Summary: feature engineering

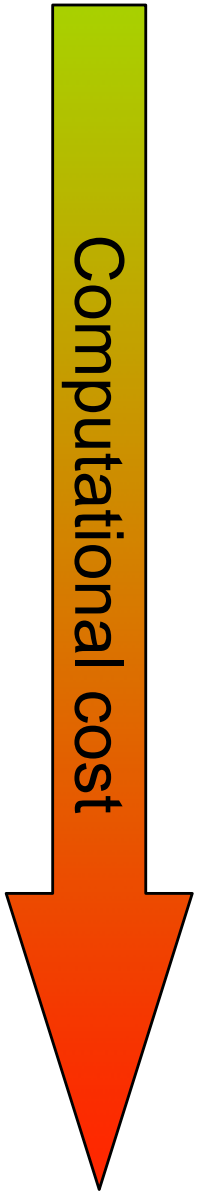
- Feature engineering is often crucial to get good results
- Strategy: overshoot and regularize
  - Come up with lots of features: better to include irrelevant features than to miss important features
  - Use regularization or feature selection to prevent overfitting
  - Evaluate your feature engineering on DEV set. Then, when the feature set is frozen, evaluate on TEST to get a final evaluation (Daniel will say more on evaluation next week)

Summary: feature selection

When should you do it?

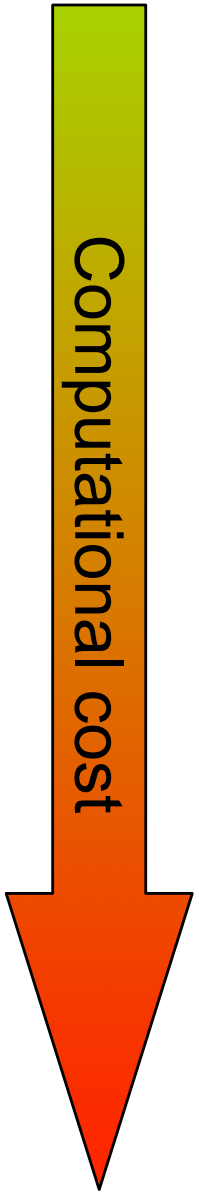
- If the only concern is accuracy, and the whole dataset can be processed, feature selection not needed (as long as there is regularization)
- If computational complexity is critical (embedded device, web-scale data, fancy learning algorithm), consider using feature selection
  - But there are alternatives: e.g. the Hash trick, a fast, non-linear dimensionality reduction technique [Weinberger et al. 2009]
- When you care about the feature themselves
  - Keep in mind the correlation/causation issues
  - See [Guyon et al., Causal feature selection, 07]

Summary: how to do feature selection



- Filtering
- L_1 regularization (embedded methods)
- Wrappers
  - Forward selection
  - Backward selection
  - Other search
  - Exhaustive

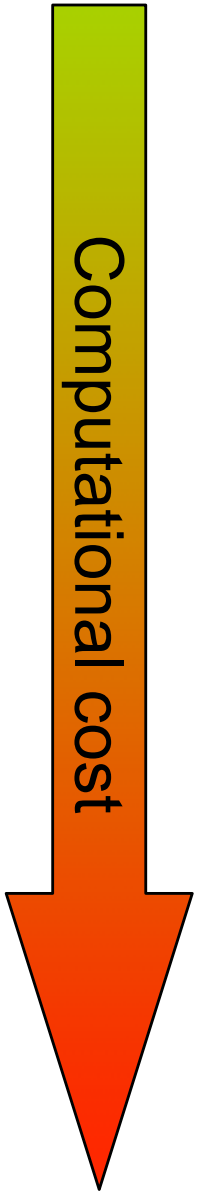
Summary: how to do feature selection



- *Filtering*
- L_1 regularization (embedded methods)
- Wrappers
  - Forward selection
  - Backward selection
  - Other search
  - Exhaustive

- Good preprocessing step
- Fails to capture relationship between features

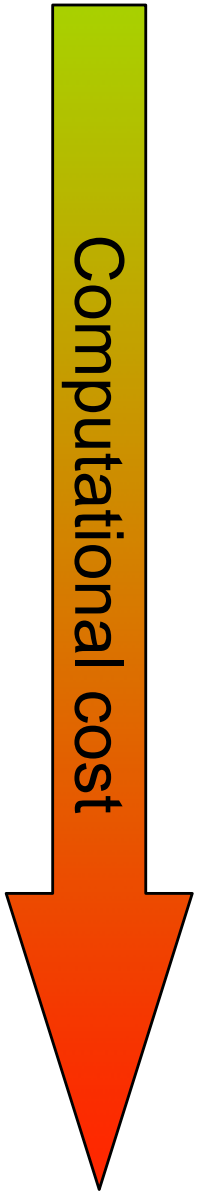
Summary: how to do feature selection



- Filtering
- *L_1 regularization (embedded methods)*
- Wrappers
  - Forward selection
  - Backward selection
  - Other search
  - Exhaustive

- Fairly efficient
  - LARS-type algorithms now exist for many linear models.

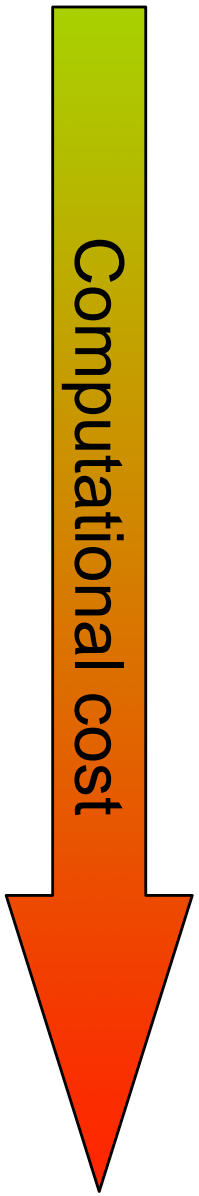
Summary: how to do feature selection



- Filtering
- L_1 regularization (embedded methods)
- *Wrappers*
  - Forward selection
  - Backward selection
  - Other search
  - Exhaustive

- Most directly optimize prediction performance
- Can be very expensive, even with greedy search methods
- Cross-validation is a good objective function to start with

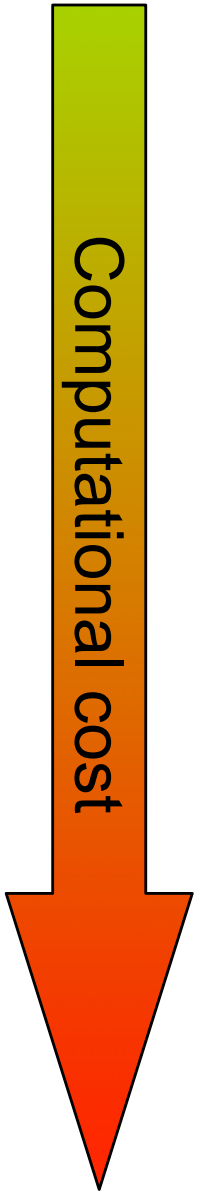
Summary: how to do feature selection



- Filtering
- L_1 regularization (embedded methods)
- Wrappers
  - *Forward selection*
  - *Backward selection*
  - Other search
  - Exhaustive

- Too greedy—ignore relationships between features
- Easy baseline
- Can be generalized in many interesting ways
  - Stagewise forward selection
  - Forward-backward search
  - Boosting

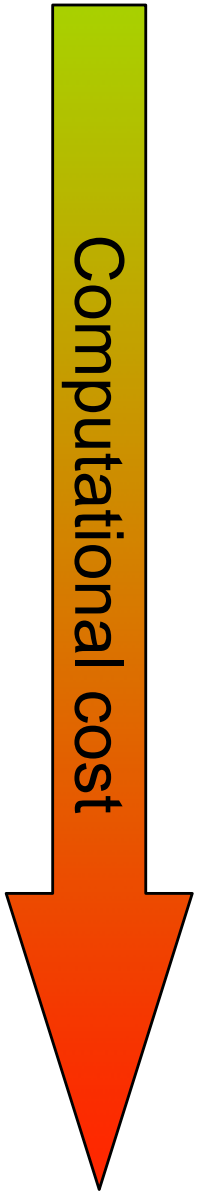
Summary: how to do feature selection



- Filtering
- L_1 regularization (embedded methods)
- Wrappers
  - Forward selection
  - Backward selection
  - *Other search*
  - Exhaustive

- Generally more effective than greedy

Summary: how to do feature selection



- Filtering
- L_1 regularization (embedded methods)
- Wrappers
  - Forward selection
  - Backward selection
  - Other search
  - *Exhaustive*

- The "ideal"
- Very seldom done in practice
- With cross-validation objective, there's a chance of over-fitting
  - *Some* subset might randomly perform quite well in cross-validation