Machine Learning

Homework 6

Not collected, not graded.

Some theory (review from Math 164)

Consider the following **minimization** problem:

$$\min_{\mathbf{x}} f(\mathbf{x}) \qquad \text{s.t.} \qquad \begin{cases} g_i(\mathbf{x}) \leq 0 \\ h_j(\mathbf{x}) = 0 \end{cases}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the optimization variable, f is the objective or cost function, g_i, $i = 1, \ldots, m$ are the inequality constraint functions, and h_j, $j = 1, \ldots, l$ are the equality constraint functions. (Mind the $\leq$!)
We can write the corresponding Lagrangian function:

$$L(\mathbf{x}, \mu_i, \lambda_j) := f(\mathbf{x}) + \sum_i \mu_i g_i(\mathbf{x}) + \sum_j \lambda_j h_j(\mathbf{x}).$$

The saddle point of this Lagrangian with respect to all its arguments ($\min_{\mathbf{x}}$, $\max_{\lambda,\mu}$) solves the constrained optimization problem above. (Mind the signs!)
Indeed, the Karush–Kuhn–Tucker conditions are necessary conditions for optimality for $\mathbf{x}^*$ to be a local optimum that satisfies the constraints:

1. Stationarity:
$$\nabla f(\mathbf{x}^*) + \sum_i \mu_i^* \nabla g_i(\mathbf{x}^*) + \sum_j \lambda_j^* \nabla h_j(\mathbf{x}^*) = 0$$
2. Primal feasibility:
$$g_i(\mathbf{x}^*) \leq 0 \qquad \forall i$$
$$h_j(\mathbf{x}^*) = 0 \qquad \forall j$$
3. Dual feasibility:
$$\mu_i^* \geq 0 \qquad \forall i$$
4. Complementary slackness:
$$\mu_i^* g_i(\mathbf{x}^*) = 0 \qquad \forall i$$

For linear constraints g_i and h_j and convex objective, these conditions are also sufficient. Dual feasibility requires the multipliers associated with the inequality constraint to be non-negative. The complementary slackness says that these multipliers have to be zero whenever the inequality is strict; conversely, whenever the multipliers are positive (active), then the corresponding constraint is an equality constraint.

1 SVM 1.0

The primal SVM problem (with hard constraints) is

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \qquad \text{s.t.} \qquad t_n(\mathbf{w}^T \phi(\mathbf{x}_n) + b) \geq 1, \quad \forall n.$$

1. Identify this with the general minimization problem given above and write down the correct Lagrangian function (signs!).

2. State the associated KKT conditions.

3. Use these KKT conditions to eliminate the primal variables $(\mathbf{w}, b)$ from the Lagrangian function and obtain the dual optimization problem. Constraints?

4. Use the KKT conditions to eliminate $\mathbf{w}$ and b from the linear classification model $y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b$. Which KKT condition makes that the resulting sum will only have few terms?

2 SVM 2.0

The primal soft SVM problem is

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + \gamma \sum_i \xi_n \qquad \text{s.t.} \qquad \begin{cases} t_n(\mathbf{w}^T \phi(\mathbf{x}_n) + b) \geq 1 - \xi_n \\ \xi_n \geq 0 \end{cases} \quad , \quad \forall n.$$

1. Identify this with the general maximization problem given above and write down the correct Lagrangian function (signs!).

2. State the associated KKT conditions.

3. Use these KKT conditions to eliminate the primal variables $(\mathbf{w}, b, \xi)$ from the Lagrangian function and obtain the dual optimization problem. Constraints? *(Note: the duals of the two SVM problems differ only in a small detail!)*

4. Use the KKT conditions to eliminate $\mathbf{w}$ and b from the linear classification model $y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b$.

3 Perceptron

Consider a two-bit binary input vector, $\mathbf{x} \in \{-1, 1\}^2$. We call logically 'false' a value of -1, and 'true' a value of $+1$. We are interested in training a single perceptron with $w \in \mathbb{R}^3$ (to include the bias term), such that $h((\mathbf{x}, 1)^T w)$ produces the desired (binary) output, with

$$h(\cdot) := \text{sgn}(\cdot) = \begin{cases} -1 & \text{if} \quad \cdot \leq 0 \\ 1 & \text{if} \quad \cdot > 0 \end{cases}$$

the given non-linear binary activation function. Do the following operations *by hand* (read: no computer, calculator, smartphone, ...). Yes, that's possible.

1. Starting from $w_0 = (1, 0, 0)^T$, train the perceptron to produce the output $t = x_1 \wedge x_2$, i.e., the logical *AND* of the input components (i.e., $t = 1$ iff $x_1 = 1$ and $x_2 = 1$, and $t = -1$ otherwise). How many different datapoints are there, i.e., what is a good choice of $\{\mathbf{x}_n\}$? Define your own sequence of $\mathbf{x}_n = (x_1, x_2)_n$ that you present to the perceptron during stochastic gradient descent training. Stop when you think you have converged (how can you know?). Plot the datapoints and the obtained perceptron decision boundary, at convergence, and check plausibility.

2. Do the same for $t = x_1 \vee x_2$, the logical *OR*, as target output.

3. Can we do the same for $t = x_1 \otimes x_2$, the logical *XOR*? Start by plotting the datapoints and the corresponding target labels. If, then why does a single perceptron fail at this job?

4 Multilayer perceptron — Artificial neural networks

A multilayer perceptron with just a single layer of hidden nodes can be shown to achieve arbitrary precision to express any target function (see universal approximation theorem; but potentially so at the expense of a number of hidden nodes in that layer increasing exponentially with the dimension of the input data). To express the logical *XOR* function seen above, a simple single-hidden-layer network with just two hidden nodes (plus a hidden bias node) is enough.

1. Expand the logical *XOR* target function as a simple expression involving only *AND*s, *OR*s, and *NOT*s.

2. How does inversion (*NOT*) of a signal happen in a neural network, in terms of the weight connecting it to the next node?

3. Using these results, you can now combine the three perceptron-like expressions required for the *XOR*-operation into the simple two hidden nodes network. No training is required here, since we are just combining the simple learned single perceptrons of the previous exercise.

5 Hidden Markov Models: Conditional independence properties (less important)

In class, we have seen numerous conditional independence instances that were useful in establishing the recursion formula for the Forward α (first five expressions) and the Backward β (sixth expression) sweeps of the Baum-Welch algorithm. The seventh and eighth CI statement allow inferring predictive distributions for $p(\mathbf{x}_{N+1} \mid \mathbf{x}_1, \ldots, \mathbf{x}_N)$.

1. In the Hidden Markov-chains shown on the back, for each of the six properties used for the α/β-recursions, shade the respective observed variable, and circle the sets of variables that become conditionally independent as a result.

2. For each of the eight properties, give the corresponding CI statement in the form $A \perp\!\!\!\perp B \mid C$.

3. Can you develop a compact expression for the predictive distribution $p(\mathbf{x}_{N+1} \mid \mathbf{x}_1, \ldots, \mathbf{x}_N)$? You should only require the sum and product rules, as well as the conditional independence statements 7 and 8. The result will include $\alpha(z_N)_k = p(\mathbf{x}_1, \ldots, \mathbf{x}_N, z_N = k)$, transition probabilities $p(z_{N+1} = k \mid z_N = j) = A_{jk}$, and emission probabilities $p(\mathbf{x}_{N+1} \mid z_{N+1} = k)$.

   Hint: this is (13.44) in the book, but try for yourself, first, and make sure to understand each step.

6 Hidden Markov Models: Viterbi algorithm (more important)

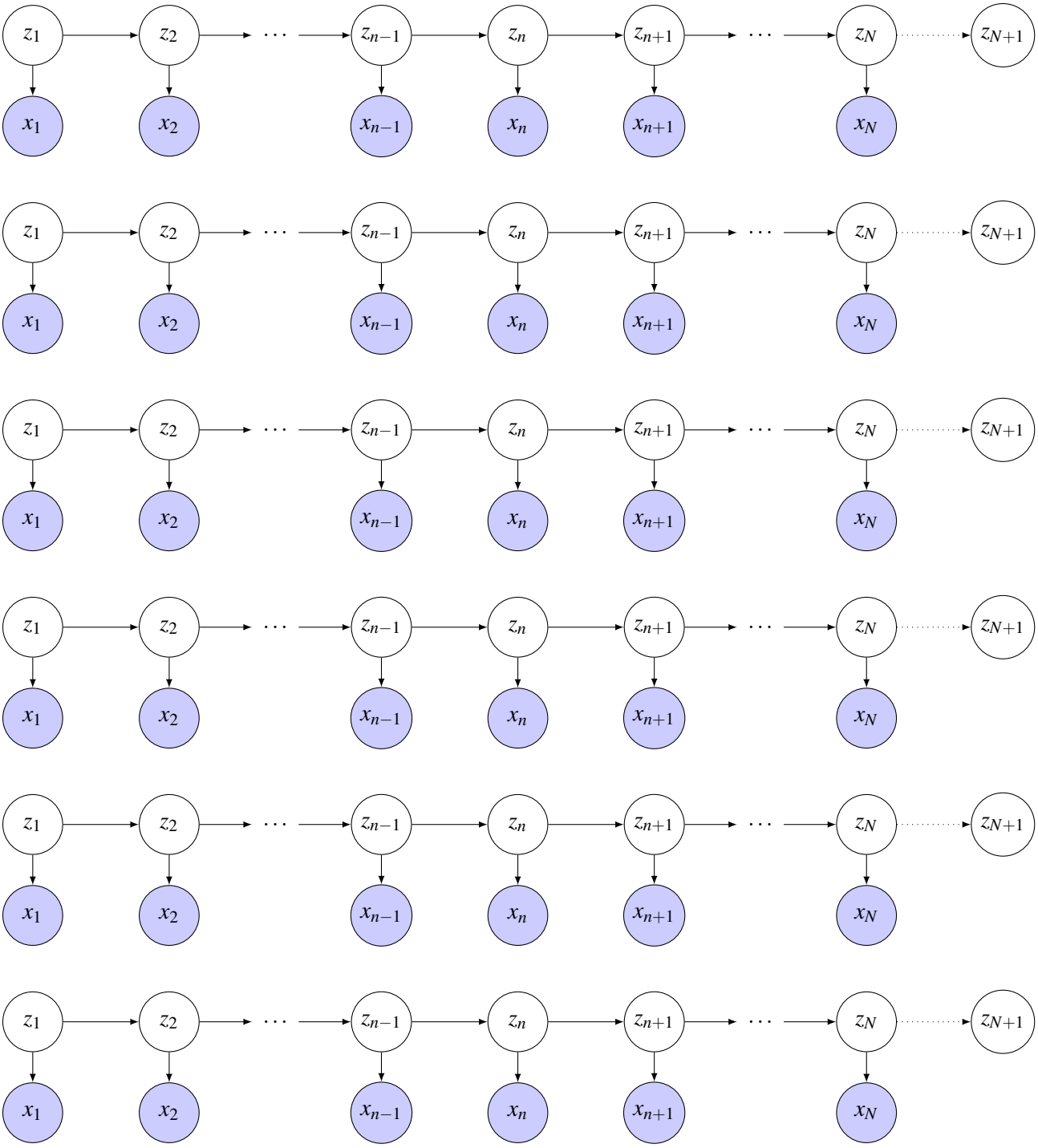
Consider an unfair casino setting, where the dishonest croupier flips a coin which is either fair or loaded. The loaded coin shows head (H) three times as often as tails (T). Starting with a 50-50 chance with either coin, the croupier switches coins on average after one in ten tosses.

1. Identify the elements of a hidden Markov model: observed and hidden variables, state space, starting state probabilities π_k, transition matrix A_{jk}, emission probabilities $p(x_n \mid z_n)$.

2. Draw the “vis. 1” state-diagram.

3. Draw the “vis. 2” trellis-diagram.

4. Given the sequence of observed tosses “HHHTHTTHHT”, infer the most likely coin-state-sequence by using the Viterbi algorithm.


z_1 z_2 $\cdots$ z_{n-1} z_n z_{n+1} $\cdots$ z_N z_{N+1}
x_1 x_2 x_{n-1} x_n x_{n+1} x_N