Linear Algebra Review

Tuesday, October 10, 2023 1:33 PM

Transpose: $(X^T)_{ij} = X_{ji}$

Dot Product: $x \cdot y = x_1 y_1 + \cdots + x_d y_d$

$x \cdot y = 0 \leftrightarrow x \perp y$

$x \cdot y = |x|^2$

$x \cdot y = y \cdot x$

Matrix-vector: $A_{r\times d} x_{d\times 1} = \begin{pmatrix} A_1 \cdot x \\ \vdots \\ A_r \cdot x \end{pmatrix}$

Identity: $I_d x = x$

Matrix-matrix: $A_{r\times k} B_{k\times p} = \begin{pmatrix} \square & \cdots & \square \\ \vdots & A_i \cdot B_j^T & \vdots \\ \square & \cdots & \square \end{pmatrix}$

Not commutative: $AB \neq BA$

Associative: $ABCD = (AB)(CD)$

Symmetry: $A_{ij} = A_{ji}$

Diagonal: $i \neq y \rightarrow A_{ij} = 0$

Linear Functions: $f(x) = Ax$

Quadratic Function: $f(x) = x^T A x$

Determinant: $|A|$

Inverse: $AA^{-1} = A^{-1}A = I$

Non-invertible matrix is called singular

$A^{-1} \leftrightarrow |A| \neq 0$

Nearest Neighbor, K Nearest Neighbor

Thursday, September 28, 2023 12:31 PM

Given training data $x_1 \ldots x_n$ and labels $y_1 \ldots y_n$,

We can classify new image x by finding its nearest neighbor x_i and returning y_i

Distance function: stretch each image into 1D int vector array, we can use Euclidean distance

$$||x - v|| = \sqrt{\sum (x_i + v_i)^2}$$

Problem: distance function is slow

Better distance functions:

ℓ_2	tangent distance	shape context
3.09	1.10	0.63

K Nearest Neighbor:

Classify point by using the labels of its k nearest neighbors

Speeding up NN Search for large N:

Locality sensitive hashing
Ball trees
K-d trees

Cross Validation

Thursday, September 28, 2023 1:26 PM

How to create a test set from training set.

1. Create a Hold-out set
   - Let S be the training set
   - Choose $V \subset S$ as validation set
   - Determine fraction of V which is misclassified
   - Not great at testing error rate on real data

2. Leave-one-out cross-validation
   - For each point $x \in S$, find the k-nearest neighbors in S without x
   - What fraction are misclassified?

3. m-fold cross validation
   - Divide training set into m pieces $S_1 \dots S_m$
   - For each piece S_i:

     Classify each point in S_i using k-NN with training set $S - S_i$

     Let ϵ_i be the fraction of S_i that is incorrectly classified
   - Average over all ϵ

Distance Functions, Metric Spaces

Tuesday, October 3, 2023 12:50 PM

Distance Functions

$$\ell_p(x, y) = \left(\sum_{i}^{d} |x_i - y_i|^p\right)^{\frac{1}{p}}$$

$$\ell_\infty(x, y) = \max(|x_i - y_i|)$$

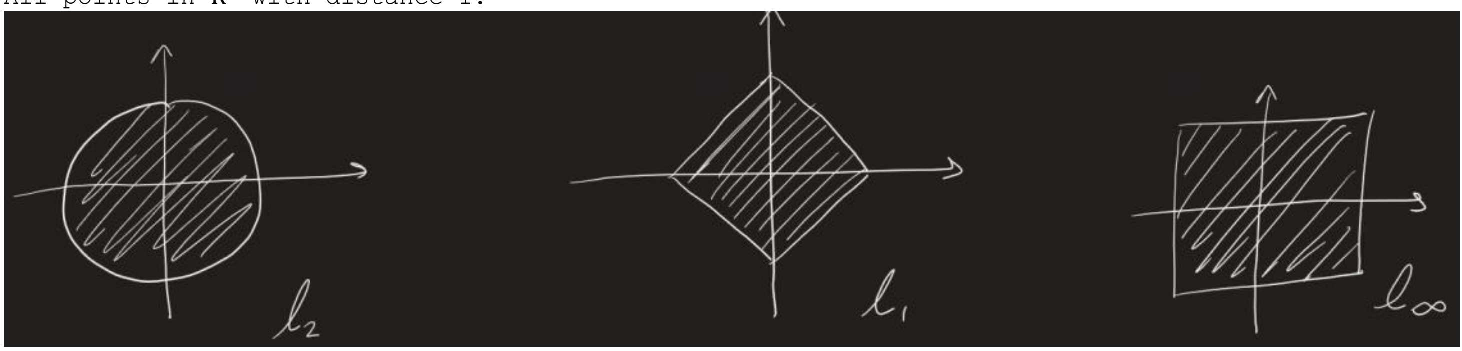
For the vector $(1 \ldots 1) \in R^d$:

$\ell_1 = d$

$\ell_2 = \sqrt{d}$

$\ell_\infty = 1$

All points in R^2 with distance 1:



Metric Spaces

Let X be the data space

A distance function $d: X * X \to R$ is a metric if:

- $d(x, y) \geq 0$
- $d(x, y) = 0 \leftrightarrow x = y$
- $d(x, y) = d(y, x)$
- $d(x, z) \leq d(x, y) + d(y, z)$

Prediction Problems

Tuesday, October 3, 2023 1:35 PM

Input space: X
Output space: Y

Discrete output space: classification
- Each possible output is uniquely different

Continuous output space: regression
- Possible outputs are ordered and results can be close

Probability output space: $Y = [0, 1]$

Binary Classification:
- Y is binary and discrete

Multiclass Classification:
- Y has many discrete values

Structured Classification:
- Y is a discrete structured value (eg. Tree)

Statistics

Thursday, October 5, 2023 12:35 PM

Mean:

$$E(X) = \sum_x x * \Pr(x)$$

$$E(X) = \int x * p(x)\, dx$$

Properties:

$$E(aX + b) = a * E(X) + b$$

Variance:

$$V(X) = E\left((X - \mu)^2\right) = E(X^2) - E(X)^2$$

Properties:

$$V(aX) = a^2 * V(X)$$

$$V(X + b) = V(X)$$

Standard Deviation:

$$\sigma = \sqrt{V(X)}$$

Independence: X, Y are independent if $\mathrm{P}(\mathrm{X}, \mathrm{Y}) = \mathrm{P}(\mathrm{X}) * \mathrm{P}(\mathrm{Y})$

Dependence:

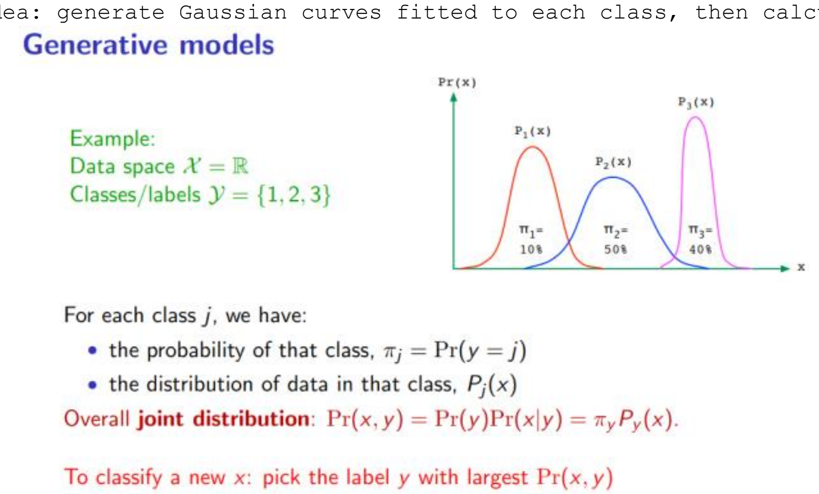
$$Cov(X, Y) = E(X * Y) - E(X) * E(Y)$$

$$Cor(X, Y) = \frac{Cov(X, Y)}{\sigma(X) * \sigma(Y)}$$

Classification

Thursday, October 5, 2023 1:34 PM

Idea: generate Gaussian curves fitted to each class, then calculate the probability of each class



Adding more than 1 feature: Multivariate Gaussian

$N(\mu, \Sigma)$ where $\mu_1 = \begin{pmatrix} E(X_1) \\ \vdots \\ E(X_d) \end{pmatrix}$ and $\Sigma_{ij} = \text{cov}\left(X_i, X_j\right)$

$p(x) = \frac{1}{(2\pi)^{\frac{d}{2}} * |\Sigma|^{\frac{1}{2}}} * \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\right)$ where $|\Sigma| = \det(\Sigma)$

Diagonal: X_i are independent, only the diagonals are non zero

Spherical: X_i are independent and have the same variance, only diagonals are non zero and are the same value

Typically, compute the log of probabilities because probabilities become tiny

Decision boundaries:

- Linear: Covariances are equal
- Spherical: Both distributions are spherical and unequal variances
- Quadratic: Everything else

Logistic Regression, Bag of Words

Thursday, October 26, 2023 12:48 PM

Idea: use a linear function as a decision boundary

Logistic regression: for data $x \in R^d$ and binary labels $y \in \{-1, 1\}$

$$P(y|x) = \frac{1}{1 + e^{-y(w \cdot x + b)}}$$

Or we can define $w = (w, b), x = (x, 1)$

$$P(y|x) = \frac{1}{1 + e^{-ywx}}$$

Training: maximize the likelihood $\prod P(y|x)$

Take the log to get the loss function:
$L(w, b) = \ln \sum (1 + e^{-y_i w x_i})$

Gradient descent:
Set $w_0 = 0$
For $t = 0, 1, 2, \ldots$ until convergence

$w_{t+1} = w_t + \eta_t \sum y_i x_i P(-y|x)$ where η_t is the step size

Bag of Words:
Fix V = some vocabulary
Treat each sentence (or document) as a vector of length |V|:
x_i is the number of times the ith word appears in the sentence

Margin and test error:
Margin(x) = $|P(y = 1|x) - \frac{1}{2}|$

Regression

Tuesday, October 17, 2023 1:36 PM

Ordinary least squares regression: Given dataset x, y

$f(x) = wx + b$

Minimize squared error:

$L(w, b) = \sum\left(y_i - (wx + b)\right)^2$

Solving: assimilate b into A

define $w = (b, w), x = (1, x)$ therefore $f(x) = wx$

Define $X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$

$L(w) = \sum\left(y_i - wx\right)^2 = |y - Xw|^2$ which is minimized at $w = \left(X^T X\right)^{-1}\left(X^T y\right)$

Ridge regression: penalize complex models

$L_R(w, b) = L(w, b) + \lambda \ell_2(w)^2$

$w = \left(X^T X + \lambda I\right)^{-1}\left(X^T y\right)$

$\lambda \approx 0$ when lot of data

$\lambda \to \infty$ when no data

$b = \mu_y - w\mu_x$

Lasso regression: tends to produce sparse w

$L_R(w, b) = L(w, b) + \lambda \ell_1(w)$

Optimization, Positive Semidefinite

Tuesday, October 31, 2023 12:32 PM

Local search: minimize a loss function iteratively

- Initialize weights (w) arbitrarily
- Repeat until w converges:
  - Find some w' close to w such that L(w') < L(w)
  - Move w to w'

Gradient descent: we can use the derivative to find w'

Then we can say the procedure is $w_{t+1} = w_t - \eta_t \nabla L(w_t)$

Step sizes:

Too small: not much progress

Too large: overshoot the mark

Picking step size: pick η_t using a line search

$\eta_t = \min L(w_t - \alpha \nabla L(w_t))$

Multivariate differentiation: Given $w \in R^k$

$$\nabla L(w) = \begin{bmatrix} \frac{dL}{dw_1} \\ \vdots \\ \frac{dL}{dw_k} \end{bmatrix}$$

Stochastic Gradient Descent: each update may involve the entire dataset, which is inconvenient

Cycle through the dataset and get each point (x, y):

$\nabla L(w) = -y * \Pr_{w_t}(-y|x)$

Minibatch Stochastic: each update involves a small batch of points

Cycle through the next batch of points B

Decomposable Loss Functions

$L(w) = \sum \ell(w, x_i, y_i))$

Convexity: Given $f(x): R^d \to R$

Convex iff = f'' is always positive semidefinite

Positive Semidefinite:

M is square

M is symmetric

$M = UU^T$

M has only positive eigenvalues

Alternatively, $x^T M x \geq 0$ for any x

Perceptron

Tuesday, November 7, 2023 12:35 PM

Linear classifier:

$\hat{y} = w \cdot x + b$

loss is $L(w, b) = -y(w \cdot x + b)$

Penalize wrong guesses which are very far from the decision boundary

Learning Algorithm: use stochastic gradient descent

- Initialize w=0, b=0
- Cycling through data (x,y)
  - If $y(w \cdot x + b) \leq 0$
    - $w = w + yx$
    - $b = b + y$

Support Vector Machines

Tuesday, November 7, 2023 1:04 PM

Idea: maximize the margin of the decision boundary on the data

Note that = $y(w \cdot x + b) > 0$ is equivaluent to $y(w \cdot x + b) \geq 1$ if we multiply w and b by some constant

Hard-margin SVM:

$\min(|w|^2)$ such that $y(w \cdot x + b) \geq 1$

The solution $w = \sum \alpha_i y^{(i)} x^{(i)}$ where $\alpha_i = 1$ only if function is on the margin

Soft-margin SVM: What if the data is not separable?

Allow each data point some slack

$\min(|w|^2) + C\sum \xi_i$ such that $y(w \cdot x + b) \geq 1 - \xi_i$ and $\xi \geq 0$

C manages the tradeoff between margin and slack

Duality in Linear Classification

Thursday, November 9, 2023 1:07 PM

Given training points x,y in the Perceptron algorithm:

A linear model solution has the form $w = \sum \alpha_i y^{(i)} x^{(i)}$

Where α_i is # of times update occurred on point I

Perceptron algorithm: primal form

- Initialize $w = 0$ and $b = 0$
- While some training point $(x^{(i)}, y^{(i)})$ is misclassified:
  - $w = w + y^{(i)} x^{(i)}$
  - $b = b + y^{(i)}$

Perceptron algorithm: dual form

- Initialize $\alpha = 0$ and ~~$b = 0$~~
- While some training point $(x^{(i)}, y^{(i)})$ is misclassified:
  - $\alpha_i = \alpha_i + 1$
  - ~~$b = b + y^{(i)}$~~

Where $w = \sum \alpha_i y^{(i)} x^{(i)}$, $b = \sum \alpha_i y^{(i)}$

Hard-margin SVM:

$$\text{(PRIMAL)} \quad \min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \|w\|^2$$

$$\text{s.t.: } y^{(i)}(w \cdot x^{(i)} + b) \geq 1 \quad \text{for all } i = 1, 2, \ldots, n$$

$$\text{(DUAL)} \quad \max_{\alpha \in \mathbb{R}^n} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} (x^{(i)} \cdot x^{(j)})$$

$$\text{s.t.: } \sum_{i=1}^{n} \alpha_i y^{(i)} = 0$$

$$\alpha \geq 0$$

Soft-margin SVM:

$$\text{(PRIMAL)} \quad \min_{w\in\mathbb{R}^d, b\in\mathbb{R}, \xi\in\mathbb{R}^n} \quad \|w\|^2 + C\sum_{i=1}^{n} \xi_i$$

$$\text{s.t.:} \quad y^{(i)}(w \cdot x^{(i)} + b) \geq 1 - \xi_i \quad \text{for all } i = 1, 2, \ldots, n$$

$$\xi \geq 0$$

$$\text{(DUAL)} \quad \max_{\alpha\in\mathbb{R}^n} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} (x^{(i)} \cdot x^{(j)})$$

$$\text{s.t.:} \quad \sum_{i=1}^{n} \alpha_i y^{(i)} = 0$$

$$0 \leq \alpha_i \leq C$$

Multiclass Classification, SoftMax

Tuesday, November 14, 2023 12:30 PM

Logistic: train a linear classifier for each class, predict the highest class value

Class 1: $w_1 x + b_1$
Class k: $w_k x + b_k$

SoftMax: $\mathrm{P}(y = j|x) = \frac{e^{w_j \cdot x + b_j}}{e^{w_1 \cdot x + b_1} + \cdots + e^{w_k \cdot x + b_k}}$

$$L(w,b) = -\sum_{i=1}^{n} \ln(P(y = y^{(i)}|x^{(i)}))$$

Perceptron: train a perceptron for each class, predict the highest class value

Class 1: $w_1 \cdot x + b_1$
Class k: $w_k \cdot x + b_k$

Training:

- Initialize $w_1 = \cdots = w_k = 0$ and $b_1 = \cdots = b_k = 0$
- Repeat while some training point (x, y) is misclassified:

$$\begin{aligned} \text{for correct label } y: \quad & w_y = w_y + x \\ & b_y = b_y + 1 \\ \text{for predicted label } \hat{y}: \quad & w_{\hat{y}} = w_{\hat{y}} - x \\ & b_{\hat{y}} = b_{\hat{y}} - 1 \end{aligned}$$

Maximum of (K choose 2) boundary pieces (depending on dimensionality)

Multiclass SVM:

Model: $w_1, \ldots, w_k \in \mathbb{R}^d$ and $b_1, \ldots, b_k \in \mathbb{R}$

Prediction: On instance x, predict label $\arg\max_j (w_j \cdot x + b_j)$

Learning. Given training set $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$:

$$\min_{w_1,\ldots,w_k \in \mathbb{R}^d, b_1,\ldots,b_k \in \mathbb{R}, \xi \in \mathbb{R}^n} \sum_{j=1}^{k} \|w_j\|^2 + C \sum_{i=1}^{n} \xi_i$$
$$w_{y^{(i)}} \cdot x^{(i)} + b_{y^{(i)}} - w_y \cdot x^{(i)} - b_y \geq 1 - \xi_i \quad \text{for all } i, \text{ all } y \neq y^{(i)}$$
$$\xi \geq 0$$

Kernel Machines, Basis Exapnsion

Tuesday, November 14, 2023 1:20 PM

Basis Expansion:

Idea: embed data in higher-dimension feature space, then use linear classifier

$\phi(x) = (x_1, \dots, x_d, x_1^2, \dots, x_d^2, x_1 x_2, \dots, x_{d-1} x_d)$

Dimensionality is $2d + dC2$

Kernel Perceptron:

Primal Form:

$w = 0$ and $b = 0$

while some $y(w \cdot \Phi(x) + b) \leq 0$:

- $w = w + y\,\Phi(x)$
- $b = b + y$

Computing and training: the data has high dimensionality

- Represent w in dual form $\alpha = (a_1, \dots, a_n)$
- Compute $w * \phi(x) = \sum \alpha_i y^{(i)} \left(\phi(x^{(i)}) \cdot \phi(x)\right) + b$
- $\phi(x) \cdot \phi(z) = (1 + x \cdot z)^2$

Dual form: $w = \sum_j \alpha_j y^{(j)} \Phi(x^{(j)})$, where $\alpha \in \mathbb{R}^n$

- $\alpha = 0$ and $b = 0$
- while some i has $y^{(i)} \left(\sum_j \alpha_j y^{(j)} \Phi(x^{(j)}) \cdot \Phi(x^{(i)}) + b\right) \leq 0$:
  - $\alpha_i = \alpha_i + 1$
  - $b = b + y^{(i)}$

To classify a new point x: sign $\left(\sum_j \alpha_j y^{(j)} \Phi(x^{(j)}) \cdot \Phi(x) + b\right)$.

Kernel SVM:

1. **Basis expansion.** Mapping $x \mapsto \Phi(x)$.
2. **Learning.** Solve the dual problem:

$$\max_{\alpha \in \mathbb{R}^n} \sum_{i=1}^{n} \alpha_i - \sum_{i,j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} (\Phi(x^{(i)}) \cdot \Phi(x^{(j)}))$$

$$\text{s.t.: } \sum_{i=1}^{n} \alpha_i y^{(i)} = 0$$

$$0 \leq \alpha_i \leq C$$

This yields $w = \sum_i \alpha_i y^{(i)} \Phi(x^{(i)})$. Offset b also follows.

3. **Classification.** Given a new point x, classify as

$$\text{sign}\left(\sum_i \alpha_i y^{(i)} (\Phi(x^{(i)}) \cdot \Phi(x)) + b\right).$$

Kernel Functions:

Let $\phi(x)$ consist of terms of order $\leq p$

Then $\phi(x) \cdot \phi(z) = (1 + x \cdot z)^p$

In general we can define $k(x, z) = \phi(x) \cdot \phi(z)$

- Similarity between x and z
- Pick any similarity function -> new decision boundary

However $K_{i,j} = k\left(x^{(i)}, x^{(j)}\right)$ must be PSD

RBF Kernel:

$k(x,z) = e^{-\frac{|x-z|^2}{s^2}}$ where s is an ajdustable scale parameter

As s increases, k(x,z) approaches 1, and decision will produce the same output everywhere
As s decreases, k(x,z) behaves like Nearest Neighbor
As we get more data, we should decrease s

Decision Trees

Friday, November 24, 2023 6:56 PM

Create a tree where each node separates the data by some decision.
- Accommodates any type of data (real, Boolean, categorical)
- Can accommodate any number of classes
- Can fit any data set
- Statistically consistent

Greedy algorithm: build tree top-down.

- Start with a single node containing all data points
- Repeat:
  - Look at all current leaves and all possible splits
  - Choose the split that most decreases the uncertainty in prediction

Uncertainty:

	$k = 2$	General k
Misclassification rate	$\min\{p, 1-p\}$	$1 - \max_i p_i = 1 - \|p\|_\infty$
Gini index	$2p(1-p)$	$\sum_{i \neq j} p_i p_j = 1 - \|p\|^2$
Entropy	$p \log \frac{1}{p} + (1-p) \log \frac{1}{1-p}$	$\sum_i p_i \log \frac{1}{p_i}$

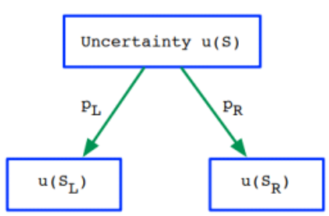
Where p_k represent the probability that a point contained by the node is classified label k

Choosing a Split:

Benefit of a split

Let $u(S)$ be the uncertainty score for a set of labeled points S.

Consider a particular split:



Of the points in S:
- p_L fraction go to S_L
- p_R fraction go to S_R

Benefit of split = reduction in uncertainty:

$$\left(u(S) - \underbrace{(p_L\, u(S_L) + p_R\, u(S_R))}_{\text{expected uncertainty after split}} \right) \times |S|$$

Pick the split where $\left[u(S) - \left(p_L u(S_L) + p_R u(S_R)\right)\right] \times |S|$ is maximized

Number of splits: Given d features and n datapoints, there are $(n-1) * d$ possible splits

Overfitting: Given enough nodes we can perfectly fit the data
- Train the tree to perfectly fit, then use pruning to correct for overfitting
- Pruning: use separate validation set, choose pruning that works best

Ensemble Methods, Random Forest

Friday, November 24, 2023 7:45 PM

Idea: want to combine different models
- No one classifier will be the final product: keep components simple
- How to train each component: on full training set? Or just on the errors?
- Combined model may be enormous

AdaBoost: Combine weak learners to boost overall performance

Weak learner: a model which is somewhat better than random guessing

Data set $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$, labels $y^{(i)} \in \{-1, +1\}$.

1. Initialize $D_1(i) = 1/n$ for all $i = 1, 2, \ldots, n$
2. For $t = 1, 2, \ldots, T$:
   - Give D_t to weak learner, get back some $h_t : \mathcal{X} \to [-1, 1]$
   - Compute h_t's margin of correctness:

$$r_t = \sum_{i=1}^{n} D_t(i) y^{(i)} h_t(x^{(i)}) \in [-1, 1]$$

$$\alpha_t = \frac{1}{2} \ln \frac{1 + r_t}{1 - r_t}$$

   - Update weights: $D_{t+1}(i) \propto D_t(i) \exp\left(-\alpha_t y^{(i)} h_t(x^{(i)})\right)$
3. Final classifier: $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

Suppose that on each round t, the weak learner returns a rule h_t whose error on the time-t weighted data distribution is $\leq 1/2 - \gamma$.

Then, after T rounds, the training error of the combined rule

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

is at most $e^{-\gamma^2 T/2}$.

Random Forest:

Given a data set S of n labeled points:
- For 1 to T:
  - Sample n' points randomly with replacement from S
  - Fit a decision tree h_t to the points
    - At each node restrict to one of k features chosen at random

Final predictor: majority vote of $h_1 \ldots h_T$

Neural Nets

Thursday, November 30, 2023 1:03 PM

Neural Nets:

Input layer -> hidden layers -> output layer

For some layer h with previous layer z

$h = t(w * z + b)$

Where t is a non-linear activation function

- Threshold function or Heaviside step function

$$\sigma(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

- Sigmoid

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

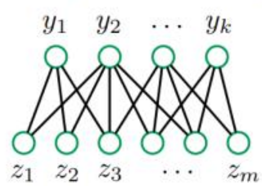
- Hyperbolic tangent

$$\sigma(z) = \tanh(z)$$

- ReLU (rectified linear unit)

$$\sigma(z) = \max(0, z)$$

Classification with k labels: want k probabilities summing to 1.



- $y_1, \ldots, y_k$ are linear functions of the parent nodes z_i.
- Get probabilities using **softmax**:

$$\Pr(\text{label } j) = \frac{e^{y_j}}{e^{y_1} + \cdots + e^{y_k}}.$$

A neural net with one hidden layer approximates any function arbitrarily well

- But use more layers to avoid an enormous layer

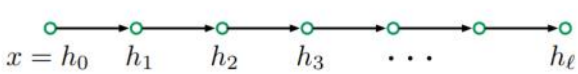
Optimization: use gradient descent

- For each parameter w, get the derivative and use gradient descent

  $L(W) = -\sum_{i=1}^{n} \ln(\Pr_W(y^{(i)}|x^{(i)}))$
- Solving for all weights as the same time: backpropagation

  Chain rule: if x -> y -> z then $\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$

  Therefore:
  - On a single forward pass, compute all the h_i.
  - On a single backward pass, compute $dL/dh_\ell, \ldots, dL/dh_1$



From $h_{i+1} = \sigma(w_{i+1}h_i + b_{i+1})$, we have

$$\frac{dL}{dh_i} = \frac{dL}{dh_{i+1}} \frac{dh_{i+1}}{dh_i} = \frac{dL}{dh_{i+1}} \sigma'(w_{i+1}h_i + b_{i+1})\, w_{i+1}$$

The derivative of ith layer depends on the derivative of the forward layer, derivative of the activation function, and weights of the forward layer.

Issues in training:

- Overfitting: stop early to avoid overfitting, use validation set
- Dropout: for each batch delete each hidden unit with probability 1/2 independently
- Batch normalization: normalize each layer so that each node's values are normalized so that mean=0, var=1
- Variants of SGD:

Suppose we have parameters θ and loss $\ell(x, y; \theta)$. Usual SGD update:

$$\theta^{(t+1)} = \theta^{(t)} - \eta_t g^{(t)}$$

where $g^{(t)} = \nabla \ell(x_t, y_t; \theta^{(t)})$ is the gradient at time t.

- **Momentum**: Accumulate gradients. For $g^{(t)}$ as above, and $v^{(0)} = 0$,

$$v^{(t)} = \mu v^{(t-1)} + \eta_t g^{(t)}$$
$$\theta^{(t+1)} = \theta^{(t)} - v^{(t)}$$

- **AdaGrad**: Different learning rate for each parameter, automatically tuned.

$$\theta_j^{(t+1)} = \theta_j^{(t)} - \frac{\eta}{\sqrt{\sum_{t'<t}(g_j^{(t')})^2 + \epsilon}} g_j^{(t)}$$

Many others: **Adam**, **RMSProp**, etc.

Generalization

Wednesday, December 6, 2023 10:39 PM

Idea: we want a model that does well on the underlying distribution of data
- Hope that the training set is representative of the data

Statistical Learning Framework:

A **learning task** is defined by:

- Instance space $\mathcal{X}$ and label space $\mathcal{Y}$
- Distribution P on $\mathcal{X} \times \mathcal{Y}$

All data (x, y) come from P.

A classifier is a function $h : \mathcal{X} \to \mathcal{Y}$. Its **true error rate** is

$$\text{err}(h) = \Pr_{(x,y)\sim P}(h(x) \neq y).$$

For a training set $(x_1, y_1), \ldots, (x_n, y_n)$, the **training error** of h is

$$\text{err}_n(h) = \frac{|\{i : h(x_i) \neq y_i\}|}{n}.$$

If the training set comes from P, and if n is large, then $\text{err}_n(h) \approx \text{err}(h)$.

How many training points?
- More complex models require more data, simpler models require less data

Distribution shifts

Covariate shift:
- Assumption that training data represent the distribution
- May encounter regions of the input space not represented by training set
- Distribution of data changes but the probable labels remain the same

  Examples:
- Speech recognizer trained on US speakers, fails on UK speakers

Label shift:
- Relative frequency of labels changes but distribution for each label remains unchanged

  Examples:
- As time goes on, prevalence of different diseases changes