Linear Models: Pros and Cons

Pros:

- Simple and easy to train.
- Fast prediction.
- Scales well to very large datasets.
- Works well with sparse data.
- Reasons for prediction are relatively easy to interpret.

Cons:

- For lower-dimensional data, other models may have superior generalization performance.
- For classification, data may not be linearly separable (more on this in SVMs with non-linear kernels)

Kernelized Support Vector Machines: pros and cons

Pros:

- Can perform well on a range of datasets.
- Versatile: different kernel functions can be specified, or custom kernels can be defined for specific data types.
- Works well for both low- and high-dimensional data.

Cons:

- Efficiency (runtime speed and memory usage) decreases as training set size increases (e.g. over 50000 samples).
- Needs careful normalization of input data and parameter tuning.
- Does not provide direct probability estimates (but can be estimated using e.g. Platt scaling).
- Difficult to interpret why a prediction was made.

Decision Trees: Pros and Cons

Pros:

- Easily visualized and interpreted.
- No feature normalization or scaling typically needed.
- Work well with datasets using a mixture of feature types (continuous, categorical, binary)

Cons:

- Even after tuning, decision trees can often still overfit.
- Usually need an ensemble of trees for better generalization performance.

Naïve Bayes classifiers: Pros and Cons

Pros:

- Easy to understand
- Simple, efficient parameter estimation
- Works well with high-dimensional data
- Often useful as a baseline comparison against more sophisticated methods

Cons:

- Assumption that features are conditionally independent given the class is not realistic.
- As a result, other classifier types often have better generalization performance.
- Their confidence estimates for predictions are not very accurate.

Random Forest: Pros and Cons

Pros:
- Widely used, excellent prediction performance on many problems.
- Doesn't require careful normalization of features or extensive parameter tuning.
- Like decision trees, handles a mixture of feature types.
- Easily parallelized across multiple CPUs.

Cons:
- The resulting models are often difficult for humans to interpret.
- Like decision trees, random forests may not be a good choice for very high-dimensional tasks (e.g. text classifiers) compared to fast, accurate linear models.

GBDT: Pros and Cons

Pros:

- Often best off-the-shelf accuracy on many problems.
- Using model for prediction requires only modest memory and is fast.
- Doesn't require careful normalization of features to perform well.
- Like decision trees, handles a mixture of feature types.

Cons:

- Like random forests, the models are often difficult for humans to interpret.
- Requires careful tuning of the learning rate and other parameters.
- Training can require significant computation.
- Like decision trees, not recommended for text classification and other problems with very high dimensional sparse features, for accuracy and computational cost reasons.

Neural Networks: Pros and Cons

Pros:

- **They form the basis of state-of-the-art models and can be formed into advanced architectures that effectively capture complex features given enough data and computation.**

Cons:

- **Larger, more complex models require significant training time, data, and customization.**
- **Careful preprocessing of the data is needed.**
- **A good choice when the features are of similar types, but less so when features of very different types.**

Pros and Cons of Deep Learning

- **Pros:**
  - *Powerful: deep learning has achieved significant gains over other machine learning approaches on many difficult learning tasks, leading to state-of-the-art performance across many different domains.*
  - *Does effective automatic feature extraction, reducing the need for guesswork and heuristics on this key problem.*
  - *Current software provides flexible architectures that can be adapted for new domains fairly easily.*
- **Cons:**
  - *Can require huge amounts of training data.*
  - *Can require huge amounts of computing power.*
  - *Architectures can be complex and often must be highly tailored to a specific application.*
  - *The resulting models may not be easily interpretable.*