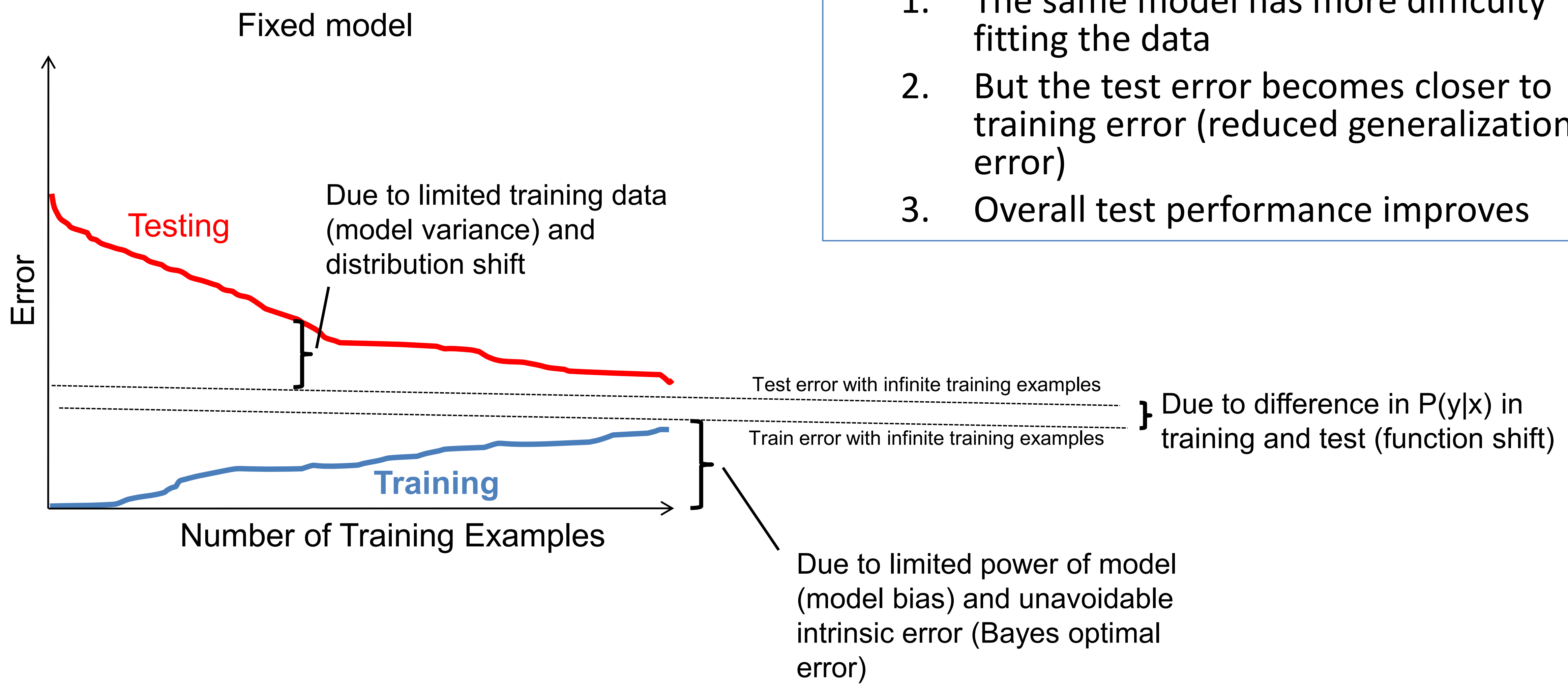
	Learning Objective	Training	Inference
Naïve Bayes	$\text{maximize} \sum_i \left[\sum_j \log P(x_{ij} \mid y_i; \theta_j) + \log P(y_i; \theta_0) \right]$	$\theta_{kj} = \frac{\sum_i \delta(x_{ij} = 1 \wedge y_i = k) + r}{\sum_i \delta(y_i = k) + Kr}$	$\boldsymbol{\theta}_1^T \mathbf{x} + \boldsymbol{\theta}_0^T (1 - \mathbf{x}) > 0$ 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	Decision Tree
Type	Instance-Based	Probabilistic	Data fit	Probabilistic
Decision Boundary	Partition by example distance	Usually linear	Linear	Partition by selected boundaries
Model / Prediction	$i^* = \operatorname{argmin}_i dist(X_{trn}[i], x)$ $y^* = y_{trn}[i^*]$	$y^* = \operatorname{argmax}_y \prod_i P(x_i\|y)P(y)$	$y^* = w^T x$	Conjunctive rules $y^* = leaf(x)$
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	* Explainable decision function * Widely applicable * Does not require feature scaling
Limitations	* Relies on good input features * Slow prediction (in basic implementation)	* Limited modeling power	* Relies on good input features	* One tree tends to either generalize poorly or underfit the data

Performance vs training size



As we get more training data:

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

Example: Breast Cancer Classification

- Motivation
  - Breast cancer diagnosis from fine needle aspirates (FNA) is reported to be 94%, but results are suspected to be biased
  - Need computer-based tests that are less subjective so that FNA is a more effective diagnostic tool for breast cancer
- Collected data from 569 patients, plus 54 for held-out testing
- A user interface was created to outline borders of suspect cells, and automated measurement of ten characteristics (e.g. radius, area, compactness, …) was performed and mean of all cells, mean of 3 largest, and std were recorded for each patient

[paper (Wolberg et al. 1995)]

Let’s explore in Python

https://colab.research.google.com/drive/1viVU62gk77THZBFuztWjxgL93xMMpiU0?usp=sharing

Method/Results from Breast Cancer Analysis Paper

- A MSM-Tree was used for classification
  - Fits a linear classifier based on a few features for each split
  - Aimed to minimize the number of splitting planes and number of features used (for simplicity and to improve generalization)
  - Final approach was splitting plane based on mean texture, worst area, and worst smoothness

- 10-fold cross validatdion
  - Achieved 3% error (+- 1.5% for 95% confidence interval)
- Perfect accuracy in held out test set

Next week

- Ensembles: model averaging and forests
- SVM and stochastic gradient descent