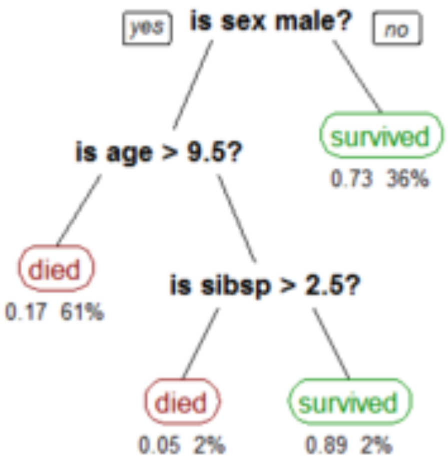
ML Workshop

Supervised Setting

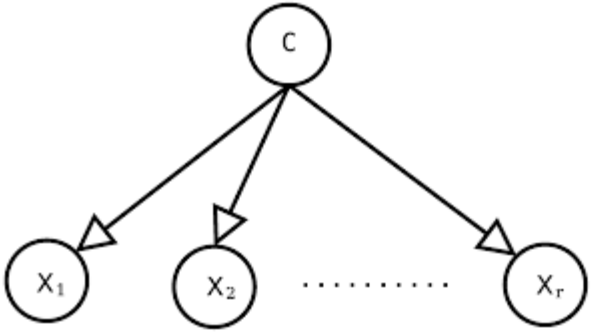
Presented by Rachel Redberg

Whirlwind Overview

Decision Trees

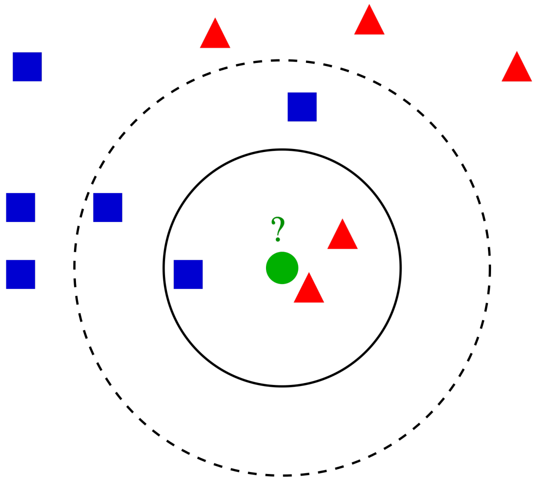


Naive Bayes



$$P(c \mid \mathbf{x}) = \frac{P(c)P(\mathbf{x} \mid c)}{P(\mathbf{x})}$$

k-NN



Supervised Learning

Given a set of labeled training examples (x_i, y_i), learn a function ***f*** mapping input to output:

$$\mathbf{f}(x_i) \approx y_i$$

Classification:

Regression:

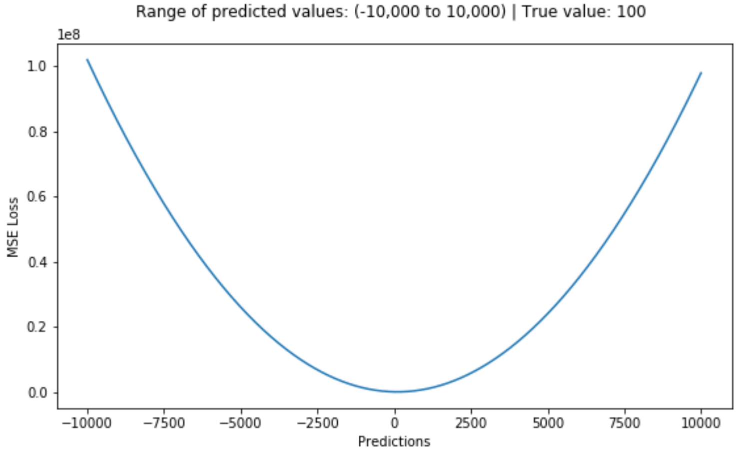
$$y_i \in \{0, 1, .., n\}$$

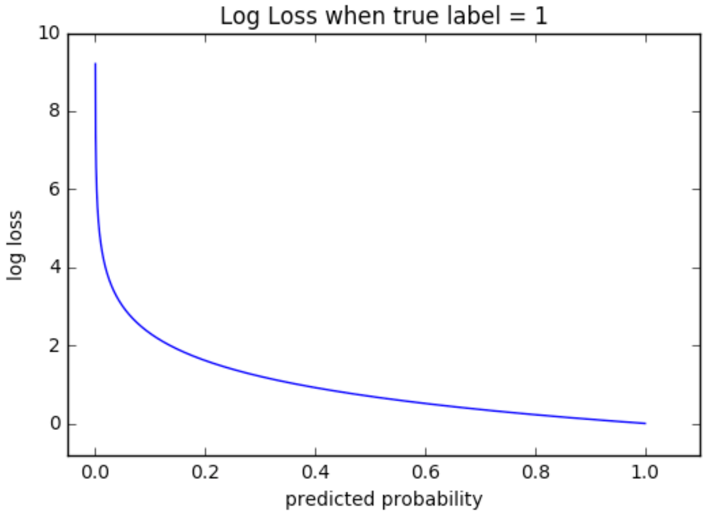
$$y_i \in \mathbf{R}$$

Oftentimes, learning a model in the supervised setting involves minimizing a **loss function** $L(y, \hat{y})$ which penalizes prediction error:

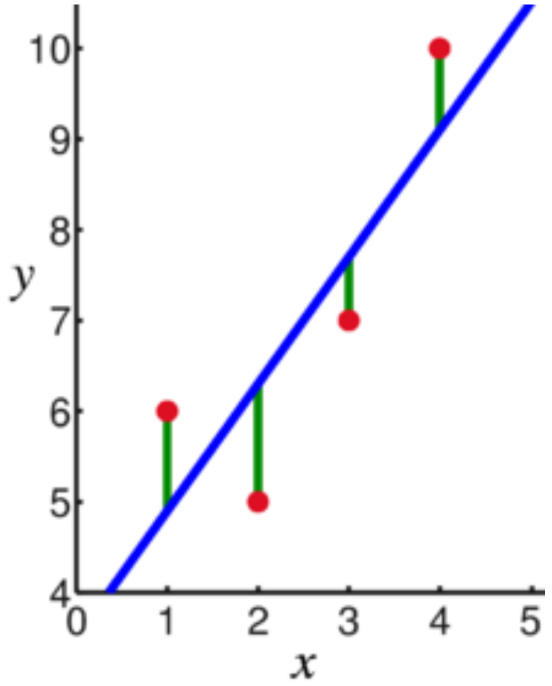
Mean Squared Error (MSE): Log loss:

$$\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \qquad -(y\log(p) + (1-y)\log(1-p))$$





Linear Regression (Least Squares)



Prediction:

$$y_i = f(x_i) = x_i^T \beta + \beta_0$$

Learning:

$$\beta = \operatorname{argmin}_b \sum_{i=1}^{n} (y_i - x_i^T b)^2$$

$$= \operatorname{argmin}_b (y - Xb)^T (y - Xb)$$

Linear Regression (Least Squares)

Closed form solution:

$$\beta = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T y$$

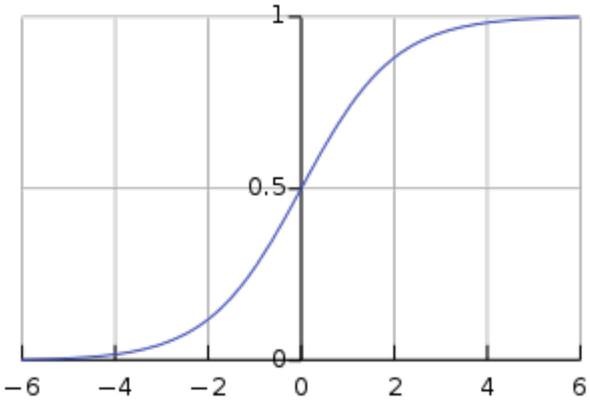
For $f(\beta) = (y - X\beta)^T (y - X\beta)$, set $\partial f / \partial \beta = 0$

and solve for β.

Logistic Regression

Models the probability of a binary independent variable as a function of multiple independent variables:

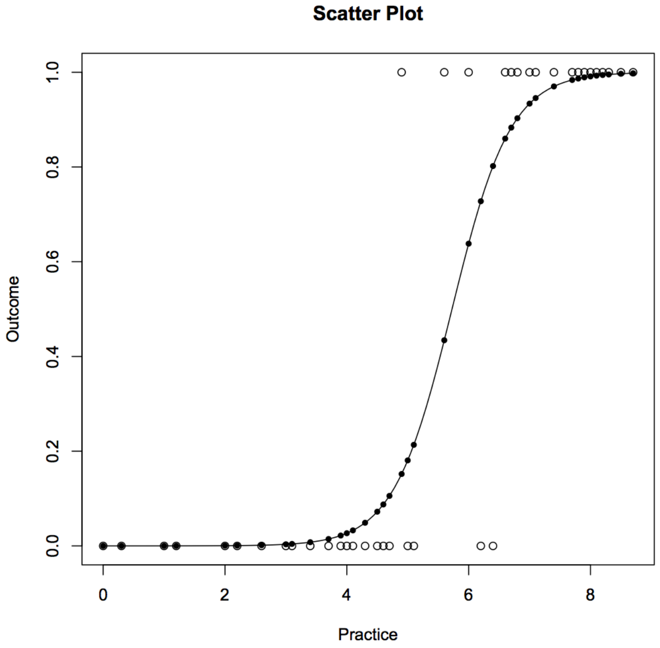
$$P(Y=1|X=x) = \frac{e^{\beta_0+\beta_1 x_1+\ldots+\beta_n x_n}}{1+e^{\beta_0+\beta_1 x_1+\ldots+\beta_n x_n}} = \sigma(x^T\beta+\beta_0)$$



$$x, \beta \in \mathbb{R}^n,\ Y \in \{0,1\},\ \sigma(x) = \frac{e^x}{1+e^x}$$

Logistic Regression

How to solve for beta? Define a likelihood function and maximize it with gradient descent. Assuming observations are independent, the likelihood is given by



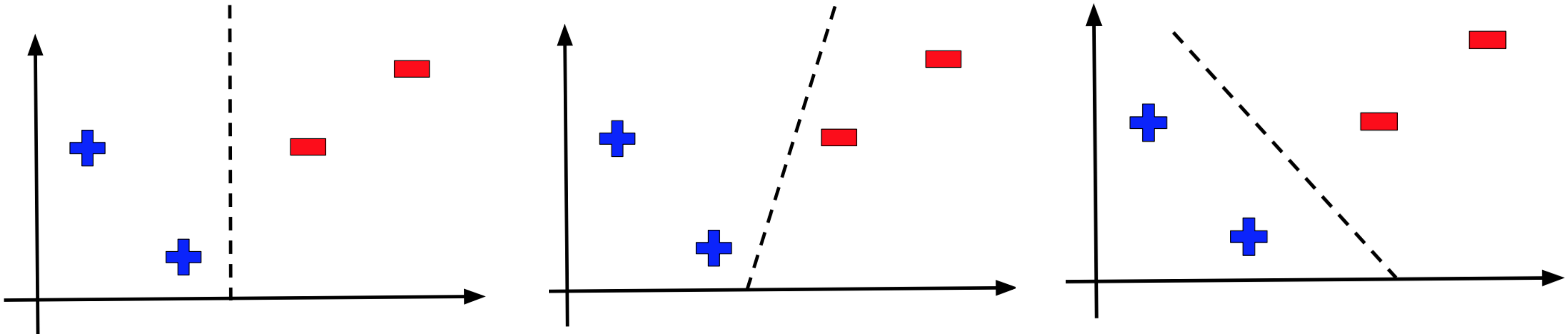
$$\mathcal{L}(\beta, \beta_o) = \prod_{i=1}^{N} \Pr(Y = y_i \mid X = x_i)$$

$$= \prod_{i=1}^{N} p(x_i)^{y_i} (1 - p(x_i))^{1-y_i}$$

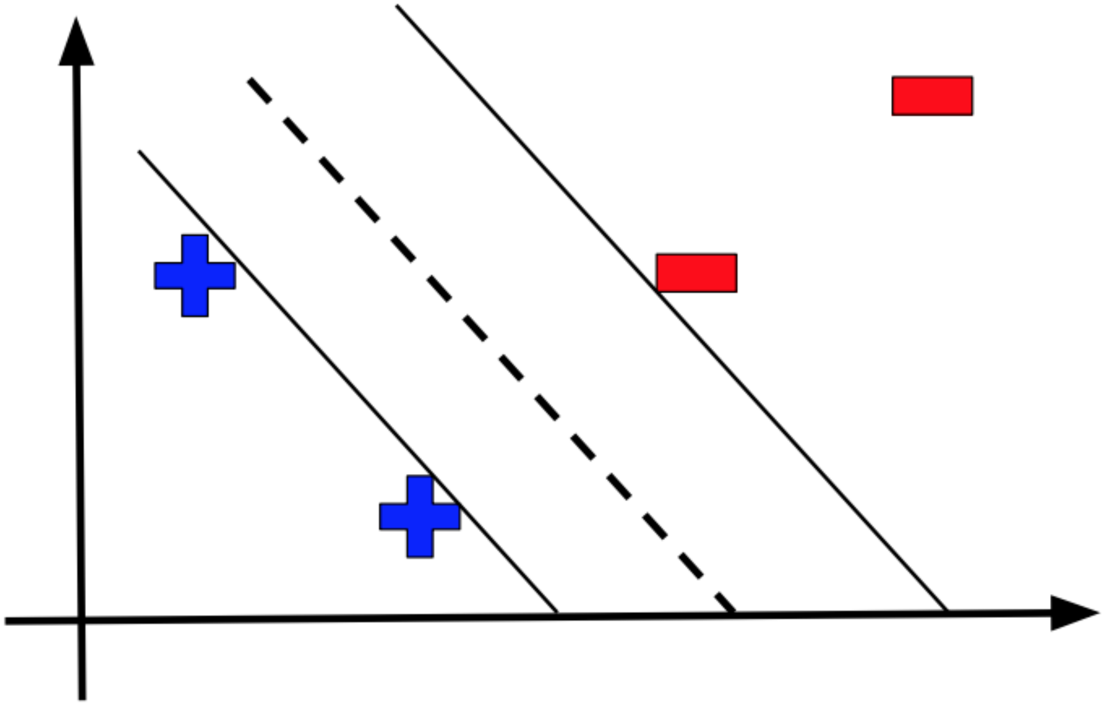
$$\text{for } p(x; \beta, \beta_0) = \frac{e^{\beta 0 + x^T \beta}}{1 + e^{\beta 0 + x^T \beta}}$$

Support Vector Machines (SVM)

Given a set of linearly separable training data, how to compute a decision boundary? Which is the best separator/hyperplane?



Widest Margin Approach (SVM)



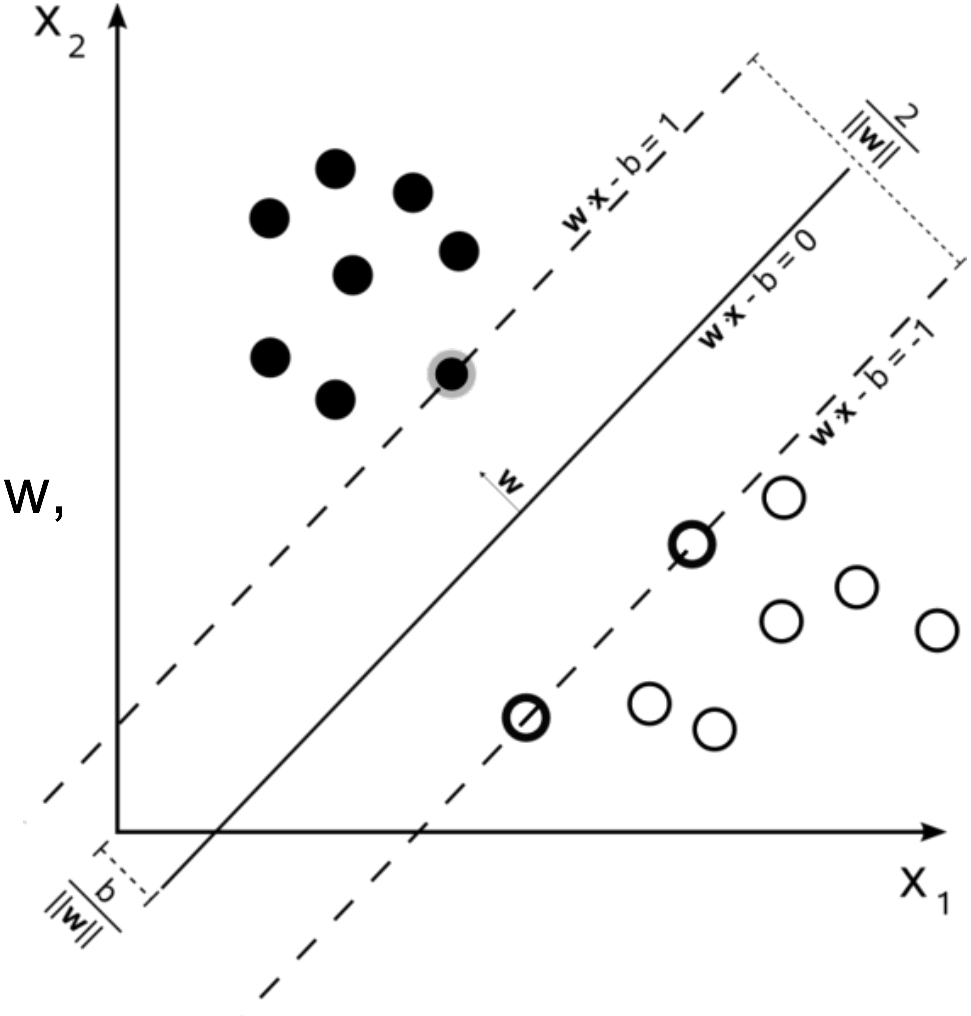
Decision Rule:

If $w^T x + b \geq 1$ ther 

If $w^T x + b \leq -1$ then  .

$w^T x$ is the projection of x onto w,
b is the bias.

Points on the margin
($w^T x + b = \pm 1$) are called
support vectors.

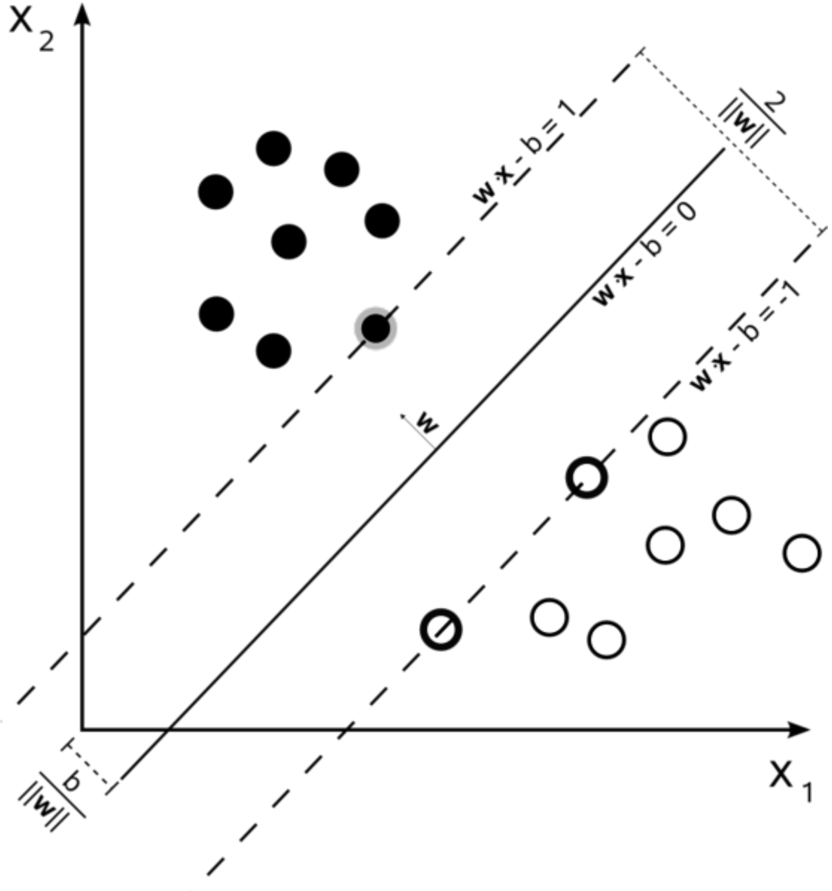


Maximize the margin

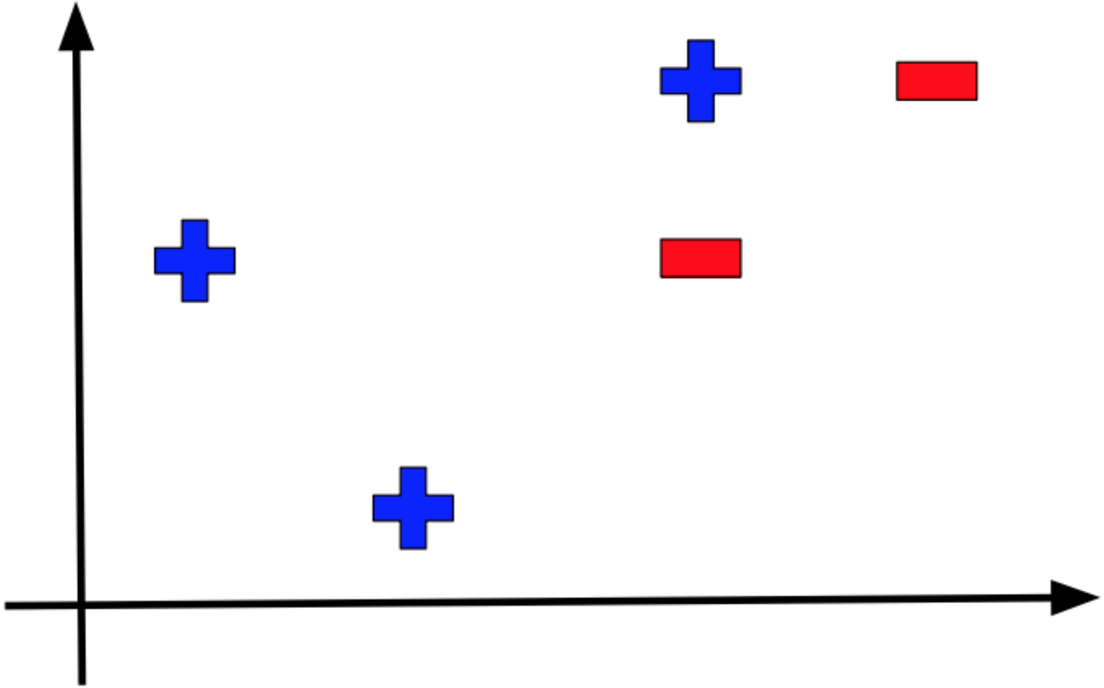
SVM :

$$\min \frac{1}{2}||w||^2$$

$$\text{s.t. } y_i(w^T x_i + b) \geq 1 \ \forall i$$



What if the data isn't linearly separable?



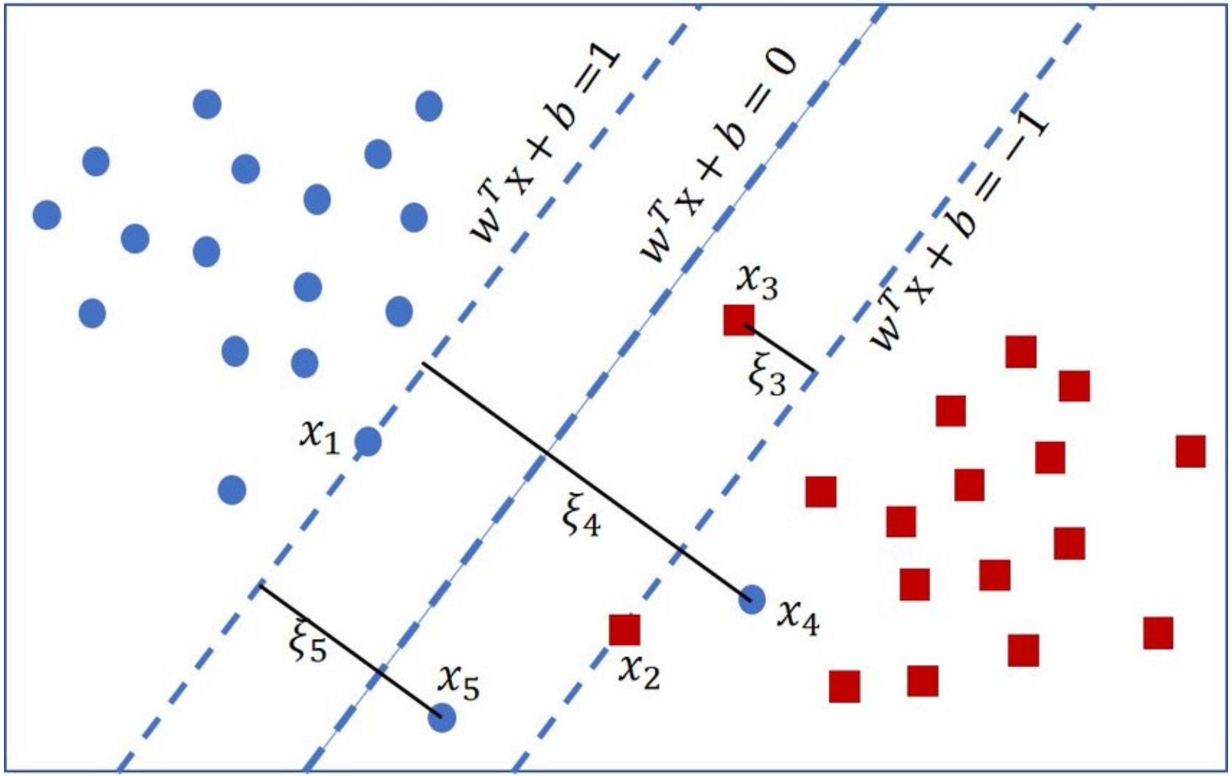
Slack variables

Soft-margin SVM :

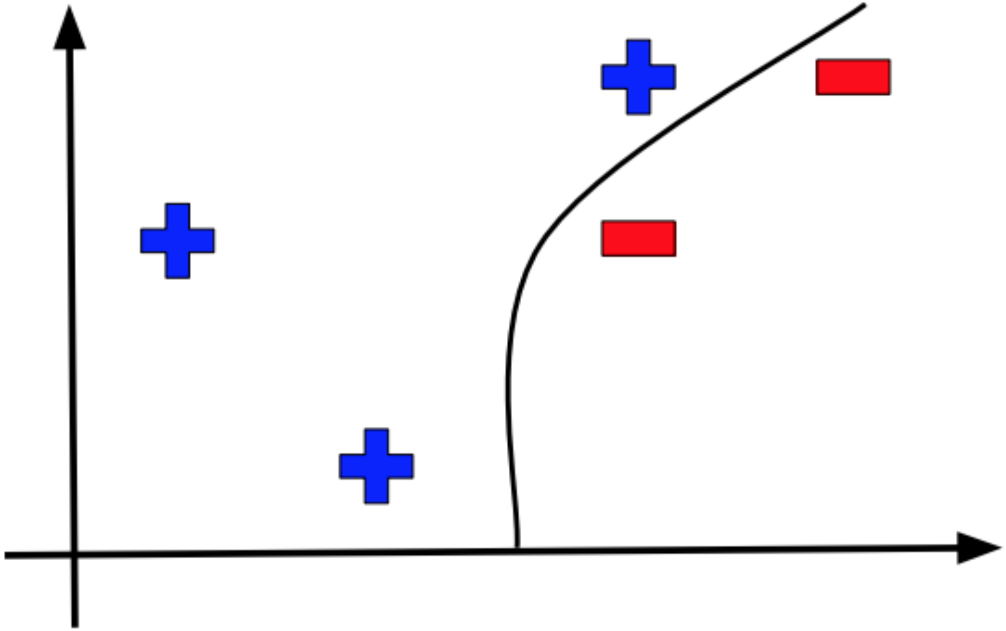
$$\min \frac{1}{2}||w||^2 + C\sum_i \xi_i$$

$$\text{s.t. } y_i(w^T x_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$



Kernel Trick



Apply a transformation ϕ :

$K(x_i, x_j) = \phi(x_i)\phi(x_j)$ is a kernel function

Primal form of SVM:

$$\begin{aligned} & \min \frac{1}{2}||w||^2 \\ st. \quad & y_i(\mathbf{w}.\mathbf{x}_i + b) \geq 1 \end{aligned}$$

Dual form of SVM:

$$\begin{aligned} & \max \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i.\mathbf{x}_j \\ st. \quad & \sum_i \alpha_i y_i = 0 \\ & \alpha_i \geq 0 \end{aligned}$$

We can apply a transformation ϕ

$$\max \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i).\phi(\mathbf{x}_j)$$

$$st. \qquad \sum_i \alpha_i y_i = 0$$

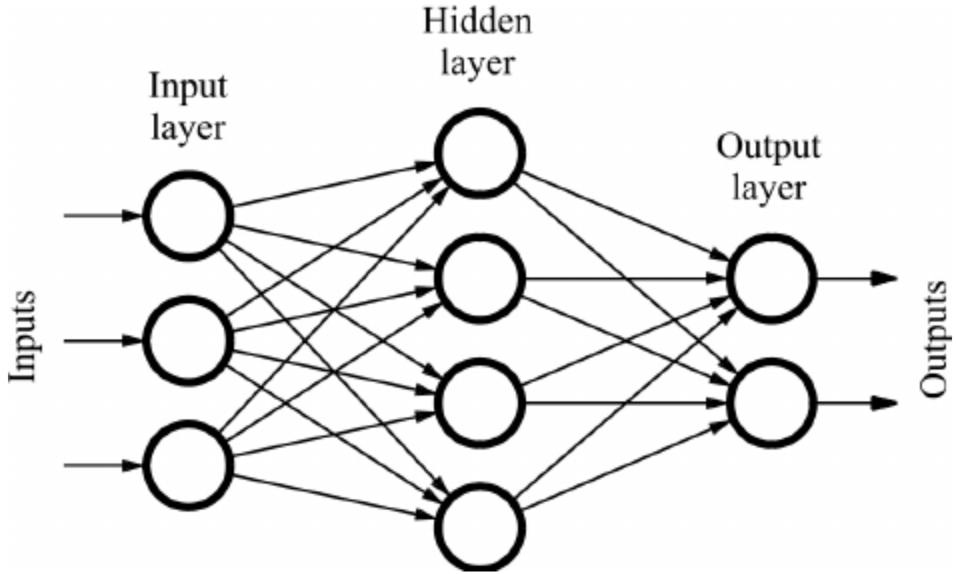
$$\alpha_i \geq 0$$

Examples of kernels:

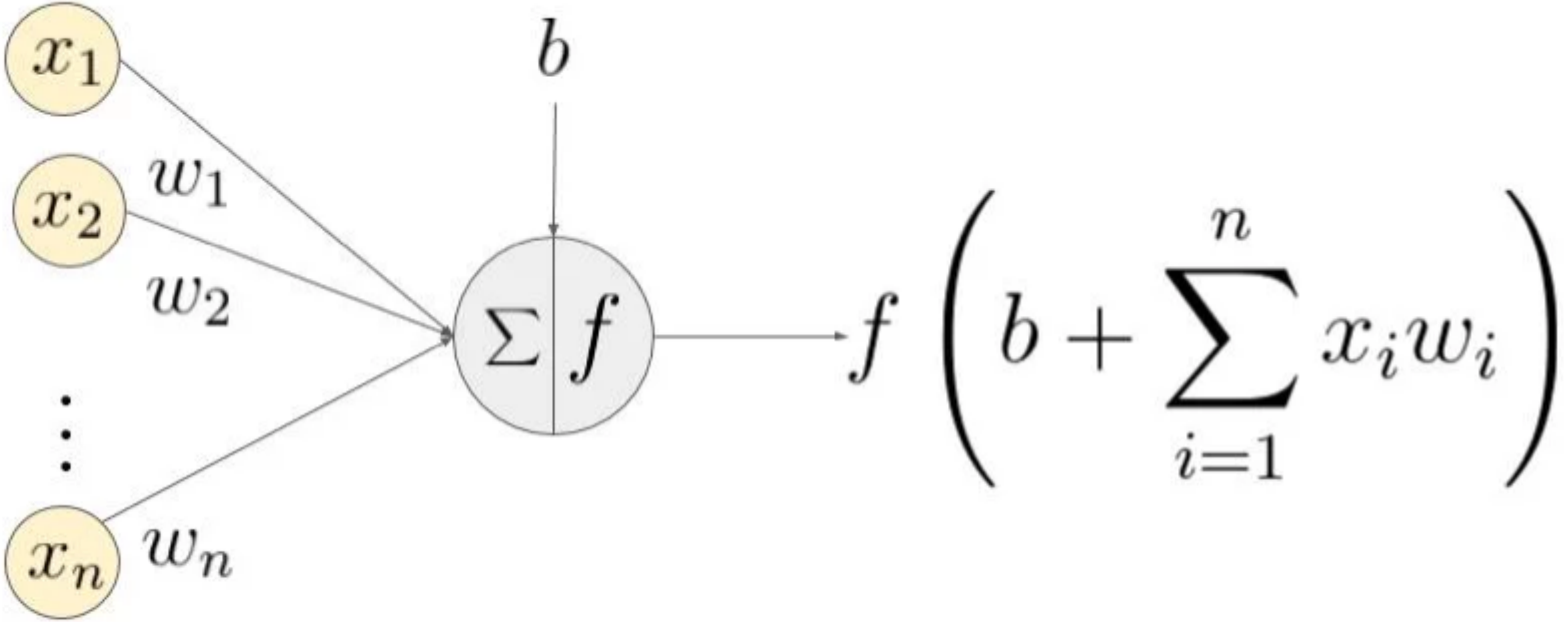
$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i.\mathbf{x}_j + 1)^n$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{||\mathbf{x}_i - \mathbf{x}_j + 1||}{\sigma}}$$

Deep Learning (Feedforward Neural Network)



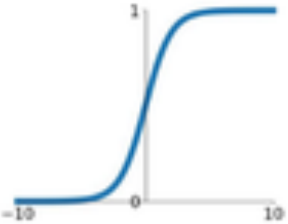
A closer look...



Activation Functions

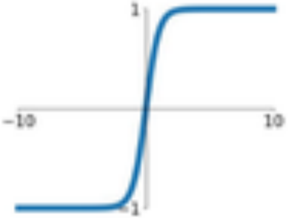
Sigmoid

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



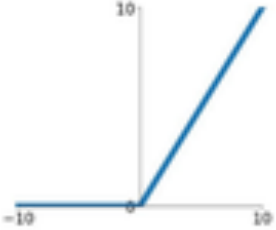
tanh

$\tanh(x)$



ReLU

$\max(0, x)$



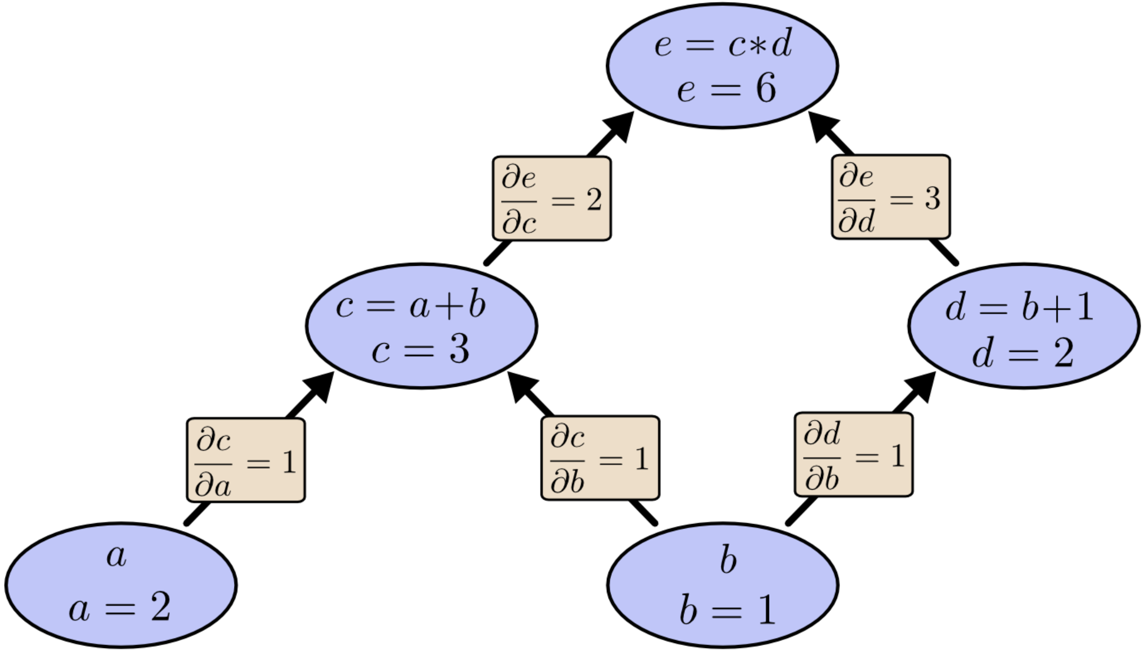
Derivatives

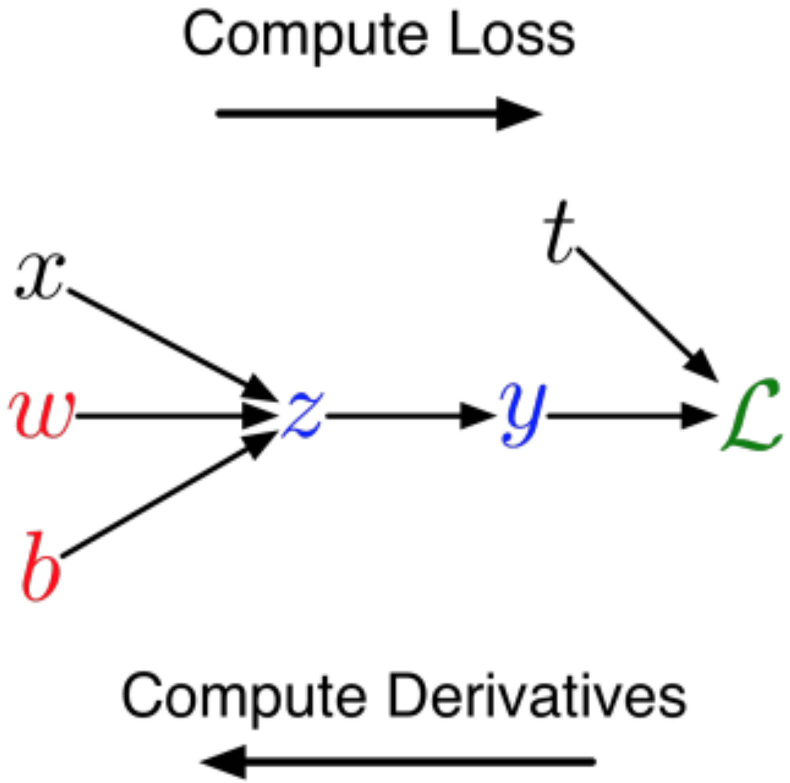
$$\sigma(x)(1 - \sigma(x))$$

$$1 - \sigma(x)^2$$

$$0 \text{ if } x \leq 0, 1 \text{ if } x > 0$$

Backpropagation (Computational Graphs)





x is the input,

w is the weight,

b is the bias,

y is the predicted output, and

t is the true value

Computing the loss:

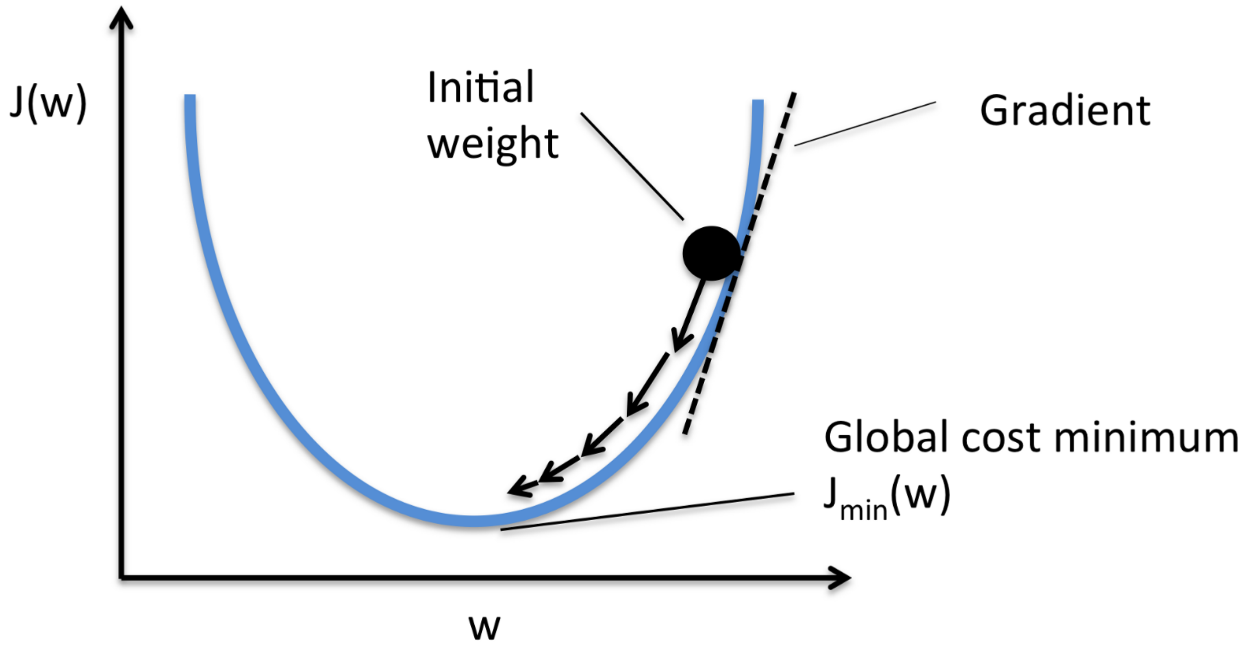
$$z = wx + b$$
$$y = \sigma(z)$$
$$\mathcal{L} = \frac{1}{2}(y - t)^2$$

Computing the derivatives:

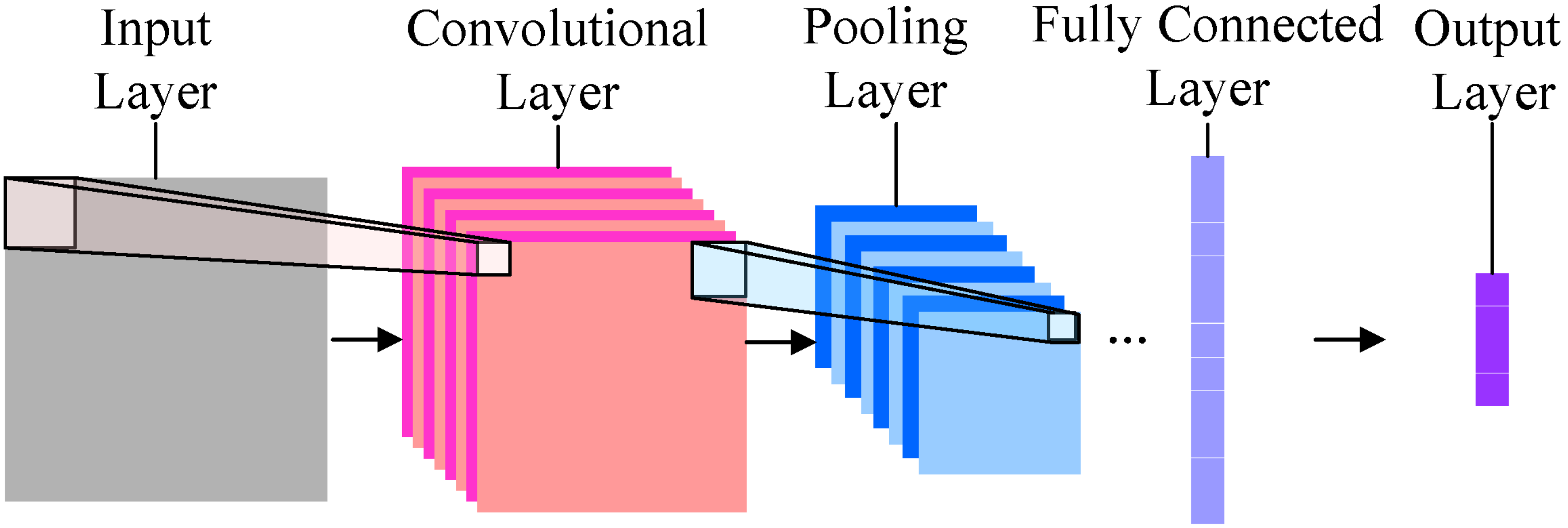
$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}y} = y - t$$
$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}z} = \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}y} \cdot \sigma'(z)$$
$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}z} \cdot x$$
$$\frac{\partial \mathcal{L}}{\partial b} = \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}z}$$

Gradient Descent

Update rule: $$w_j^{(t+1)} = w_j^{(t)} \; - \eta \frac{\partial}{\partial w_j} J(w)$$



Convolutional Neural Network



Convolutional neural networks

- Often used for image processing tasks.
- Inspired by biological processes in the visual cortex.
- Exploit statistical patterns in the data:
    - Shared weights
    - Translation invariance
- Network learns filters which extract a particular pattern (e.g. edges, textures) from the image.
- Hierarchical structure: complex patterns are built out of simpler ones.

CNN Layers

Convolutional Layer:

Convolutional kernels are matrices of a given size (e.g., 3 x 3) which are "convolved" with the image to extract features.

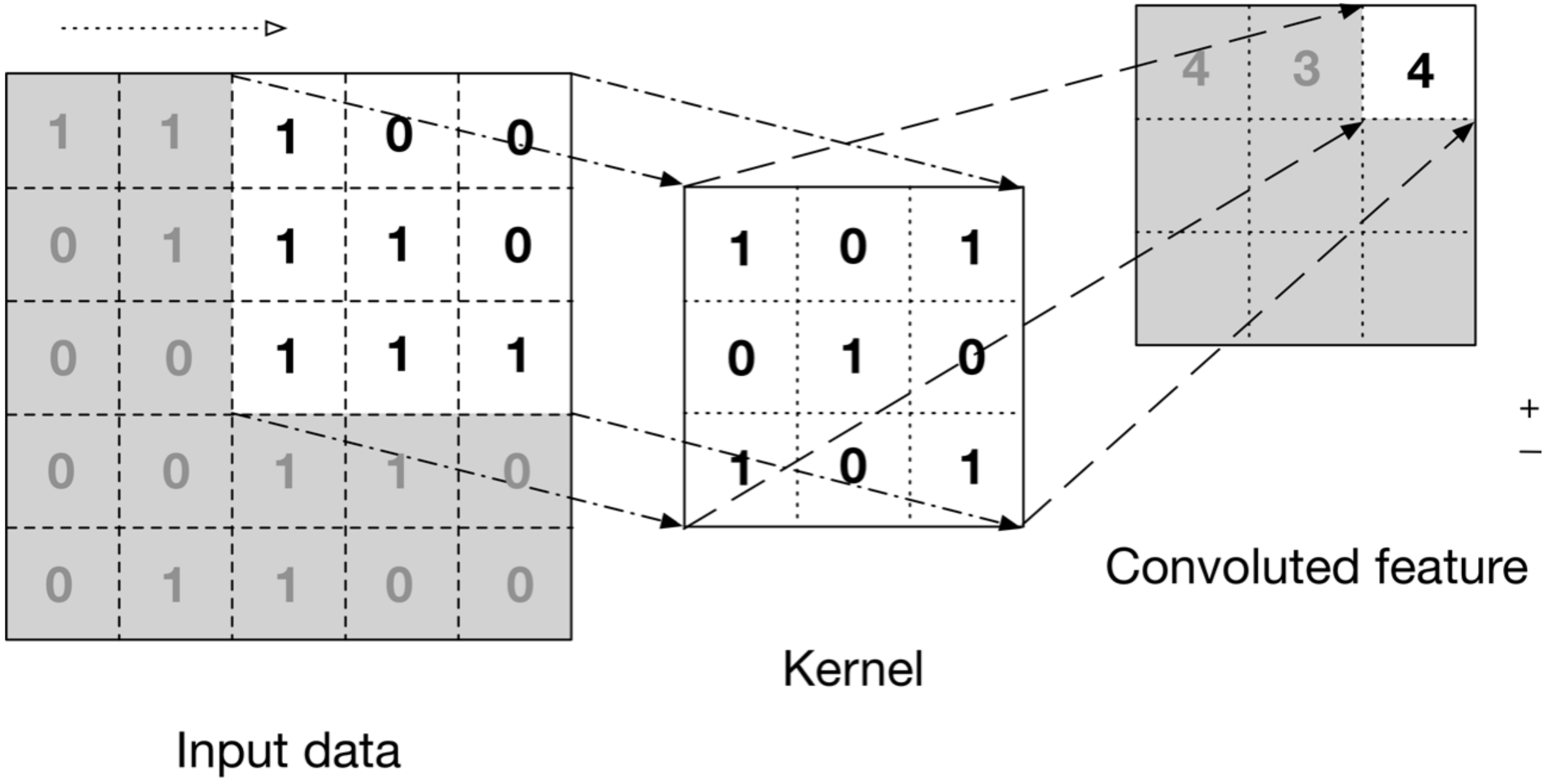
(Max) Pooling Layer:

Pooling layers reduce the dimensionality of the previous layer's output ("downsample") by combining the output of non-overlapping sub-regions of the original representation.

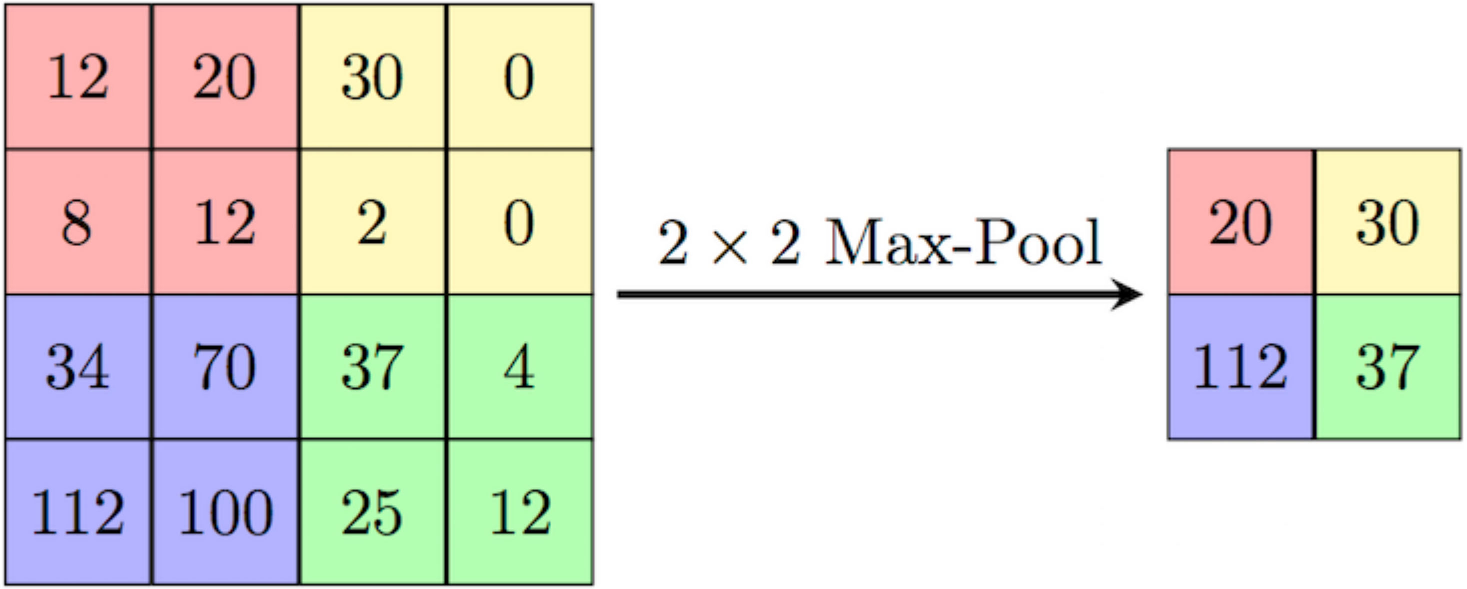
Fully Connected Layer:

Typically need to "flatten" the matrix and feed it into a fully-connected layer to aggregate output from the convolutional/pooling layers, for the final classification.

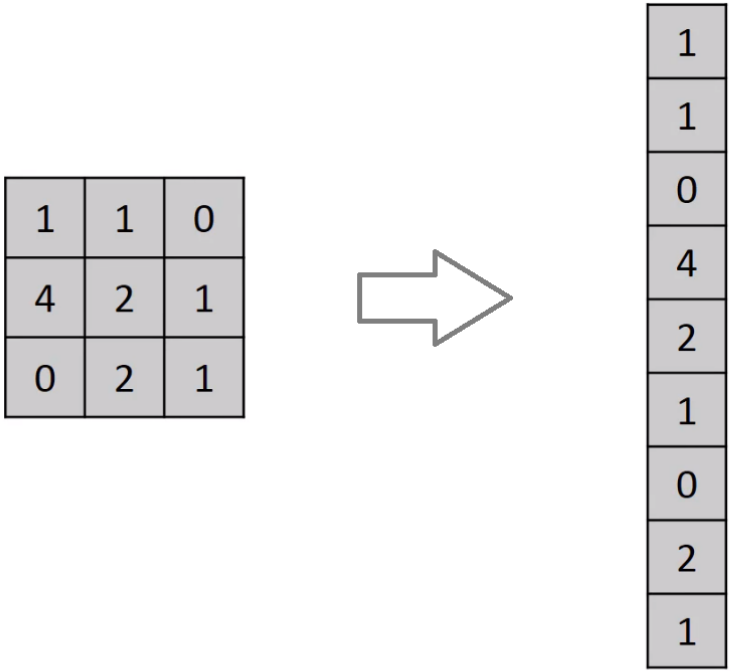
Convolutional Layer



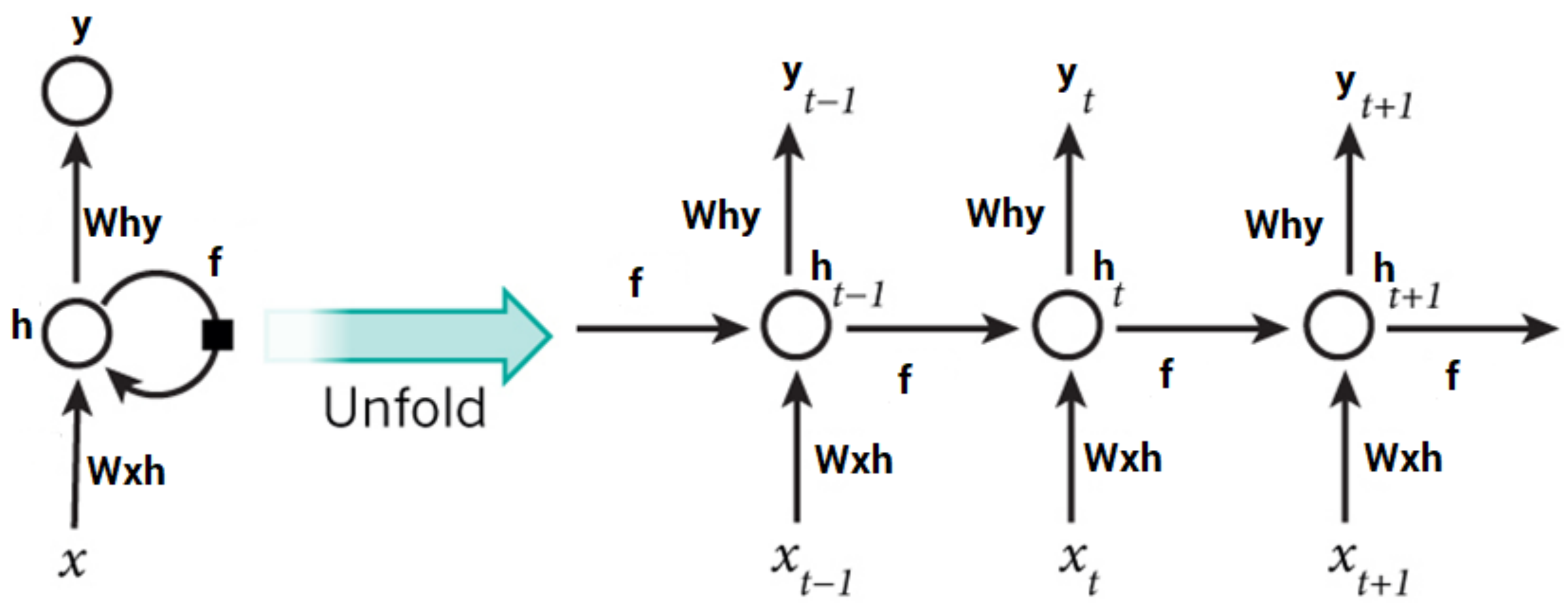
Max Pooling Layer



Flatten Matrix



Recurrent Neural Networks



Recurrent Neural Networks

- Networks with loops to “remember” information from a previous time step.
- Useful for tasks that involve sequences of data:
  - Language models (character-level, word-level), translation, speech recognition, speech-to-text, handwritten character generation, time series forecasting, etc.
- Previous information might be helpful for the present task:
  - E.g., in video captioning, use information from previous video frames to make a prediction related to the current frame.
- Issues with vanishing/exploding gradient:
  - Long short-term memory (LSTM) architecture introduces “forget” gates to learn long-term dependencies; decide which values to forget and which to update.

References

Slides are based on:

https://en.wikipedia.org/wiki/Decision_tree_learning
https://www.mdpi.com/1099-4300/19/6/247
https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/
https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0
https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html
https://en.wikipedia.org/wiki/Linear_regression
https://www.kdnuggets.com/2017/10/learn-generalized-linear-models-glm-r.html/2
https://www.researchgate.net/figure/SVM-with-soft-margin-kernel-with-different-cases-of-slack-variables_fig2_327015448
https://iq.opengenus.org/feed-forward-neural-networks/
https://www.learnopencv.com/understanding-feedforward-neural-networks/
https://sebastianraschka.com/faq/docs/relu-derivative.html
https://www.quora.com/Why-is-ReLU-the-most-common-activation-function-used-in-neural-networks
https://colah.github.io/posts/2015-08-Backprop/
http://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/slides/lec6.pdf
https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/
https://rubikscode.net/2018/02/26/introduction-to-convolutional-neural-networks/
https://www.superdatascience.com/the-ultimate-guide-to-convolutional-neural-networks-cnn/
http://www.davidsbatista.net/blog/2018/03/31/SentenceClassificationConvNets/
https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/
https://hackernoon.com/gradient-descent-aynk-7cbe95a778da
Arlei Silva’s IGERT bootcamp presentation