0.1 Supervised Learning

0.2 Weighted Least Squares

0.3 Regression

0.4 Perception

0.5 Naive Bayes

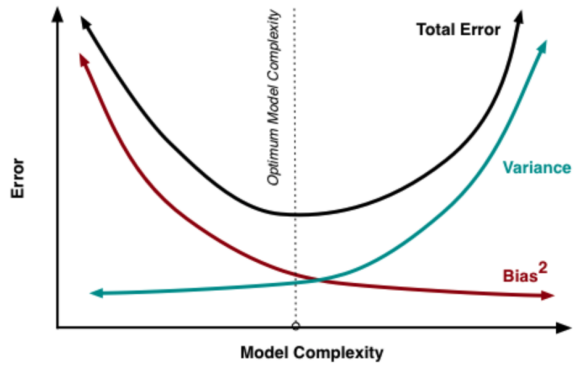
0.6 Neural Networks

0.7 Backpropagation

0.8 Bias-variance

- Underfitting = high bias, the error is pretty big
- Underfit: Train loss $\approx$ test loss but error is high
- Overfitting = high variance, fits the data well, but changes on each sample
- Overfit: train loss $<$ test loss, but error maybe low

Figure 1: Bias-variance tradeoff diagram



-
- Bias$(\hat{\theta}) = E[\hat{\theta}] - \theta$
- Variance$(\hat{\theta}) = E[(E[\hat{\theta}] - \hat{\theta}])^2$
- Squared loss $= (y - \hat{y})^2$ where y is the true value and $\hat{y}$ is the predicted value
- Bias-variance decomposition
  - We can decompose a loss function such as the squared loss into three terms: a variance, bias, and noise term

$$E[(y - \hat{y})^2] = (y - E[\hat{y}])^2 + E[E[(\hat{y}] - \hat{y})^2]$$
$$= [Bias]^2 + Variance$$

- Decompose the 0-1 loss that we commonly use for clsasification accuracy or error
- Blog post about decomposition

0.9 Regularization

- Reduce variance to obtain more robust model

$$\underset{\theta \in \mathbb{R}^d}{argmin} \frac{1}{2} \sum_{i=1}^{n} (x^{(i)}\theta - y^{(i)})^2 + \frac{\lambda}{2} \|\theta\|_2^2$$

λ = hyperparamater
$\|\theta\|_2^2$ = penalty for model complexity
$\lambda = 0$ = ordinary least squares

- Set λ to balance bias-variance tradeoff

0.10 Expectation Maximization

0.11 PCA

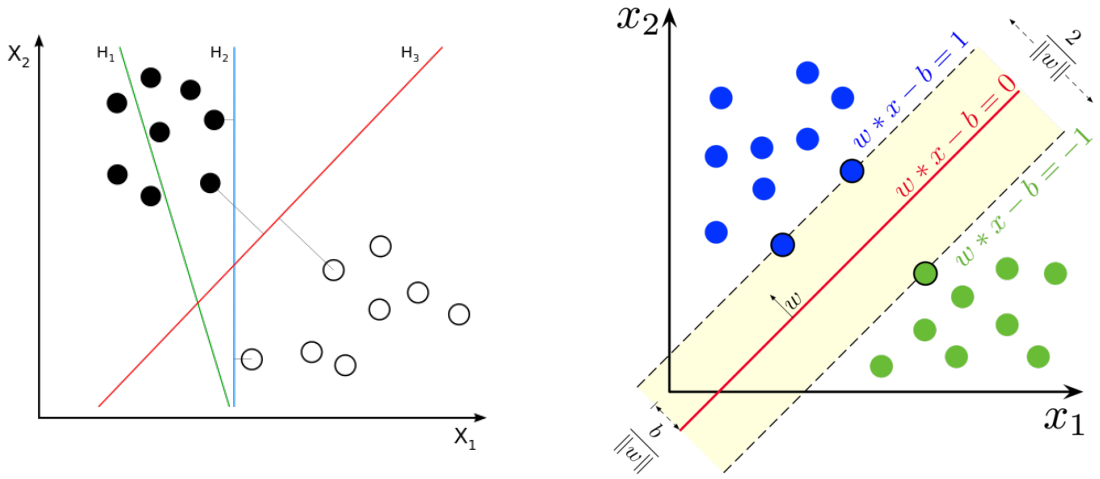
0.12 Support Vector Machines

- svm-tutorial
- Similar to logistic regression in that it is driven by a linear function $w^T x + b$
- SVMs does not provide probabilities, but outputs a class identity
- Given some data points, figure out the class of a new data point
- Find the best hyperplane that represents the largest separation, or margin, between two classes
- Larger the margin, the lower the generalization error
- SVM predicts positive class then $w^T x + b$ is positive, else negative
- Kernel trick: ML algorithms can be written in terms of dot products between examples, mapping inputs to high-dimensional feature spaces

$$w^T x + b = b + \sum_{i=1}^{m} \alpha_i x^T x^{(i)}$$

- Replace x with the output of feature function $\phi(x)$ and dot product with the kernel, $k(x, x^{(i)}) = \phi(x) \cdot \phi(x^{(i)})$

Figure 2: SVM



- Apply $\phi(x)$ to all inputs and then learning a linear model in a new transformed space
- Gaussial kernel: $k(u, v) = \mathcal{N}(u - v; 0, \sigma^2 I)$, also known as the radial basis kernel
- Training examples that have nonzero α_i are known as support vectors
- Can perform both linear and nonlinear classification using the kernel trick
- Hyperplane can be written in the form of a line, $y = ax+b$ or equivalently $w^T x-b = 0$ where w is the normal vector to the hyperplane
- We want to find two parallel hyperplanes, "margins", that separate the two class of data so that the distance between them is as large as possible
- We want

$$w^T x - b = 1$$
$$w^T x - b = -1$$

- Training objective: minimize $\|w\|$ subject to $y_i(w^T x_i - b) \geq 1 for i = 1, \ldots, n$
- These constraints say that data points much lie on the correct side of the margin

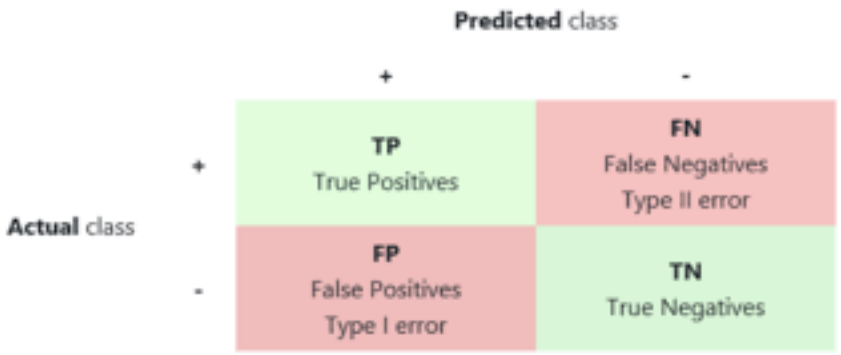
0.13 Precision v.s. Recall

- Precision (positive predictive value): the fraction of relevant instances among retrieved instances
- Recall (sensitivity): fraction of total amount of relevant instances that were retrieved

- F-measure: harmonic mean of precision and recall

$$\text{precision} = \frac{tp}{tp + fp}$$
$$\text{recall} = \frac{tp}{tp + fn}$$
$$\text{f-score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Figure 3: Confusion Matrix



- Type I error: false positive, Type II error: false negative
- Precision-recall tradeoff
- Precision is more important than recall when you would like to have less false positives in tradeoff to have more false negatives. Getting a false positive is very costly, and a false negative not as much
- For example, in a zombie apocalypse, you want to accept as many healthy people in the safe zone. Don't want to mistaken pass a zombie into safe zone (aka false positive). It's fine if some of the healthy people don't get into safe zone (aka false negative)
- Recall is more important when you want to capture all the positive cases. It is most costly to miss a positive than including a negative. Very important for medical purposes
- For example, we're more willing to tell someone they have a cancer than to let a positive slip through the crack. We don't care about false positives, but we want to get all the positives possible
- Accuracy is not a good metric when there is heavy class imbalance
- ROC plots the true positive rate against the false positive rate at various thresholds
- Area under ROC (AUROC)

0.14 Information Theory

0.14.1 Entropy

- Entropy of a random variable is the average level of "information", "surprise", or "uncertainty" in the variables possible outcomes.

$$H(X) = -\sum_{i=1}^{n} P(x_i)\log P(x_i)$$

- Created by Shannon as part of theory of communication
- Entropy is 0 means that the outcome is always certain, no new information

0.14.2 Cross Entropy

0.14.3 KL Divergence

- Measure of how one probability distribution is different from a second
- For two probability distributions P and Q

$$D_{KL}(P||Q) = \sum_{x\in\mathcal{X}} P(X)log(\frac{P(x)}{Q(x)}) = \sum_{x\in\mathcal{X}} P(X)log(\frac{Q(x)}{P(x)})$$

- $D_{KL}(P||Q)$ is often called the *information gain* if P were used instead of Q, it is also called the *relative entropy* of P wrt Q
- KL divergence is always non-negative, also known as Gibb's inequality, $D_{KL}(P||Q) \geq 0$
- KL divergence is not symmetric, hence it cannot be a "distance metric", $D_{KL}(P||Q) \neq D_{KL}(Q||P)$

0.14.4 Mutual Information

- hehehehe

$$I(X;Y) = D_{KL}(P(X,Y)||P(X)P(Y))$$

0.15 Decision Trees

0.16 K Nearest Neighbors

- non-parametric method used for classification and regression
- in classification the output is class membership: an object is classified by plurality vote of neighbors (mode)
- in regression, the output is a property value of an object, usually the average of its k-nearest neighbors

- compute distance between a query point and all other examples in the data, and selecting the K closest and take a vote or average depending on the problem
- distance metrics:
  - Euclidean distance
  - Taxicab / manhattan distance
  - Minkowski distance
  - Jaccard index
  - Hamming distance
- no need to build model, tune parameters
- KNN doesn't scale well as the number of examples increases

0.17 Logistic Regression

- a statistical model that uses a logistic function to model a binary dependent variable
- linear regression doesn't work for classification because a linear model doesn't output probabilities, also doesn't scale to classification problems with multiple classes
- logistic function

$$\text{logistic}(n) = \frac{1}{1 + \exp(-n)}$$

- the logistic function forces the output to assume values between 0 and 1
-
- Multinomial (softmax) Logistic Regression

0.18 Gradient Descent

0.19 Optimizers

0.20 Weak supervision

0.21 Transfer Learning

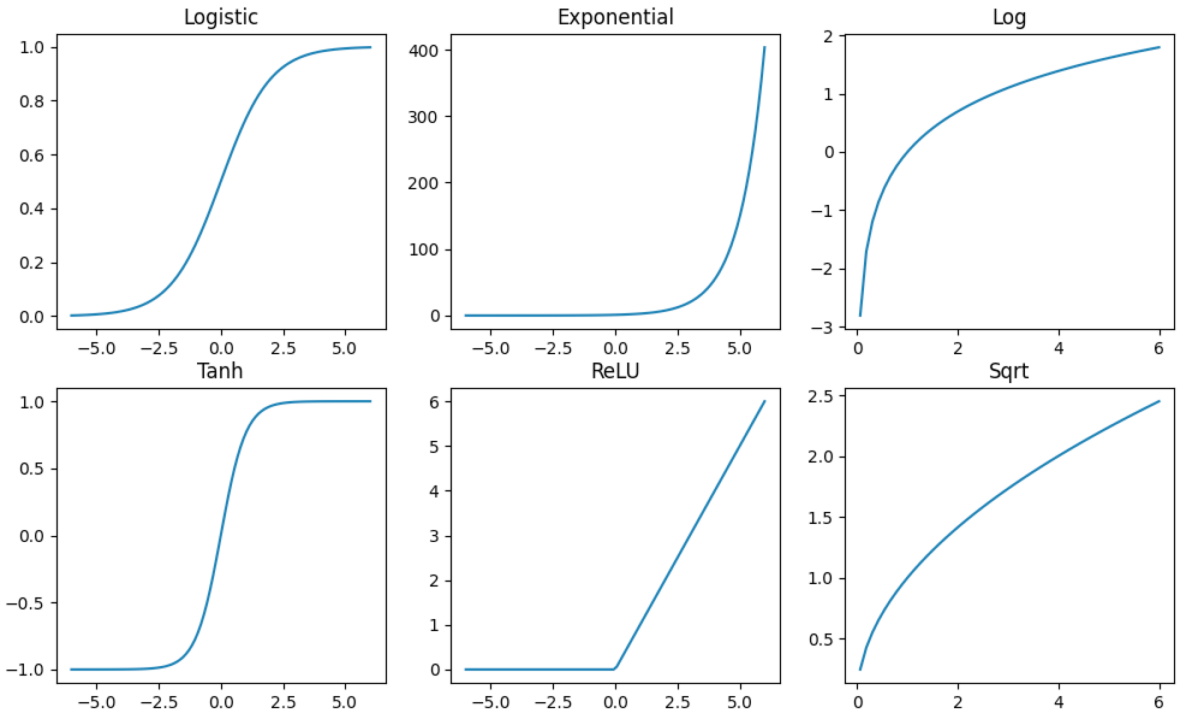
-

0.22 Graph of various useful functions

0.23 ML Tips and Advice

7 Steps to ML

Figure 4: Important functions

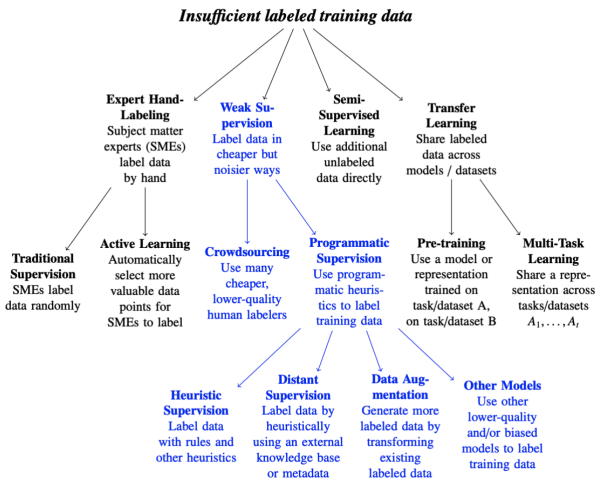


1. Acquire data
    - hidden data artifacts are challenging
2. Look at your data!
    - always look at your data!
    - have the right people look at your data (e.g. physicians)
3. Create train/dev/test splits
    - Fit model to training dataset
    - Fit hyperparameters to validation or development dataset
    - Test model performance on test set
    - Avoid leakage and adaptive overfitting
    - Splits should be randomly sampled, but depends on application
4. Create a specification
    - A good specification has little ambiguity
5. Build a model (simplest that works!)
    - Try linear of logistic regression w/ simple features
    - Good baseline for future work
    - Avoid getting bogged down in new models, use them to understand the data

- Ablation studies: figure out which part matters and which parts are stable
- Remove one feature at a time
- What could be wrong?
  - Maybe it's the data or your features? Try get more training data. Try smaller set of features. Try adding more features.
  - Maybe it's the optimization algorithm? Try running GD longer. Try SGD, GD, Newton, etc.
  - Maybe it's the hyperparamters?
  - Try different model?

6. Measurement
   - Measure end-to-end quality metrics
   - Automatic monitoring
   - Labels and input drift (change) over time
   - Variance diagnostics - use k-fold cross validation and check that dev scores have small relative error
7. Repeat
8. Lasso path: sweep regularize parameter for L1, train the model, and see when features turn on

Figure 5: Techniques for limited labeled data



9.