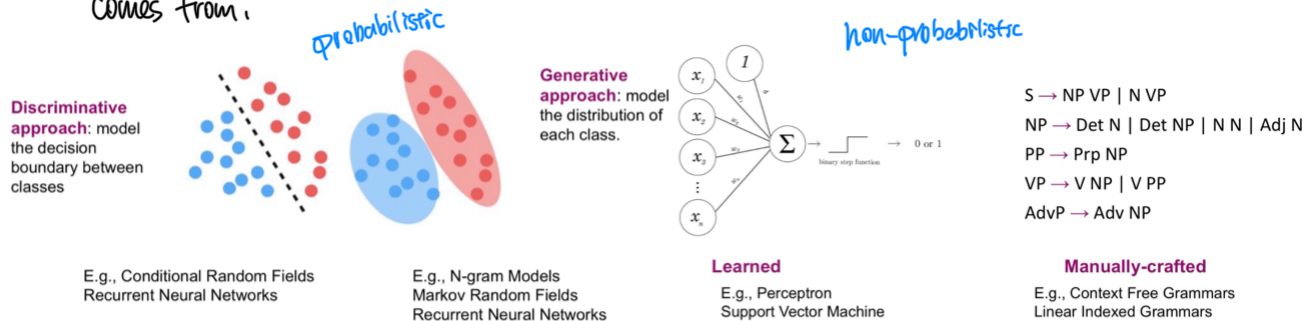


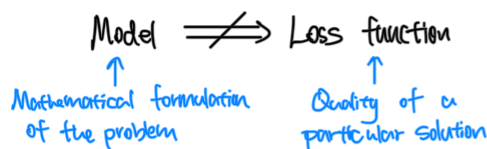
11. Axes of Modeling

Modeling

- Probabilistic models: represent conditional distributions over Y . To train a classifier, one uses dataset to compute $p(y|x)$, for $x, y \in X \times Y$. A classification rule based on these conditional probabilities is then used to choose a label for y .
- Non-probabilistic models: operate by learning rules to separate the feature space. The model then returns the class associated with the space where it believes a sample comes from.



Loss functions and evaluation metrics



- Loss function can be any function of the model parameters that quantifies the (mis)fit to our training data.

- For a fixed dataset $D = \{(x_i, y_i)\}$, consisting of n training pairs, the loss function is a function $L: \theta \rightarrow \mathbb{R}$.
- Usually, a loss function is defined via an intermediate function $l: Y \times Y \rightarrow \mathbb{R}$, where L is then defined as $L(\theta) = \frac{1}{n} \sum_{i=1}^n l(y_i, f_\theta(x_i))$.
- Some desired properties for loss functions
 - Convexity: ensures convergence guarantees by iterative methods.
 - Differentiability: facilitates the mathematical analysis.
 - Robustness: the function should lead to models that are not sensitive to the noise in the dataset.
 - Tractability: one should be able to minimize this function with a computationally efficient procedure.
 - Reliability: the function should lead to meaningful models that match reality.

Maximum Likelihood Estimation

$$L(\theta) = - \sum_{i \in n} \log p(y_i | x_i; \theta) \quad \text{* negative log-likelihood}$$

- Minimizer of this loss is a maximum likelihood estimator (MLE).

- Such estimators have attractive properties:

- Consistency: Let $F = \{p_\theta \mid \theta \in \Theta\}$ be a parametric family of distributions. Assume that D is drawn from p_{θ^*} . Then the MLE $\hat{\theta}$ converges in probability to θ^* as $n \rightarrow \infty$.
- Asymptotic efficiency: The MLE minimizes $E[(\theta - \theta^*)^2]$ as $n \rightarrow \infty$.
- Lowest KL-divergence with respect to the distribution: Even if D is a sample from a distribution $\tilde{p}$ outside the model family F , the MLE yields the distribution in F that minimizes the KL-divergence with respect to $\tilde{p}$.

* MLE can only be computed for probabilistic models. If N is not sufficiently large, they show high variance and risk overfitting.

Other losses

- Hinge loss: $l(y, \hat{y}) = \max\{0, 1 - y\hat{y}\}$
- Logistic loss: $l(y, \hat{y}) = \log(1 + e^{-y\hat{y}})$

Regularization

Given a learning algorithm $A(D)$ that takes as input a dataset and outputs a parameter in θ . The generalization error of a learning algorithm A is given by $E_{x,y,D}[l(Y, f_{A(D)}(x))]$. Overfitting occurs when there is an unacceptable difference between the generalization error and the empirical loss:

$$\frac{1}{n} \sum_{i=1}^n l(y_i, f_{A(D)}(x_i)).$$

Weight penalties (Lasso & ridge)

- The regularized loss functions are:

$$\mathcal{L}_{l_1}(\theta) = \mathcal{L}(\theta) + \lambda \|\theta\|_1 \quad \mathcal{L}_{l_2}(\theta) = \mathcal{L}(\theta) + \lambda \|\theta\|_2^2$$

- λ is the **regularization parameter** that determines how severely large values of the parameters will be punished
- They result in estimates with different properties:
 - l_1 loss encourages many coefficients to be exactly zero
 - l_2 loss shrinks the parameters to small non-zero values

The regularization strength depends on λ



Model Evaluation

Evaluation metric $\neq$ Loss function

- Loss function is a function of the model parameters that we directly optimize.
- Evaluation metric may include any aspect of the model we want to evaluate.

Some common evaluation metrics

- Confusion matrix
- Curve scores

- Precision } trade-off
- Recall
- Accuracy (and its derivatives)

Can readily be extended to a higher number of classes

		True condition	
		Total population	Condition negative
Predicted condition	Condition positive	True positive	False positive
	Condition negative	False negative	True negative

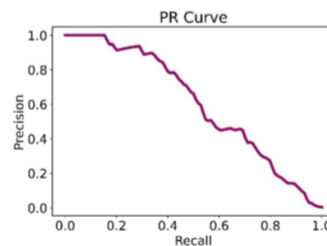
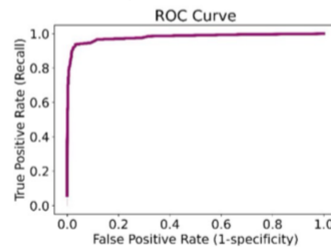
$$\text{Precision} = \frac{\#TP}{\#PP}$$

$$\text{Recall} = \frac{\#TP}{\#CP}$$

$$\text{Accuracy} = \frac{\#TP + \#TN}{N}$$

TP = true positive, PP = predicted positive
CP = condition positive, TN = true negative

- Area under the ROC curve
- Area under the Precision-Recall curve
- Often offers a better overview of the performance over multiple decision thresholds.



- F_β score

$$F_\beta = (1 + \beta^2) \frac{\text{precision} \cdot \text{recall}}{\beta^2 \text{precision} + \text{recall}}$$

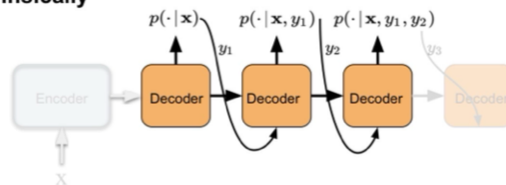
takes account both the precision and recall.

$$F_1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}$$

harmonic mean between precision and recall.

Evaluating Language Models: Intrinsic vs. Extrinsic validation

- Language models are usually *not an end application in itself*
 - They are used as components of larger systems (e.g. translation), and would thus be best evaluated **extrinsically**



- Does the model improve the performance on the final task?
- Hard to estimate reliably, since the final performance depends on many (non language-modelling) details
- We thus resort to **intrinsic evaluation**, which is task-neutral and thus more general, but can result in models that overfit to artificial metrics
 - We evaluate the quality of the model based on some criteria we think are appropriate

Evaluating Language Models

- Two intrinsic metrics can easily be computed: **held-out (log) likelihood** and the derived **perplexity**
- Since language models are probabilistic, we can always compute the likelihood of a held-out set that the model was not trained on:

$$l(\mathbf{w}) = \sum_{m=1}^M \log p(w_m | w_{m-1}, \dots, w_1)$$

- A more unified way to report it is **perplexity**:

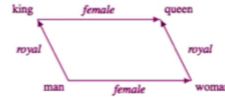
$$\text{Perplex}(\mathbf{w}) = 2^{-\frac{l(\mathbf{w})}{M}}$$

- Lower perplexity $\Leftrightarrow$ higher likelihood
- Perfect model: $\text{Perplex}(\mathbf{w}) = 1$

- Extremely bad model: $\text{Perplex}(\mathbf{w}) \rightarrow \infty$

Evaluating Word Embeddings

- Just as language models, we can evaluate embeddings intrinsically and extrinsically
- **Intrinsic metrics:**
 - Correlation between the **cosine similarity** of word embeddings and human judgements of similarity
 - We use existing datasets which include pairs of words and their similarities as determined by human judges and measure how correlated they are with the cosine similarities of the given pair of words
 - Testing for **word analogies**
 - The model is provided with the first three parts of the analogy ($i_1 : j_1 :: i_2 : ?$), and the final element is predicted by finding the word embedding most similar to $\mathbf{u}_{i_1} - \mathbf{u}_{j_1} + \mathbf{u}_{i_2}$



Model Selection

* Not that important, skipped. Please see the slides.