Bayesian Machine Learning

&

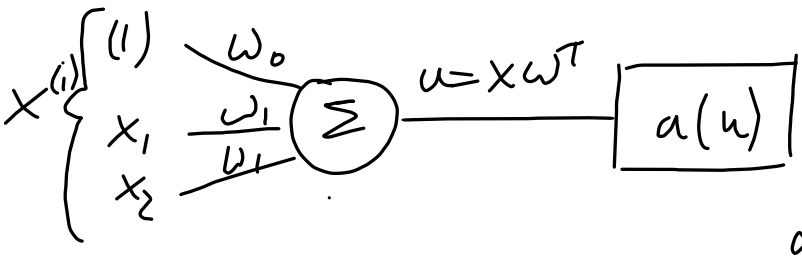
Its Application to Neural Networks

Questions

- Why Bayesian?
- Why do we need uncertainty estimates?
- What is overfitting? Where does it come from?

- What are we optimizing?
- Where does the Cost Function come from?
- How can we solve complicated Probability Distributions?

Motivating Example



$p(y|x,w) \Rightarrow 0.0, 0.1 \ldots 1.0$

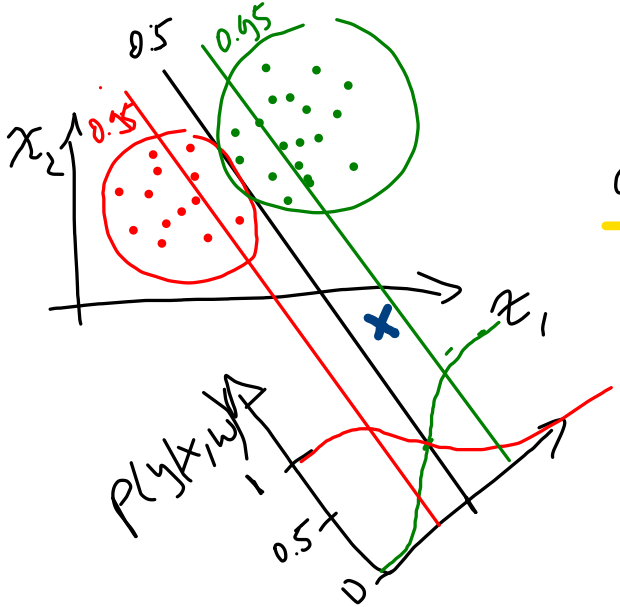
probabilities

$\alpha(w)$ - Loss Function

$\nabla_w \alpha(w)$ - Gradient

$w \leftarrow w - \alpha \nabla_w \alpha(w)$

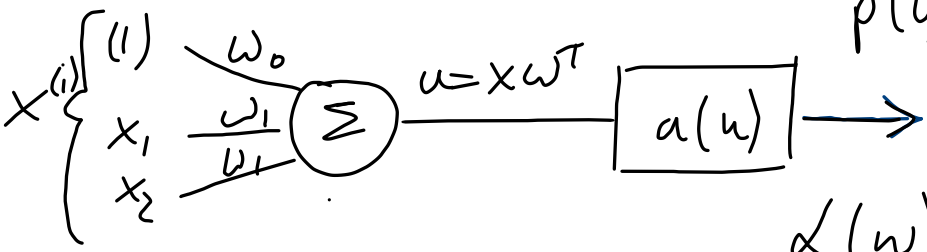
Gradient descend



Frequentist

Point Estimate

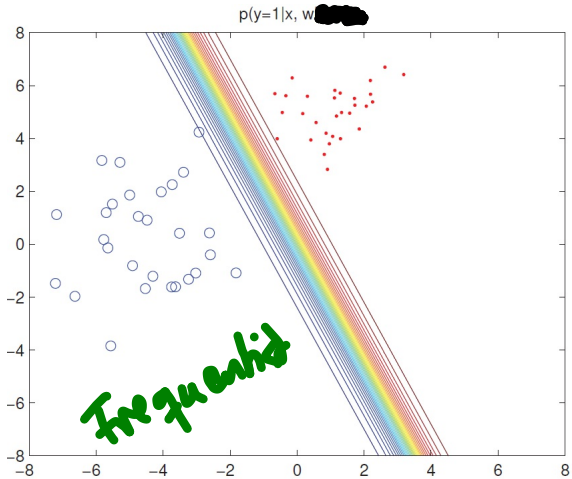
Motivating Example

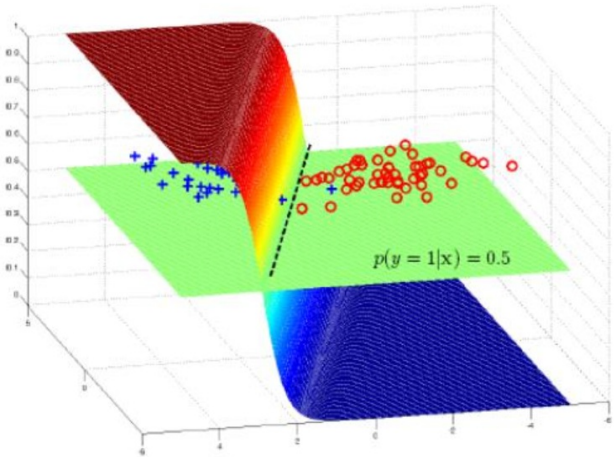


$p(y|x,w) \Rightarrow 0.0, 0.1 - 1.0$ probabilities

$\alpha(w)$ - Loss Function

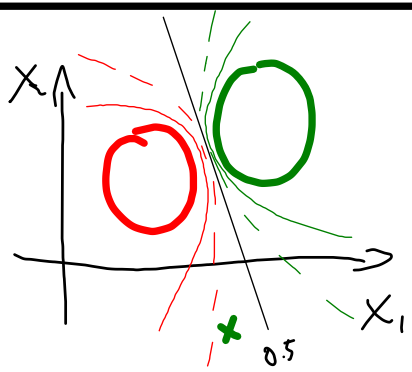
$\nabla_w \alpha(w)$ - Gradient



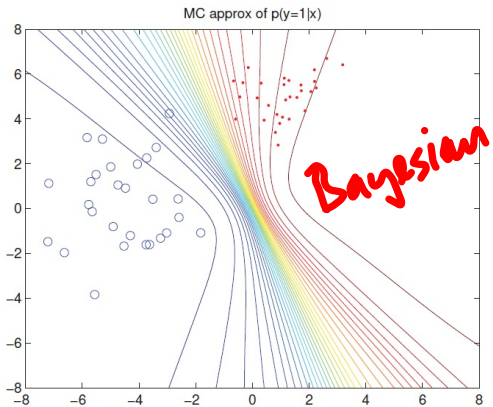


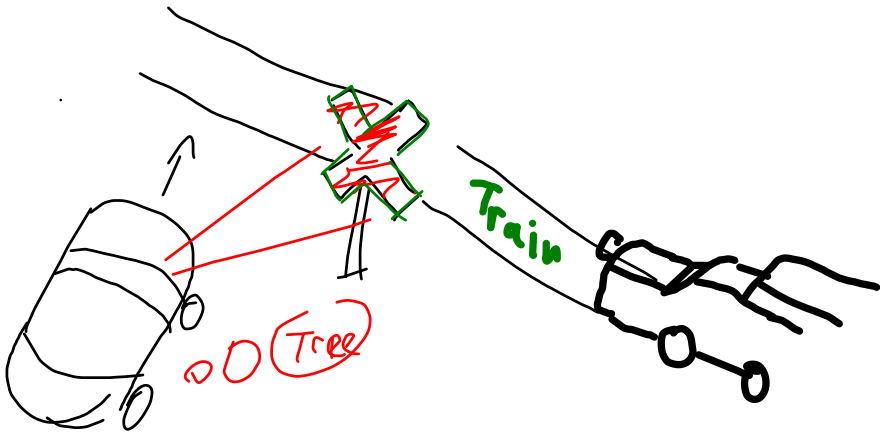
$\alpha(w)$
t descend

Uncertainty: Single Neuron → Bayesian



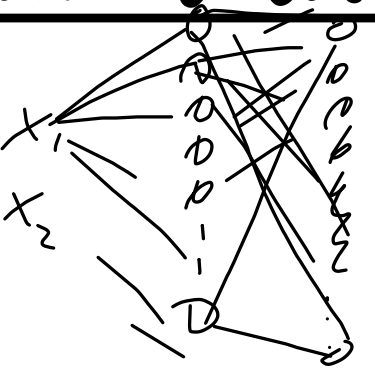
- Medical field
- Automotive

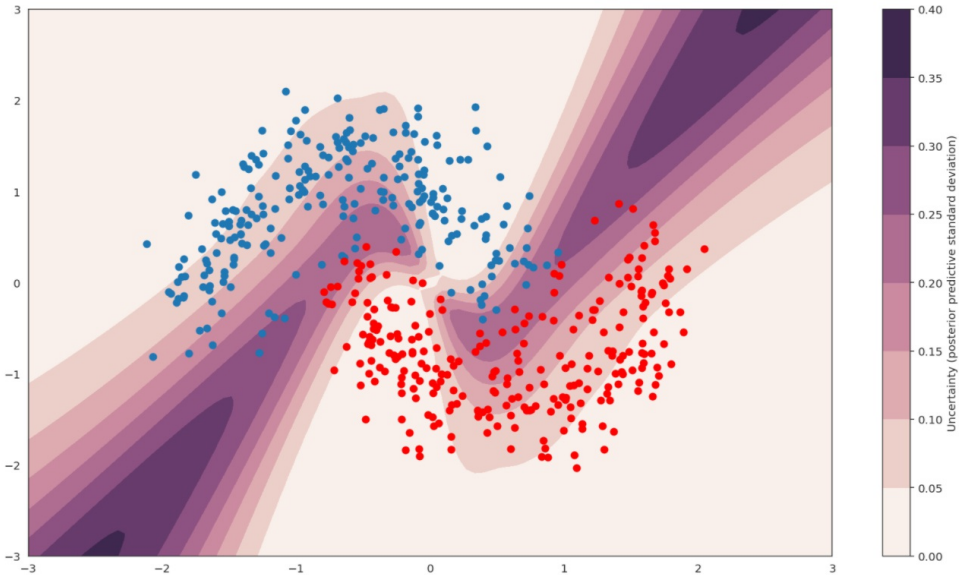




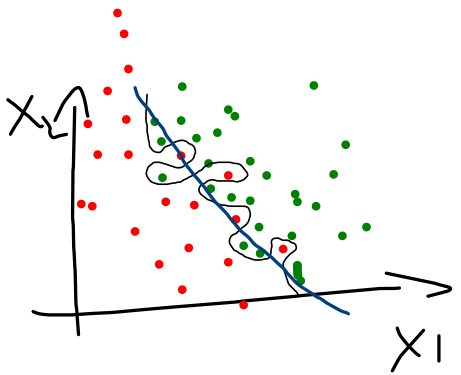
More Complex Data and Model

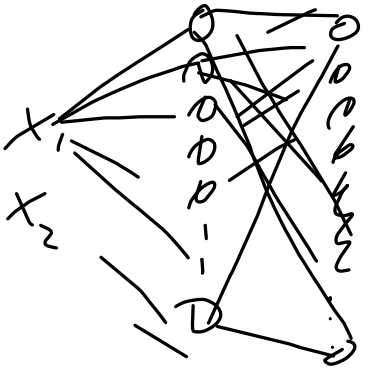
Neural Net





Overfitting



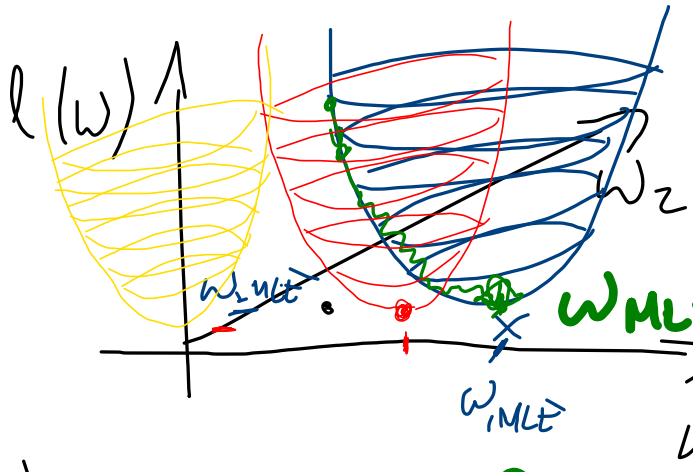


Optimization World

$$\ell_r(\omega) = \ell(\omega) + \underbrace{\alpha\, r(\omega)}_{\text{Regularizer}}$$

Bayesian Point Estimate

Point Estimates vs. Probability Distribution over Parameters (weights)



w_{MLE} → Frequentist Point Estimate

w_{MAP} → Bayesian Point Est.

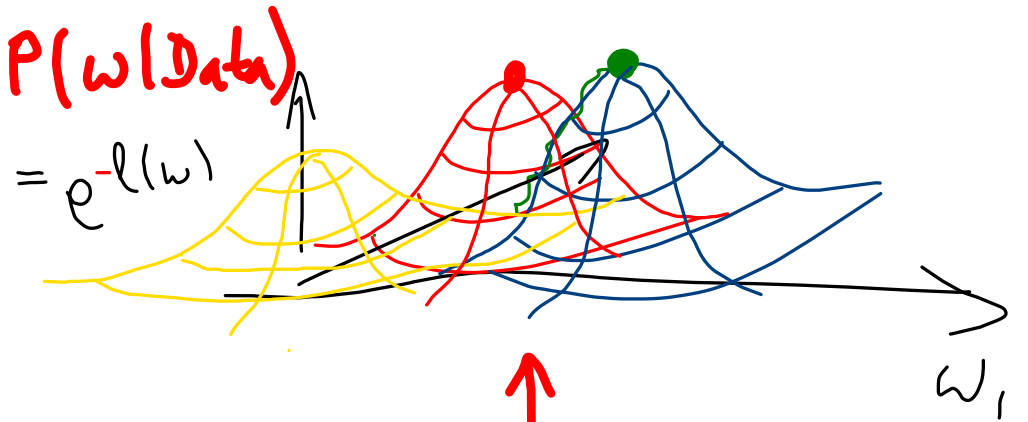
Regularizer

regularized probability

likelihood

Regularized Logistic Regression

Goal: Probability Distribution over parameters



We want the whole distribution over the weights,

instead of a point estimate

0

0

Bayesian Probability

Intuition

Biased Coin Flips with „Small Data"

bend coin

$$y = \{H, H, H, H, H, H, H, H, H, H\}$$

$n = 10$

probability of getting Heads

$$\theta_H = \frac{10}{10} = 1 //$$

Is this correct?

Can it be 100%?

Black Swan Paradox

Biased Coin Flips with „Small Data"

$y = \{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0\}$ ← [H H H H H H H H H H H T]

„Hallucinations"

probability of getting Heads

$$\theta_H = \frac{10+1}{10+2} = \frac{11}{12} = 0.833$$

Choose your „Hallucinations" according to your „PRIOR" knowledge!

Probabilistic Model of a Coin Flip

Over fitting with MLE

[H H H H H H H H H H]

$y = \{1, 1, 1, 1, 1, 1, 1, 1, 1, 1\}$

$$P(y=\{\dots\} \mid \theta) = \prod_{i=1}^{n} \text{Ber}(y^{(i)} \mid \theta) = \prod_{i=1}^{n} \theta^{y^{(i)}} \cdot (1-\theta)^{(1-y^{(i)})}$$

$$= \theta^{n_1} \cdot (1-\theta)^{n_0}$$

likelihood

$$\theta_{MLE} = \arg\max_{\theta} P(y \mid \theta)$$

Short:

$$\hat{\theta}_{MLE} = \frac{\#\ 1\text{'s}}{\#\ \text{flips}}$$

$$p(y \mid \theta = 0.3) = \prod_{i=1}^{10} 0.3^{1} = 0.3^{10}$$ ✗

$$p(y \mid \theta = 0.5) = 0.5^{10}$$ ✗

$$p(y \mid \theta = 1) = 1^{10} = 1$$ ✓ Max

Black Swan Paradox

Overfitting – Bayesian Fix

$y = \{1, 1, 1, 1, 1, 1, 1, 1, 1, 1\}$ (H H H H H H H H H H)

Hallucinations

$P(\theta \mid y)$ — Posterior

Proportionality

$\propto \prod_{i=1}^{n} \text{Ber}(y^{(i)} \mid \theta)$ • "?"

$\propto \theta^{n_1} \cdot (1-\theta)^{n_0}$ • $\theta^{\alpha-1} \cdot (1-\theta)^{\beta-1}$

$$\propto \theta^{n_1+\alpha-1} \cdot (1-\theta)^{n_0+\beta-1}$$

normalization constant $\frac{1}{z}$ is missing

$$\rightarrow \theta_{Bayes} = \frac{\#\,1\text{'s} + \alpha}{\#\,\text{flips} + \alpha + \beta}$$

$\alpha = 1, \beta = 1$

$$\theta_{Bayes} = \frac{10+1}{10+1+1} = \frac{10}{12} = 0.8\overline{3}$$

Beta - Bernoulli Model

$P(\theta|y)$ — Posterior

$$\propto \prod_{i=1}^{n} Ber(y^{(i)}|\theta) \cdot \text{"?"}$$

$$\propto \theta^{n_1} \cdot (1-\theta)^{n_2} \cdot \theta^{\alpha-1} \cdot (1-\theta)^{\beta-1}$$

$$\propto \theta^{n_1+\alpha-1} \cdot (1-\theta)^{n_0+\beta-1}$$

$$P(\theta|y) = Ber(y|\theta) \cdot Beta(\alpha, \beta)$$
$$= Beta(\alpha+n_1, \beta+n_0)$$
$$= Beta(\alpha', \beta')$$

} Conjugate Prior

Prior - Posterior

deterministic

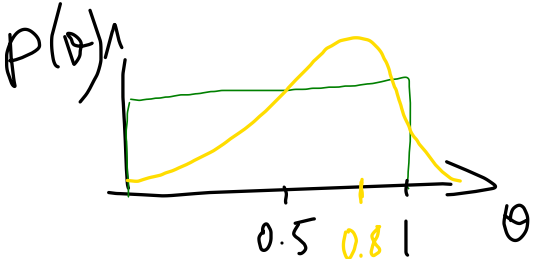
$\theta_i \rightarrow \boxed{\arg\max_\theta Ber(y|\theta)} \rightarrow \theta_{MLE}$ (dirac) $= \delta(\theta_{MLE})$

$\theta \sim p(\theta)$ → $\boxed{Ber(y|\theta)} - \boxed{\frac{1}{Z}}$ → $\theta|y \sim p(\theta|y)$

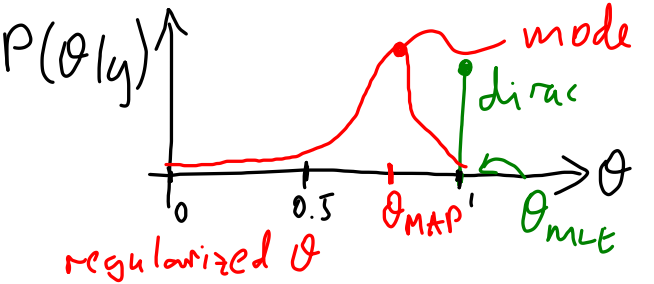
random

Prior: **Likelihood** **Norm.** **Posterior**

i.e. $Beta(\alpha, \beta)$



$Beta(\alpha+n_1, \beta+n_0)$



General Bayes Formula

(Model) likelihood prior (Hallucinations)

$$p(\theta \mid y) = \frac{p(y \mid \theta)\, p(\theta)}{p(y)}$$

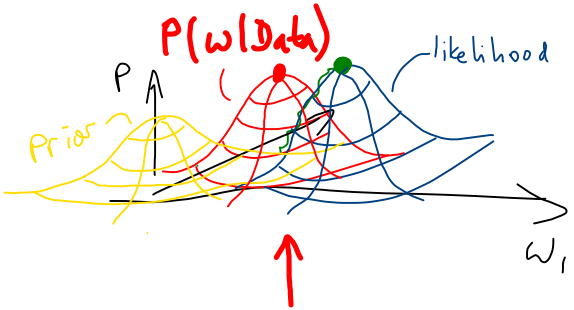
posterior over parameters (weights)

normalization constant
aka. prob. of data

→ makes it sum to one

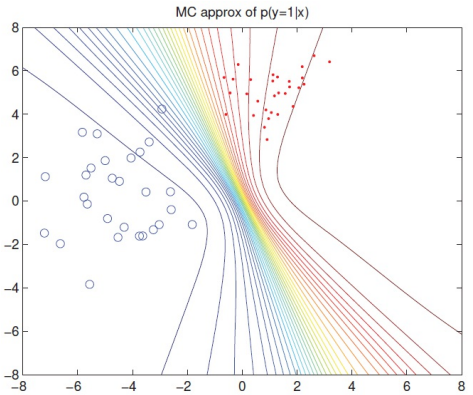
Remember?

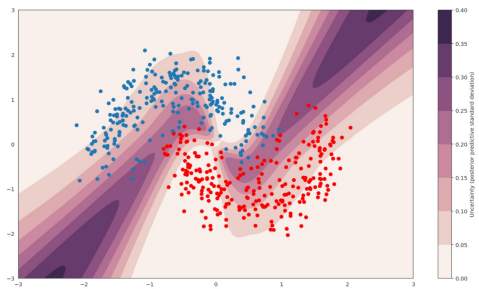
Goal: Probability Distribution over Parameters



We want the whole distribution over the weights,

instead of a point estimate





Easy Posterior vs. Nasty Posterior

Models with a Closed Form Solution aka. Conjugate Prior

Binary Data • Beta-Bernoulli

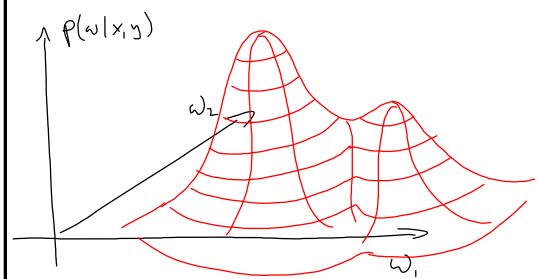
Multiclass Data • Dirichlet-Categorical

Linear Regression • Gaussian-Gaussian

⋮

Difficult/Intractable Posteriors

- Logistic Regression/ Single Neuron
- Neural Networks



How to solve Nasty Posterior Distributions

→ Approximate Methods:

- Variational Methods (i.e. VAE)
- Monte Carlo Methods

Connecting Bayes to the Optimization of a Single Neuron

General Bayes Formula – Coin Model

(Model) likelihood prior (Hallucinations)

$$p(\theta \mid y) = \frac{p(y \mid \theta)\, p(\theta)}{p(y)}$$

Posterior over parameters (weights)

normalization constant
aka. prob. of data

→ makes it sum to one

General Bayes Formula - Single Neuron

?

(Model) likelihood

?

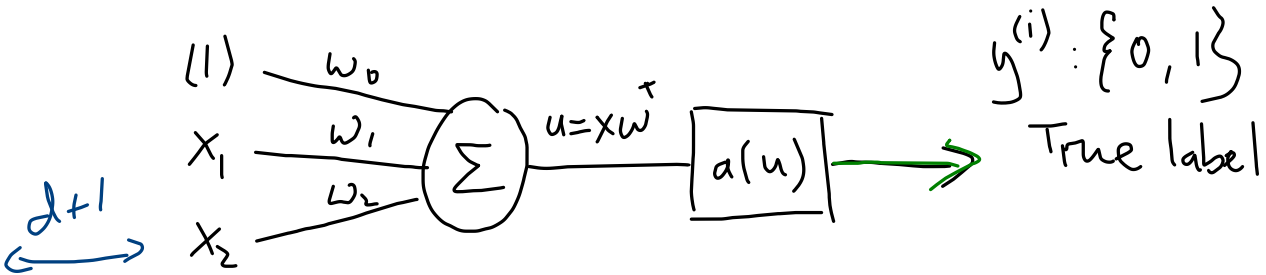
prior (Hallucinations)

$$p(\theta|x,y) = \frac{p(y|x,\theta)\,p(\theta)}{Z}$$

↑ Also adding features x

↖ normalization const? integral of numerator

Recap: Single Neuron Optimization



$$X = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} \\ \vdots & \vdots & \vdots \\ 1 & x_1^{(n)} & x_2^{(n)} \end{bmatrix} \quad n$$

$$w = \begin{bmatrix} w_0 & w_1 & w_2 \end{bmatrix}$$

$a(u) = \hat{y}$ Prediction

Learning as Optimization

Regularized Loss Function: $\ell_r(w) = \ell(w) + \alpha r(w)$

Gradient: $\nabla_w \ell(w)$

Update: $w \leftarrow w - \alpha \nabla_w \ell(w)$

From Regularized Loss Function to Bayes

$$\ell_r(w) = \ell(w) + \alpha r(w)$$

| expl...)

$$e^{-\ell_r(w)} = \boxed{e^{-\ell(w)} \cdot e^{-\alpha r(w)}}$$

(Model) likelihood prior (Hallucinations)

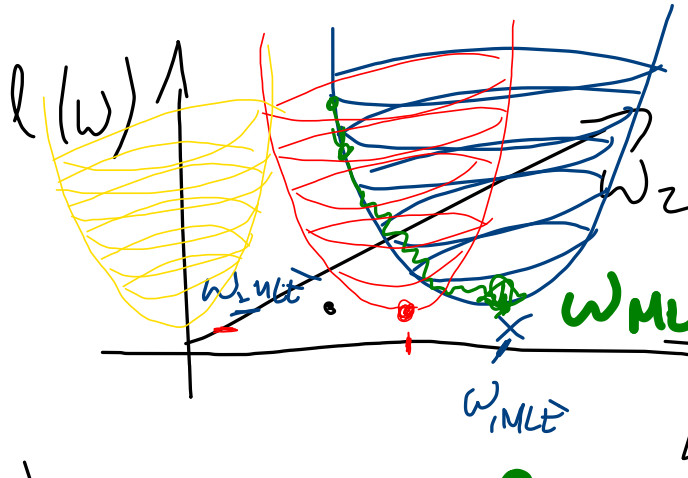
$$p(\theta|x,y) = \frac{p(y|x,\theta)\,p(\theta)}{Z}$$

Z ← normalization const.

integral of numerator

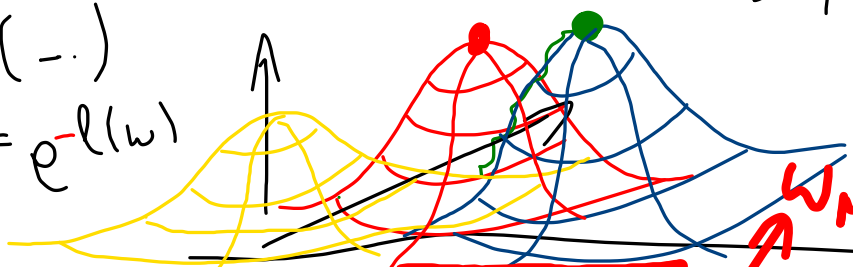
$$\int_{-\infty}^{\infty} e^{-\ell(w)} \cdot e^{-\alpha r(w)}\, dw$$

Remember this Connection?



w_{MLE} ⇝ Frequentist Point Estimate

$p(\cdot) = e^{-l(w)}$



w_{MAP} → Bayesian Point Est.

Regularizer

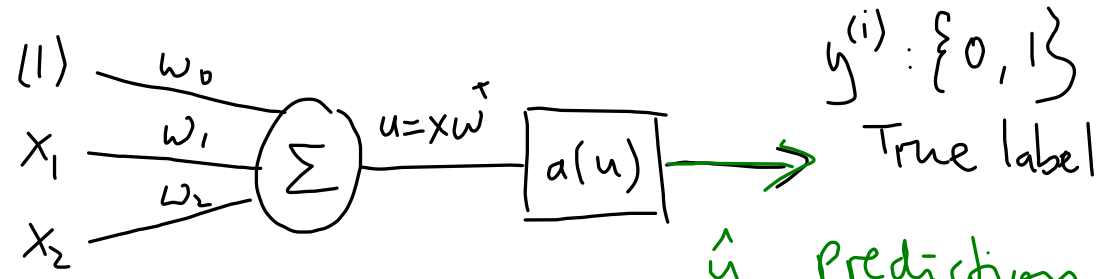
regularized probility

likelihood

Regularized Logistic Regression

Going Deeper into the Likelihood and the Prior of a Single Neuron

Mathematical Details: Single Neuron



$$X = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} \\ \vdots & \vdots & \vdots \\ 1 & x_1^{(n)} & x_2^{(n)} \end{bmatrix} \quad n$$

$$w = \begin{bmatrix} w_0 & w_1 & w_2 \end{bmatrix}$$

$$a(u) = \sigma(u) = \frac{1}{1 + e^{-xw^T}}$$

Learning as Optimization

$$\ell_r(w) = \ell(w) + \alpha r(w)$$

$$\ell_r(w) = -\sum_{i=1}^{n} \left[y^{(i)} \log\left(\hat{y}^{(i)}\right) + (1 - y^{(i)}) \log\left(1 - \hat{y}^{(i)}\right) \right] + \frac{\alpha}{2} w w^T$$

From Unregularized Loss to Likelihood

$$\ell(w) = \sum_{i=1}^{n} \left[y^{(i)} \cdot \log\left(\hat{y}^{(i)}\right) + (1-y^{(i)}) \log\left(1-\hat{y}^{(i)}\right) \right]$$

$$e^{-\ell(w)} = \prod_{i=1}^{n} \hat{y}^{y} \cdot (1-\hat{y})^{(1-y)}$$

Remember?

$$\text{Ber}(y|\theta) = \prod_{i=1}^{n} \theta^{y^{(i)}} \cdot (1-\theta)^{(1-y^{(i)})}$$

$$\underbrace{P(y|x,w)}_{\text{Likelihood}} = \text{Ber}\left(y^{(i)}|\hat{y}^{(i)}\right)$$

$$= \text{Ber}\left(y|\sigma(xw^{T})\right)$$

From Regularizer to Prior

$$\alpha r(w) = \frac{\alpha}{2} w w^T$$

$$e^{-\alpha r(w)} = e^{-\frac{\alpha}{2} w w^T}$$

Remember?

$$N(w \mid \mu = 0, \sigma = \alpha^{-1}) = \frac{1}{Z} e^{-\frac{\alpha}{2} w \cdot w^T}$$

$$P(\theta) = N(w \mid \mu = 0, \sigma = \alpha^{-1})$$

Gaussian Prior

Posterior over the Weights

$$P(w \mid x, y) = \frac{\mathrm{Ber}(y \mid \sigma(x w^T))\, \mathcal{N}(\mu_w, \sigma_w)}{P(y)}$$

Bayesian Prediction

$$P(w \mid x, y) = \frac{\text{Ber}(y \mid \sigma(x w^T))\, N(\mu_w, \sigma_w)}{P(y)}$$

$$P(y^* \mid D) = \int P(y \mid x^*, w) \cdot P(w \mid D)\, dw$$

x^*: new input

So far so good...

What's the problem now?

Nasty Likelihood Dist. and Intractable Integrals

$$P(w|x,y) = \frac{P(y|x,w) \cdot P(w)}{P(y)}$$

$$P(y) = \int_{-\infty} P(y|x,w) \cdot P(w)\, dw$$

Problem 1

Nasty Likelihood

Problem 2

Intractable Integrals

$$P(y^*|D) = \int P(y|x^*,w) \cdot P(w|D)\, dw$$

How to solve an intractable integral?

From Integral to Sample Mean

$$P(y^*|D) = \int P(y|x^*,w) \cdot P(w|D)\, dw$$

$$= \mathop{E}_{w \sim P(w|D)} \left[y(x^*, w) \right]$$

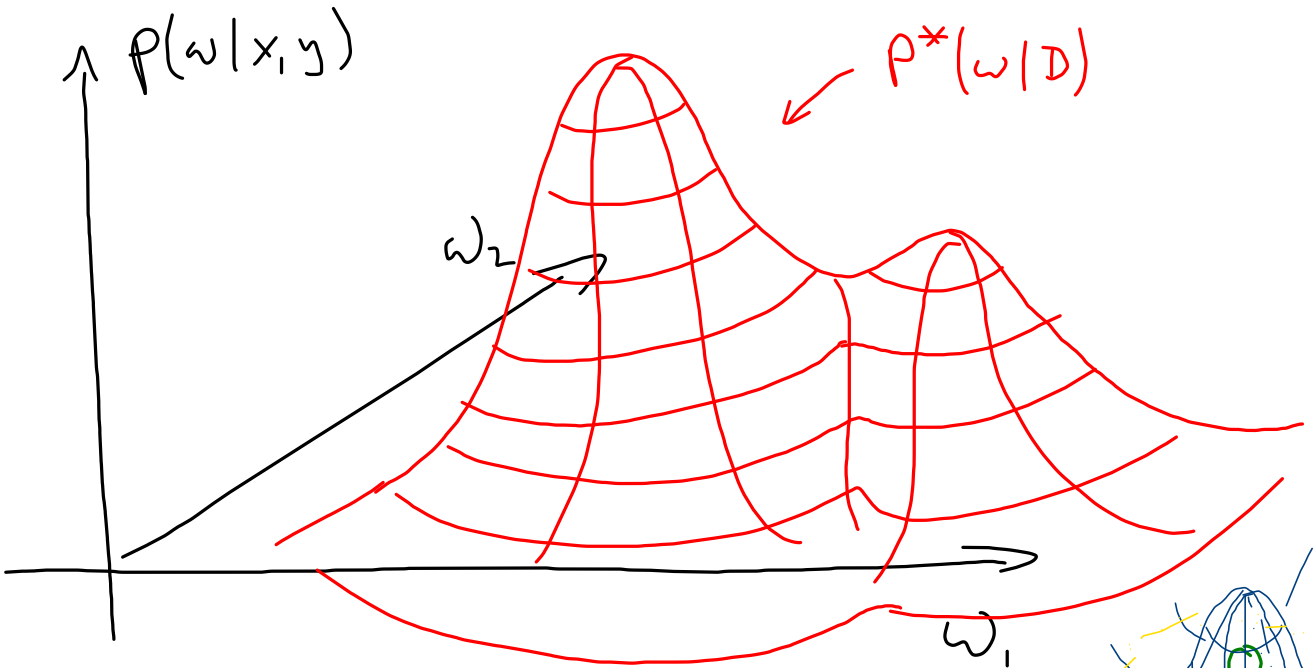
$$\approx \frac{1}{n} \sum_{i=1}^{n} y(x^*, w_i)$$

↑ given

where $w_i \sim P(w|D)$

↑ needed

Sampling from the Posterior: MCMC



$$P(w|D) = \frac{p^*(w|D)}{Z}$$

Pseudocode of MCMC

Algorithm 24.2: Metropolis Hastings algorithm

Initialize x^0 ; $\leftarrow \omega$

for $s = 0, 1, 2, \ldots$ **do**

  Define $x = x^s$;

  Sample $x' \sim q(x'|x)$;

  Compute acceptance probability

$$\alpha = \frac{\tilde{p}(x')q(x|x')}{\tilde{p}(x)q(x'|x)}$$ if Gaussian

  Compute $r = \min(1, \alpha)$;

  Sample $u \sim U(0, 1)$;

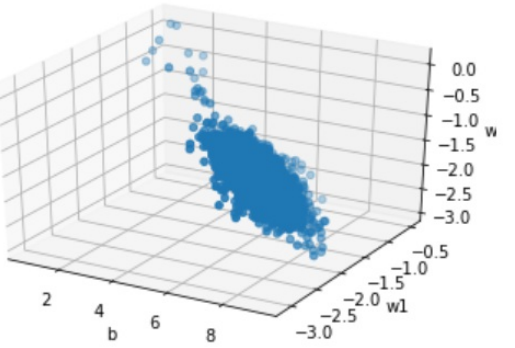
  Set new sample to

$$x^{s+1} = \begin{cases} x' & \text{if } u < r \\ x^s & \text{if } u \geq r \end{cases}$$

here $x = \omega$

Python Code

```
def MCMC(function, X, Y, N, d, sigma=0.1):
    mu0 = [0]*d
    cov = np.diag([sigma]*d)
    w_old = np.random.multivariate_normal(mu0, cov)
    sample_list = []
    for i in range(N):
        w_new = np.random.multivariate_normal(w_old, cov)
        a = np.exp(function(X, Y, w_new) - function(X, Y, w_old))
        u = np.random.uniform(0, 1)
        if (a > u):
            sample_list.append(w_new)
            w_old = w_new
    return np.array(sample_list)
```



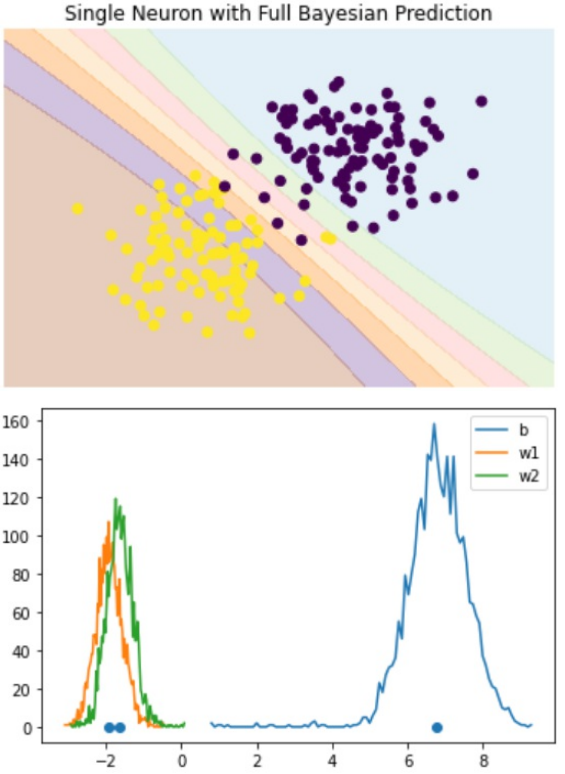

```
def sigmoid_layer(X, W):
    x_biases = np.full((X.shape[0],1), 1)
    X_ = np.concatenate((x_biases, X), axis=1)
    odds1 = np.exp(np.dot(X_, W.T))
    U = odds1/(1 + odds1)
    return U
def log_posterior_over_parameters_logreg(X, Y, W):
    sigma_prior = 1.5
    log_prior = (-np.dot(W, W.T)/(2*sigma_prior**2))
    likelihood = sigmoid_layer(X, W)
    full_log_likelihood = (Y.T) * np.log(likelihood) + (1 - Y.T) * np.log(1-likelihood)
    post = log_prior + np.sum(full_log_likelihood)
    return post

W_samples = (MCMC(log_posterior_over_parameters_logreg, X, Y, N=10000, d=3))
```

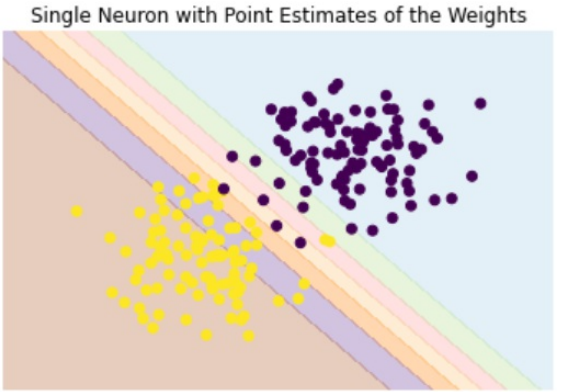
Python Code

```
def predict_MC(X, W_samples):
    return (np.nansum(sigmoid_layer(X, W_samples), axis=1)/W_samples.shape[0])
```

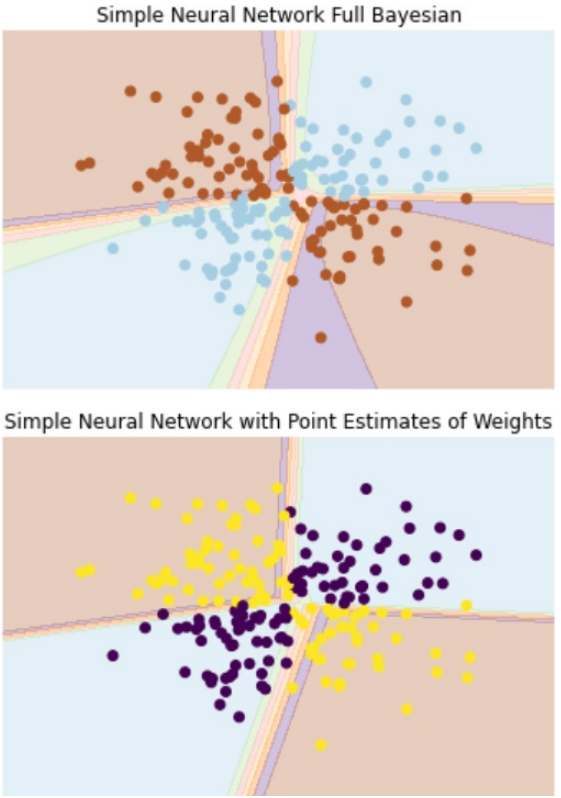
```
y_pred_full = predict_MC(X_pred, W_samples)
```




```
# POINT ESTIMATE
W_map = np.median(W_samples, axis=0).reshape(1, -1)
y_pred_map = sigmoid_layer(X_pred, W_map)
```

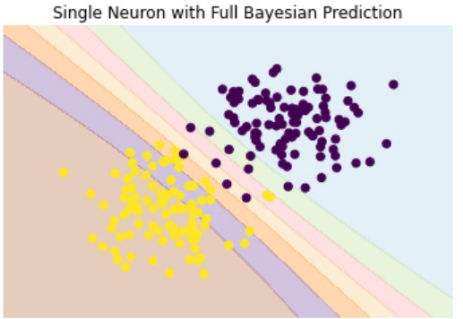


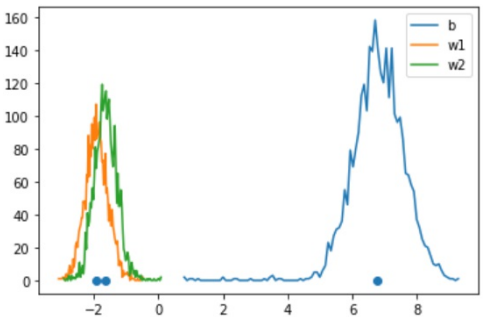
XOR Data with Simple Neural Network



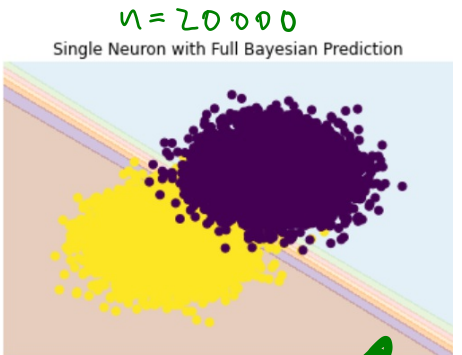
Dependence of Uncertainty on Sample Size

$n = 200$

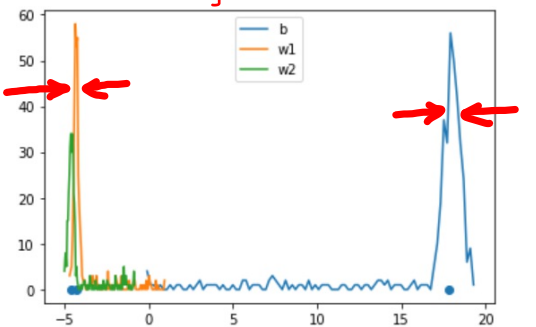




$n = 20000$

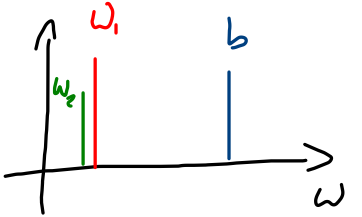


Uncertainty shrinks if $n \to \infty$



Looks like w_{MLE}

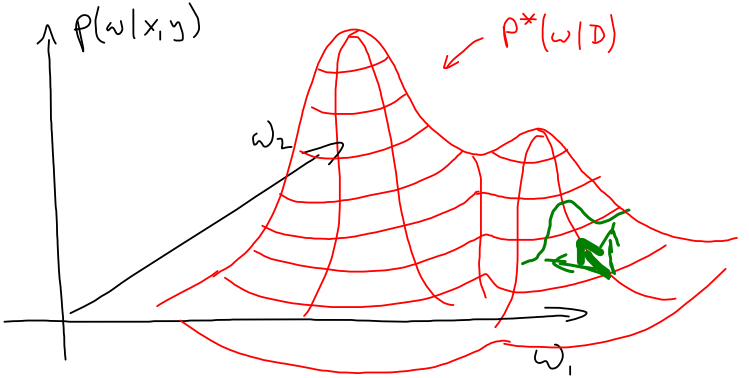
$$\lim_{n \to \infty} w_{MLE} = w_{MAP}$$



Future Work

→ Hamiltonian Monte Carlo ("Hybrid")

Optimization and Random Walk Hybrid

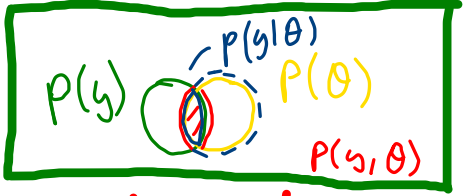


→ Speed

Understanding Numerator and Denominator of Bayes

(Optional?)

Rules of Probability



Creates a joint probability

Product Rule $P(y, \theta) = p(y|\theta)\, p(\theta)$

$$p(\theta|y) = \frac{p(y|\theta)\, p(\theta)}{p(y)} = \frac{P(y, \theta)}{P(y)}$$

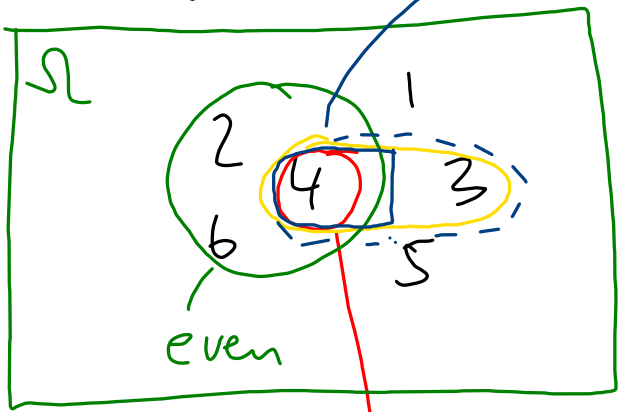
All possible y's

$P(\Omega) \rightarrow P(y)$

The reference system switches from $\Omega \rightarrow y$

Example

Die



$p(\text{even} \mid d12) = \frac{1}{2}$

divides of 12 "d12"

$p(d12) = \frac{1}{3}$

$p(\text{even}) = \frac{1}{2}$

both $P(\text{even}, d12) = \frac{1}{6} \leftarrow |\Omega|$ #4

$$P(d12 \mid \text{even}) = \frac{P(\text{even}, d12)}{P(\text{even})} = \frac{\frac{1}{2} \cdot \frac{1}{3}}{\frac{1}{2}} = \frac{1}{3}$$

$$P(\text{even} \mid d12) \cdot P(d12) + P(\text{even} \mid \text{not } d12) \cdot P(\text{not } d12)$$