Today:
- Seq2Seq model
- Learning to search
- Submodularity

Beam search:

Example: \[
\begin{align*}
\log p(y_t | y_{<t}, x) \quad \text{for} \quad t \in [1, k] \\
\text{beam of size } L \quad \text{(memory)}
\end{align*}
\]
- After \( k \), you have \( L \) candidate solution prefixes: \( y_1^{(k)}, y_2^{(k)}, \ldots, y_L^{(k)} \)
- Expand possible next choice: \( L \cdot \text{size of } \{a, b, c\} \)
  - Score them (e.g. \( \log p(y_t | y_{<t}, x) \))
  - Then keep \( L \) candidates as \( y_1^{(k)}, y_2^{(k)}, \ldots, y_L^{(k)} \)

(approximate search, vs. Viterbi which does "backtracking" to correct past mistakes)

Generalization of greedy search to \( L \)-candidates
pre-greedy \( \Rightarrow L = 1 \)

Seq2Seq | encoder/decoder

* Useful way to get \( p(y_t | y_{<t}, x) \) for RNN

\[
\begin{align*}
& h_1, h_2, \ldots, h_T \\
& y_t
\end{align*}
\]
Learning to search (L2S)

\[ h_{w}: x \rightarrow y \]

\[ h_{w}(x) = \arg\max_{y \in Y} s(x; y; w) \]

Learning to search:  split `y` into ordered set of decisions

\[ (y_1, \ldots, y_T) \]

Learn a policy \( \pi_t(x_{1:t}) = y_t \)

Policy which takes an history to make next decision

\text{L2S: } \text{from} \{x^n(i), y^n(i)\}_{i=1}^N \text{ and } \ell(y, y^i) \]

Learn a good policy \( \pi_{w_0} \) s.t.

\[ \hat{y}_t = \pi_{w_0}(\hat{y}_{t-1} ; x_t) \]

Issues:

a) variable length output?  
   \( \rightarrow \) end of sequence special character

b) large input sequence \( x \)  
   needs to be summarized in fixed length  
   content vector  
   \( \rightarrow \) attention mechanism

c) vanishing gradient?  

- LSTM  
- gated recurrent unit (GRU)
\[
\begin{align*}
\pi_u^*(x) &= \left( \hat{x}_1, \ldots, \hat{x}_t \right) \\
\text{generate training data for } \pi_u \text{ as a multiclass classifier} \\
\text{roll-in} &\rightarrow \text{determines how you get } \hat{y}_{t-1} \text{ context} \\
\text{roll-out} &\rightarrow \text{determines how you get } y_t, \ldots, y_T \text{ "target"} \\
\end{align*}
\]

\[
\begin{align*}
\forall \theta \in \Theta, \quad L(y_t, \hat{y}_t) \overset{\text{goal}}{=} L(y_t, \hat{y}_t) \\
\text{LDSL ICML 2015} \\
\text{Locally Optimized Learning to Search} \\
\end{align*}
\]

- Centralized here is "reduction" of learning problem to another
- [Langford]

\[
\begin{align*}
\text{Submodularity} \\
\text{Submodularity is analog of convexity for tractable set functions (combinatorial optimization)} \\
F : \mathcal{P}(V) \rightarrow \mathbb{R} \\
V = \{x_1, \ldots, x_n\} \text{ is "ground set"} \\
\end{align*}
\]

WLOG \( F(\emptyset) = 0 \)
V = \{e_1, \ldots, d, i\} is "ground set"

Concrete example: Using model \(y_i \in \{0, 1\}\)

\[ E(y) = \sum_i \sum_j \text{higher} \]

\[ F(A) = \sum_i y_i \]

Where \( A_y = \sum_i y_i \)

\( F \) is submodular \( \Leftrightarrow \) \( F(A) + F(B) \geq F(A \cup B) + F(A \cap B) \) \( \forall A, B \)

\( \Leftrightarrow \) function \( A \mapsto F(A \cup B) - F(A) \) is non-increasing for all \( K \)

i.e. \( F(A \cup B) - F(A) \leq F(B) - F(B \cap A) \)

"Diminishing return"

\[ F(A) = g(1A) \] if \( g \) is concave

\[ \text{concavity} \] then \( F \) is submodular

\[ F \text{ submodular } \Leftrightarrow \text{Grausse extension } f \text{ is convex} \]
can write $f(w) = \max_{s \in \mathcal{B}(F)} \langle s, w \rangle$

$\min_{A \in V} \mathbb{E} \left[ \max_{s \in \mathcal{B}(F)} \left( \frac{\langle s, w \rangle}{f(s)} \right) \right] \rightarrow \text{use projected subgradient method}$

$\ast$ with $L_2$ regularization, use duality to get smooth problem

$\min_{s \in \mathcal{B}(F)} \frac{1}{2} \| s \|^2$

$
\rightarrow \text{use } \text{min-norm point algorithm, variant of FW algorithm.}
$

pointers:

- encoder-decoder model (seq2seq):
  - see chapter 10.4 of deep learning book
  - attention mechanism: 12.4.5.1 of deep learning book
  - gated RNNs (LSTM or GRU) - see 10.10

- learning to search
  - see great ICML 2015 tutorial

- submodularity:
  - website with tutorials and pointers: http://submodularity.org/

aside: mentioned the counter-example by Nesterov that showed that Frank-Wolfe can sometimes not converge if the function is not differentiable;
  - see Example 1 in: Yurii Nesterov, "Complexity bounds for primal-dual methods minimizing the model of objective function", 2016