In this lecture

- Perceptron
- Logistic regression
- Support Vector Machines
Perceptron

The simplest neural network: one layer binary classifier, with step function

\[ y = \theta \left( \sum_{\alpha} w_{\alpha} x_{\alpha} \right) \]

We update weights with \( w_{\alpha} \leftarrow w_{\alpha} + \eta \sum_{i} (\hat{y}_i - y_i) x_{\alpha}^i \)

(we cannot backpropagate through the step function)
Logistic regression

Regression for a binary dependent variable $y=0, 1$

We model with a sigmoid or logistic function:

$$
\sigma(t) = \frac{1}{1 + e^{-t}}
$$

Where $t$ are the log-odds, which we take as linear combination of inputs.

Cost function (log likelihood): 

$$
C(w) = - \sum_{i} \log[\sigma(wx_i)^{y_i} (1 - \sigma(wx_i))^{1-y_i}]
$$

then optimize weights with backpropagation and SGD or similar...

Can be used as a classification model.
Support Vector Machines (SVM)

Binary classifier: we want to find the “maximum margin plane”

\[ y_i (\vec{w} \cdot \vec{x}_i - b) \geq 1 \]

while maximizing the gap: minimizing \( |\vec{w}| \).

If not linearly classifiable, we want to minimize

\[
\sum_i \max(0, 1 - y_i (\vec{w} \cdot \vec{x}_i - b)) + \lambda |\vec{w}|^2
\]

Or we can try to apply the kernel trick...
Support Vector Machine (with kernel)

We want to derive a form only involving the scalar product between examples:

\[
\vec{w} = \sum_i \alpha_i y_i \vec{x}_i \\
\max \sum_i \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j
\]

So we can just compute \( k(x,x') \) (Gaussian kernel or similar)
That’s all for now...