Hypothesis Space for MLPs
Hypothesis Space for MLPs

Notice several properties:

1) There are many alternative ways to adjust to a misclassified training example (motivates minibatch training)
2) Any adjustment to a misclassified examples has non-local effects
3) Most regions do not use most hyperplanes
   • Representational efficiency
   • Redundant computations in a naïve parallelization
4) Hyperplanes induce many specious regions
   • Many regions may be formed that include no training examples
   • But the classifier will have an opinion
   • OK if these examples are unlikely but can yield bizarre behavior

These are unfortunate properties
Activation Functions

- **Sigmoid** (now “bias” rather than “threshold”)
- **Hyperbolic tangent**
- **Rectified linear unit (ReLU)**
- **ArcTangent**
- **Leaky ReLU**
- **many others** (some with additional shape parameters)

These change the ANN response within a region (esp. near boundaries)
Two Wrinkles

• This is a greedy / hill-climbing / gradient descent algorithm
  – So...
  – Is Expected Utility/Loss convex in \( w \)?
    For: \( \hat{L}_Z = F(Z, w) \)
  – No, far from it...

• ANN = structure + weights
  – We can adjust the weights
  – How do we choose a structure? (Wrinkle #1)
    – Structure makes a difference and it is combinatorial...

• Some heuristic approaches
  – Based on non-systematic search (remember simulated annealing)
  – Random restarts
  – Weight resets
  – Incremental structure changes
  – Ablations / Additions
With enough units, an ANN (MLP) can learn any assignment of training labels (Wrinkle #2)

Is this a good thing?

No, perfect in training $\Rightarrow$ poor in test

Synonymous parametric optimizers | training
Overfitting! (seems less in ConvNets [but share other issues])
Overfitting in Neural Nets

• With enough hidden units
  – Can achieve perfect training accuracy
  – Will overfit
  – Will perform poorly on new inputs

• Reduce the expressiveness
  – Specify a known-impoverished neural net structure...
  – Limit the dynamic range of weights (limit their absolute value)
  – Delete certain units and retrain
  – ...

• “Regularization” (important term)
  – Often a policy or procedure (extra loss) included in the classifier
  – Reduces expressiveness but can be sensitive to the training
  – To avoid overfitting and improving test behavior
  – Stabilizes learning by increasing bias and reducing variance
  – Serves as a kind of prior distribution over $H$
    (eg, Prefer simplicity, Prefer parameters w/ a small dynamic range)
Back to Perceptrons

• Linearity initially killed interest in perceptrons
  – Features contribute independently
  – Features contribute monotonically

• Non-learnable problematic phenomena:
  – non monotonicity
    • # wheels to recognize a car  peak at 4 wheels
    • # engines...
  – context dependent sign of partial derivative
    • XOR when x0 = 1,  x1: 0→1 turns output off to on
    • but when x0=0,  x1: 0→1 turns output on to off

• Rescued by ANNs as multi-layer perceptrons

• More recent renaissance of the humble perceptron
  (and other linear classifiers)
Perceptron Renaissance

• Classify the species of gilled mushroom (Agaricus vs. Leptota Family)
  – 100 Boolean features for each instance
  – How well can a perceptron do?
  – Quite poorly; there is no simple rule; the problem is not linear

• Classify news articles by topic (Sports vs. Politics)
  – How well can a perceptron do?
  – Quite well; why? how?? This should be harder than gilled mushrooms
Features
AI’s dirty little secret

• Today’s AI successes rely on machine learning
• Machine learning is largely statistical
• So... think statistically (not functionally, not logically)
• Mushrooms vs. News Stories, what’s the difference?
• Features!
  – Feature engineering (for car |wheels – 4| is a great feature)
  – Feature selection / creation
  – Dimensionality reduction
  – Finding a good set of features is
    • crucial to success
    • a black art; not science...
• Features should encode evidence for classification
• Redundancy is usually very desirable
• Mushrooms: 100 Booleans; News Articles: ~20,000 Booleans
Deep Dive into NLP Features
Simple Natural Language Processing
How to represent a news article to a classifier

• Nuanced deep natural language understanding may be in the future...not for today’s AI
• Simple unstructured text analysis
  – Spam vs. Useful email
  – Transcriptions of Fox vs. CNN news stories
  – Obama’s speeches vs. Trump’s speeches
  – rec.sport.baseball vs. talk.politics.guncontrol
  – Tweet analysis
  – Detect false (alternative?) Amazon / Yelp recommendation
• *Bag of words* model works surprisingly well
• Best with lots of training data
• What’s a *bag*? What’s a *set*?
• Simple membership vs. cardinality
Bag of Words

• Text article is a *sequence* of words and punctuation
• 10,000 – 50,000 common words in English
• Represent a document as a bag of words; lose syntax (!)
• News article is a vertex in 10K Boolean hypercube
• Finite number of news articles?
• $2^{10,000}$ vertices (about $2^{266}$ atoms in universe)
Bag of Words

• Consider two utterances:
  – John likes to watch movies. Mary likes movies too.
  – John also likes to watch football games.

• For these “documents” the universe of (all 10) words:
  – { "John", "likes", "to", "watch", "movies", "also", "football", "games", "Mary", "too" }

• Each utterance / document can be represented as a 10-entry vector:
  – [1, 2, 1, 1, 2, 0, 0, 0, 1, 1]
  – [1, 1, 1, 1, 0, 1, 1, 1, 0, 0]
Two Obvious Problems

• Problem 1: these utterances do not mean the same thing
  – Dogs chase cats.
  – Cats chase dogs.
• But they have the same representation
• Solution: NONE – don’t use BOW,
  BOW is very simple and doesn’t work for everything!
  Rely on the redundancy of BIG statistics
• Problem 2: with 50,000 possible words the representation really looks like
  – [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...]
    0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
    0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, ...
• Solution: Sparse vector representation:
  – {“John” 1, “likes” 2, “to” 2...}
Refinements
What’s wrong with a Bag?

• A count of the words is not so meaningful
  – document length is too prominent
  – longer documents have higher counts
  – compare
    • the representation of a long document
    • the first half of the same long document

• Similar documents can look very different

• Unfortunate for statistical modeling
Normalized Term Frequency
(sometimes just TF)

- Normalize the simple count
  - instead of raw counts
  - measure proportion / probability / rate that term t is used in a document d

\[ TF(t,d) = \frac{\text{raw count of } t \text{ in } d}{\text{length of } d} \]

- Many variants
  - should numerator grow linearly with count?
  - is document length the best normalization?
Normalized Term Frequency

• As features
  – document d has features \{t_i\} (ie, words name features)
  – each feature \(t_i\) has feature value \(\text{TF}(t_i,d)\)

• Information Retrieval (IR)
  – query determines features of interest
  – a document’s relevance is judged by combining feature values (eg, \(\text{argmax} \ \text{dot product of query & documents}\))

• Classification of a new document
  – there is some probability it came from each class
  – each document is a draw from its topic’s underlying distribution
  – use training documents to estimate these class distributions
  – assign new document to the class that makes its draw most likely
TF as Classification Features

- TF is, in a sense, universal or document-blind
- $\text{TF}(t,d) = \frac{\text{raw count of } t \text{ in } d}{\text{length of } d}$
  Approximates $\text{Pr}(t \mid d)$
- Features apply to any document set the same way
- Value is independent of discriminating power on this set of document classes
- Appreciate that (with similar TF) some terms may be more discriminative for these documents than others
- Introduce IDF, Inverse Document Frequency
Inverse Document Frequency

• With little effort the discriminating power of TF features can be improved
• For term $t$ (“terms” could be just “words” – more later)
  \[\text{IDF}(t) = \log\left(\frac{\# \text{ of documents}}{\# \text{ of documents w/ t}}\right)\]
• Again variants are possible
• Use words as features (as before)
• But the feature value for term $t$ in document $d$ becomes \(\text{TF}(t, d) \cdot \text{IDF}(t)\)
• Known as TF-IDF
• Thus TF is then specialized to this document set
Further Improvements

• Often punctuation is dropped
• Convert everything to all lower case
• Stop words
  – How about the words “the” or “sesquipedality”
  – Some words are too common to be discriminative
  – Others are too unlikely to be seen
  – Stop words: a list of words to be removed from the feature set
• Stemming
  – combine variants of a word
  – earthquake and earthquakes
  – eat and eating
• N-grams: keep statistics on term sequences
  – use pairs of terms (2-gram), triples (3-gram), etc.
  – requires many more training inputs (why?)
Further Improvements

• Synonym sets
  – augment (replace?) words with designated synonyms (e.g., WordNet synsets; “big” “large” “huge”...)

• Named entity resolution (like lexical analysis)
  – replace “the Statue of Liberty” with a single token

• Guess at the word sense using a window around the target word:
  table1: furniture; table2: mathematical illustration;
  table3: half of a backgammon board;...

• Include the part of speech POS; shallow syntactic parsing
  – “saw” noun vs. “saw” verb
Perceptron Renaissance

• Even the lowly perceptron can perform quite well
• Given a good set of features
  – class probabilities monotonic in feature values
  – many not-completely-redundant information-bearing features
• Given adequate training examples
  – for NLP often hundreds of thousands / millions of utterances
  – there are a number of corpora
  – luckily we have the web...
Back to ANNs

• Where are the features?
• After the input, each layer “invents” features to present to the next layer
• No feature engineering!
• Never worked very well
• Until ConvNets
• Use convolution instead of perceptron units
Convolution: Sobel Edge Detector

\[
\begin{array}{ccc}
-1 & 0 & +1 \\
-2 & 0 & +2 \\
-1 & 0 & +1 \\
\end{array}
\quad
\begin{array}{ccc}
+1 & +2 & +1 \\
0 & 0 & 0 \\
-1 & -2 & -1 \\
\end{array}
\]

G_x 
G_y